An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations

Stefan Steinke, Markus Knauer, Lars Wehmeyer, Peter Marwedel

University of Dortmund, Computer Science 12
Otto-Hahn-Strasse 16, 44221 Dortmund, Germany
{steinke,knauer,wehmeyer,marwedel}@ls12.cs.uni-dortmund.de

Abstract. Power aware compilers have been under research during the last few years. However, there is still a need for accurate energy models for supporting software optimizations.
In this paper we present a new energy model on the instruction level. As an addition to former models, the bit toggling on internal and external busses as well as accesses to off-chip memories are considered.
To determine the characteristics, a measuring method is presented which can be used to establish the energy model without detailed knowledge of the internal processor structures.
Finally, the proposed energy model is established for the ARM7TDMI RISC processor.

1 Introduction

In recent years, the number of embedded systems has increased especially in the automotive, consumer electronic and communication sectors. New applications e.g. airbags, global positioning systems, mp3 players and PDAs based on processor controlled embedded systems are being developed. The percentage of software compared to the hardware components is steadily increasing since generating software is cheaper and more flexible. This is convenient for late changes in the development or even during the maintenance phase. The energy supply of these mobile systems is usually based on batteries, whose technology is also being improved, but at a slower rate. This has increased the importance of considering energy consumption during the design of embedded systems.

Hardware designers are steadily decreasing the size of chip structures and the supply voltages. Furthermore, system power management techniques implemented by the operating system are used to reduce energy consumption. However, due to the fact that the number of gates and the clock frequencies are constantly increasing, overall power consumption is still a limiting design factor.

Since the size of software is increasing, it becomes necessary to optimize the software with respect to its energy consumption. This can be achieved by modifying the algorithm of the high level application itself and by generating optimized machine code using a compiler.

Common compilers only optimize code with respect to performance or code size. These optimization goals are evaluated using a cost function. The cost function for

performance is based on the number of execution cycles whereas the cost function for code size uses the total number of instructions which defines the size of the program.

The optimization for energy is a recent development for compilers. For this optimization, the cost function is the product of time and power. In general, time depends on the number of executed cycles, the access time of memories and the clock frequency. Power is the product of the supply voltage and the current. The current depends on the technology of the processor, memory types and other energy consuming system components as well as the system activity. An energy model has to consider all these dependencies for a precise evaluation of overall energy consumption.

Following the presentation of related work in the next section, the energy model is presented with its properties and equations. A measurement technique for the determination of the model parameters is proposed in section 4. The technique is then used to generate an energy model for the ARM7 RISC processor [1] to demonstrate the practicability and the precision of the model. Section 6 concludes this contribution and mentions the ongoing work.

2 Related Work

One of the first instruction level power models was presented by Tiwari et al [10, 11]. All processor instructions are measured by execution within a loop. The measured current is the so called base cost. Costs caused by circuit state changes due to different instructions, called inter-instruction costs, are also considered. This model does not take into account other system components like memories. Especially for ultra low power processors with off-chip memory, a high amount of energy is spent in these memories. Since the compiler has an influence on the memory accesses, the energy consumption of the memories should be considered in the model.

The use of different coding techniques is another issue. Several techniques have been proposed [9] which optimize the bus coding and can be integrated into the compiler. It is therefore essential to compute the energy consumption caused by bit toggling in order to evaluate these coding techniques.

A power analysis of the ARM7 processor was published by Sinevriotis et al [8] based on Tiwari's power model. Several methods (e.g. scheduling for low power) have been proposed based on the analysis of base costs and inter-instruction costs.

Measurements with specialized equipment were done by Chang et al [3]. This equipment generates test data and stores the measured current value in a fast RAM. It is shown that simple power models and simulations are not sufficient to model the influence of '0's and '1's, i.e. the state of the bit lines.

Another energy model for simulation was presented by Simunic et al. [6, 7]. The processor is simulated in a cycle-accurate way and the current is computed as the sum of the current of all considered components. The components can be in one of two states at a time: active or idle. In each of these states, a fixed value for the current is assumed. This current is generally available from the vendor's data sheets. Simulation results were compared to actual measurements and a complete program's power dissipation can be estimated by the simulation with a precision of 5%. However, there is no distinction between individual instructions or states of the bus lines.

Other energy models which incorporate more details of the electrical effects require more detailed data about the internal design of the processor which is generally not available for common processors. Furthermore, the simulation results of these models are not very precise compared to the presented measurement.

3 Energy Model

3.1 Properties

An energy model needs to consider the following issues for use within a power aware compiler:

- different machine instructions
  During the code selection phase, the compiler has to choose one instruction sequence from a set of alternatives. Hence, the energy consumption of all instructions has to be included in the model.
- different instruction schedules
  When two subsequent instructions require different functional units, these units are activated or deactivated as required. The energy consumed due to these state changes can be minimized by optimizing the instruction schedule in such a way that one functional unit is used for a longer period of time. This optimization must take data and control flow dependencies into consideration.
- memory hierarchy
  In systems with two or more memories, e.g. on-chip and off-chip memory, the compiler has to decide on the location of memory objects. Therefore the differences in energy consumption have to be included in the energy model.
- bit toggling on busses
  The bit toggling on busses increases energy consumption. This effect has to be integrated into the model to support the compiler in optimizing the employed coding technique. Besides, the memory layout can be varied to minimize bit toggling.
- parameters of the energy model
  The aforementioned requirements cannot be satisfied by only using the data from datasheets. Therefore, electrical measurements have to be done. It is important that these measurements are simple to perform without requiring knowledge about the internal structure.
- reusability of the model
  A model should always be as general as possible. In this case, the model should be able to describe more than one specific processor.

3.2 Definition of Terms and Functions

For the definition of the energy model, several auxiliary functions are required:

$w(x)$**:** the number of bits in word x with the state '1' (weighted using parameter α_i)
$h(x, y)$**:** the number of bus lines with different state in x and y (Hamming distance, weighted using parameter β_i)

$BaseCPU(x), BaseMem(x)$**:** The main costs which are caused within the CPU or memory, respectively, by the execution of a single instruction x. The effects which are modelled more specifically by the functions w and h are subtracted and not included in this figure.

$FUChange(x, y)$**:** the cost for activating or deactivating a functional unit if first x, then y is executed

3.3 Definition of the Energy Model

The processor and its block structure including the functional units, internal and external busses as well as memories need to be specified. This can be done with the help of a datasheet. For the general specification, a RISC processor with a load/store architecture, Harvard architecture (different memories for instruction and data) and the functional units multiplier, barrel shifter and ALU was chosen. The block structure of this processor is shown in figure 1.

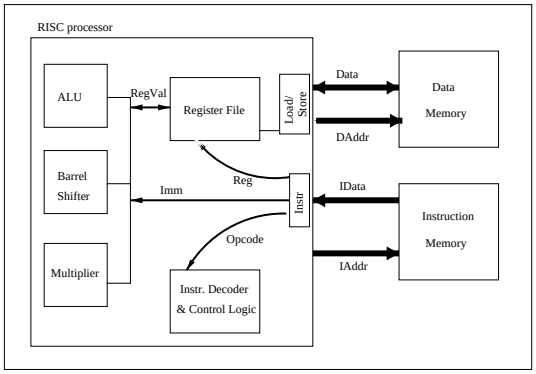

Fig. 1. block diagram of a system with a RISC processor and memories

The energy model consists of four parts:

1. instruction-dependent costs inside the CPU (E_{cpu_instr})
2. data-dependent costs inside the CPU (E_{cpu_data})
3. instruction-dependent costs in the instruction memory (E_{mem_instr})
4. data-dependent costs in the data memory (E_{mem_data})

These components can be summed up to the total energy consumption E_{total} which is caused by a sequence of instructions within the system consisting of processor and memory:

$$E_{total} = E_{cpu\_instr} + E_{cpu\_data} + E_{mem\_instr} + E_{mem\_data}$$

The instruction-dependent costs inside the CPU depend on the internal busses carrying the immediate value Imm, the register numbers Reg, values kept within the registers $RegVal$ and the instruction address $IAddr$. Except for $RegVal$ and $IAddr$, all these values are part of the instruction word.

For each parameter α_i, the dependency between the '0's and '1's and the energy consumption is modelled using function w. Function h in conjunction with parameter β_i models the energy consumption of toggling bits in subsequent instructions. Because instructions can generally include more than one immediate value and more than one register, the occurrences have to be summed up for each parameter.

The energy consumption for a sequence of m instructions (with s immediate values and t registers occuring in the instruction word) can now be determined as follows:

$$
\begin{aligned}
E_{cpu\_instr} = \sum_{i=1}^{m} \Big(& \\
& BaseCPU(Opcode_i) + \\
& \sum_{j=1}^{s} (\alpha_1 * w(Imm_{i,j}) + \beta_1 * h(Imm_{i-1,j}, Imm_{i,j})) + \\
& \sum_{k=1}^{t} (\alpha_2 * w(Reg_{i,k}) + \beta_2 * h(Reg_{i-1,k}, Reg_{i,k})) + \\
& \sum_{k=1}^{t} (\alpha_3 * w(RegVal_{i,k}) + \beta_3 * h(RegVal_{i-1,k}, RegVal_{i,k})) + \\
& \alpha_4 * w(IAddr_i) + \beta_4 * h(IAddr_{i-1}, IAddr_i) + \\
& FUChange(Instr_{i-1}, Instr_i) \Big)
\end{aligned}
$$

The data-dependent costs inside the CPU for n data accesses depend on the data address $DAddr$, the $Data$ itself and on the direction dir (read/write). Their are summed up as follows:

. It is summed up as follows:

$$
\begin{aligned}
E_{cpu\_data} = \sum_{i=1}^{n} \Big(& \alpha_5 * w(DAddr_i) + \beta_5 * h(DAddr_{i-1}, DAddr_i) + \\
& \alpha_{6,dir} * w(Data_i) + \beta_{6,dir} * h(Data_{i-1}, Data_i) \Big)
\end{aligned}
$$

The instruction-dependent costs in the instruction memory ($Word\_width$ = bit width of memory access) of m instructions are calculated using the following equation:

$$
\begin{aligned}
E_{mem\_instr} = \sum_{i=1}^{m} \Big(& \\
& BaseMem(InstrMem, Word\_width_i) + \\
& \alpha_7 * w(IAddr_i) + \beta_7 * h(IAddr_{i-1}, IAddr_i) + \\
& \alpha_8 * w(IData_i) + \beta_8 * h(IData_{i-1}, IData_i) \Big)
\end{aligned}
$$

The data-dependent costs in the data memory (direction of data transfer (read/write) = dir) of n data accesses are summed up as follows:

$$E_{mem\_data} = \sum_{i=1}^{n} \Big(BaseMem(DataMem, dir, Word\_width_i) + \alpha_9 * w(DAddr_i) + \beta_9 * h(DAddr_{i-1}, DAddr_i) \alpha_{10,dir} * w(Data_i) + \beta_{10,dir} * h(Data_{i-1}, Data_i) \Big)$$

4 Measuring Method

For the energy model of a specific processor the parameters α_1 to $\alpha_{10,dir}$, β_1 to $\beta_{10,dir}$, $BaseCPU$ and $BaseMem$ and $FUChanges$ have to be determined. The following method allows the consideration of processors without detailed information about the internal structure, since e.g. VHDL models are usually not available. Electrical measurements have to be taken, because the data available from the manufacturer is insufficient to determine the parameters.

The method proposed here assumes that the voltage does not change and does not depend on the execution of instructions. Based on this assumption, measuring the current is sufficient. This can be done by cutting the power supply pins of the processor and the memory. The measurements can be performed using a precise amperemeter.

Differences between instructions can not be measured with an amperemeter in a single run. Hence, each instruction i is measured individually. 100 instances of instruction i form the body of a loop. Using many instances is necessary in order to minimize the impact of the loop.

Different measurement sessions are necessary to determine all parameters of the energy model. First, for a certain i, parameter α_i is measured, because the parameters β_i, $BaseCPU$ and $BaseMem$ depend on this value. The determination of parameter α_i can be performed by measuring the currents drawn by instruction words or data words with a varying number of '1's.

The obtained results are evaluated with a linear regression method [5]. Linear regression analyzes the relationship between the current and e.g. number of '1's. A straight interpolating line is calculated which approximates all data points with minimal deviation. This interpolation is necessary due to errors caused by the precision of the amperemeter and the ambient temperature.

Following this procedure for α_i, the parameter β_i can be measured. Because this has to be done using bit toggles, two different instruction or data words have to be used in an alternating sequence to determine the Hamming distance. The effect of the number of '1's is subtracted from the measured data and a straight line is calculated using linear regression. This combination of starting the determination of α_i followed by the corresponding β_i is repeated for all values of i.

In the final step, $BaseCPU$ and $BaseMem$ are determined by subtracting the terms including α_i and β_i from the measured data according to the given equations.

This concludes the determination of all parameters necessary to build the energy model.

5 Results

In this section, the results for the ARM7TDMI processor AT91M40400 and the ATMEL Evaluation Board EB01 [2] are presented.

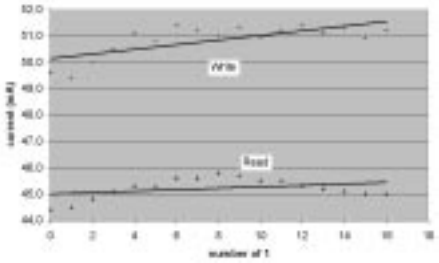

Fig. 2. CPU current depending on number of ’1’s on data bus

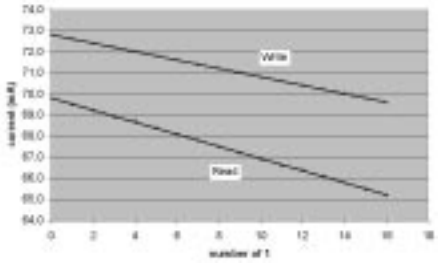

Fig. 3. memory current depending on number of ’1’s on data bus

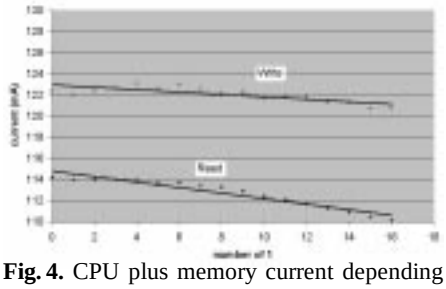

Fig. 4. CPU plus memory current depending on number of ’1’s on data bus

Note that not all parameters are relevant for all processors. This becomes clear for the used evaluation board by looking at the given figures.

The energy model is defined for the most general case, a Harvard architecture, whereas the used ARM7TDMI board employs a common instruction and data memory (von Neumann). Thus, the parameters are identical for data and instruction memory.

In figure 2, a positive regression coefficient $\alpha_{6,dir}$ can be seen. But in figure 3 the memory current results in a negative regression coefficient $\alpha_{10,dir}$. The result of both figures is a minor negative trend, shown in figure 4. This measurement stresses the necessity to incorporate the memory in the modelled system. If the processor had been analyzed in isolation, the wrong conclusion would have been a positive dependence.

Analyzing the energy consumption inside the CPU in conjunction with the Hamming distance on the data bus (figure 5) shows an important difference between read and write accesses. Read accesses do not depend on the Hamming distance, but write accesses do with a high regression coefficient $\beta_{6,dir}$. This is combined with the effect

inside the memory (figure 6) which is summed up in figure 7. Again there is a significant difference between the effects inside the CPU and in the sum of processor and memory.

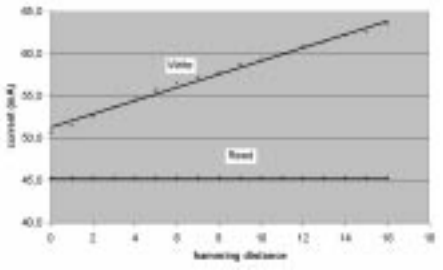

Fig. 5. CPU current depending on Hamming distance on data bus

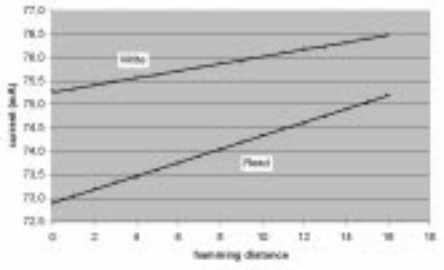

Fig. 6. memory current depending on Hamming distance on data bus

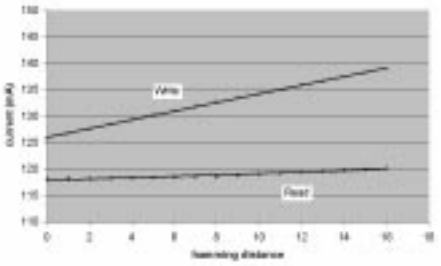

Fig. 7. CPU plus memory current depending on Hamming distance on data bus

Furthermore, there is a clear effect resulting from the number of '1's (figure 8) and the Hamming distance on the address bus (figure 9). There is a sizeable amount of energy which depends on the coding of the address bus. Coding strategies based on this energy model can be applied to optimize the energy consumption based on this effect.

The measurements and regression analysis were performed for all coefficients, the most relevant ones being shown in table 1.

Table 1. *parameters of ARM7TDMI energy model*

parameter	energy (pJ)		parameter	energy (pJ)	
	Read	Write		Read	Write
α_4, α_5	n.a.	48.0	β_4, β_5	n.a.	219.9
α_6	11.0	26.4	β_6	-5.5	224.1
α_7, α_9	n.a.	-19.2	β_7, β_9	n.a.	138.9
α_8	-115.3	n.a.	β_8	57.7	n.a.
α_{10}	-115.3	-60.4	β_{10}	57.7	22.8

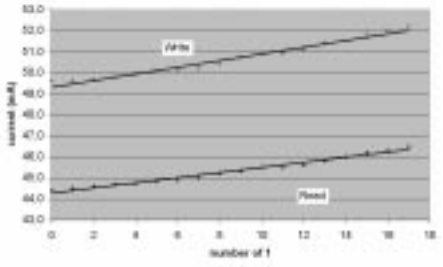

Fig. 8. CPU current depending on number of '1's on address bus

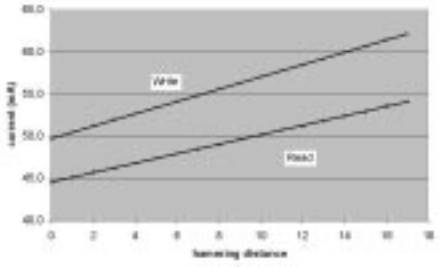

Fig. 9. CPU current depending on number of toggling bits on address bus

Beside the coefficients, measurements of instructions using different functional units were undertaken. The obtained results are presented in table 2. The energy amount required for activating or deactivating functional units is significant and therefore has to be part of the ARM7 processor model. For this specific processor it can be stated that the values for activating and deactivating are identical.

Table 2. *overhead for activating or deactivating functional units*

instruction i	instruction $i+1$	overhead for activating/ deactivating (mA)
ALU	Load/Store	2.2
Multiplier	Load/Store	2.5
BarrelShifter	ALU	3.3
Register File	ALU	3.8
Register File	Multiplier	2.1

Different series of measurements were undertaken to determine the parameters of the energy model for the ARM7TDMI. Beside of the execution of one single instruction, the measurements also include the execution of two different alternating instructions. Long sequences of different instructions do not deliver a constant value on the amperemeter and are therefore not possible with this measurement method. To verify that there are no additional effects when more than 2 instructions are considered, also a sequence of 12 instructions was measured in an endless loop. The sum calculated with the presented energy model shows a precision of 1.7% and justify the chosen approach.

6 Conclusion and Future Work

The instruction-level energy model presented in this work is based on the individual instructions executed on a processor. Additionally, the switching activity on busses which is responsible for a high amount of energy consumption was integrated into this model and supports the optimization of bit toggling on address and data busses by employing coding techniques. Furthermore, the use of different functional units is taken into account. This information can be used for optimizing the instruction schedule. Especially

for systems with low power processors, the off-chip memories have a high impact on the energy consumption and are therefore taken into consideration. This energy model is particularly well suited for RISC processors. In this work, we have shown in detail the adaption of the proposed model for the ARM7TDMI.

It can be used without detailed knowledge of the internal structure, since this information is generally not available for real processors. The measured precision of 1.7% is sufficient for this model to be used within an energy aware compiler.

Based on this energy model, the energy aware compiler `encc` [4] will be extended with new techniques using the capability of modeling the influence of bus toggling on the energy consumption. Several coding techniques can be integrated into the compiler to optimize the generated code. Furthermore, the influence of different memory types and the impact on code generation can be studied.

7 Acknowledgements

The authors would like to thank Michael Theokharidis for the measurements he performed in the course of his diploma thesis.

References

1. http://www.arm.com, Advanced RISC Machines Ltd.
2. AT91M40400 processor, http://www.atmel.com, ATMEL Corporation
3. Chang, N. and Kim, K. and Lee, H. G., "Cycle-Accurate Energy Consumption Measurement and Analysis: Case Study of ARM7TDMI", Proc. of International Symposium on Low Power Electronics and Design, 2000
4. `http://ls12-www.cs.uni-dortmund.de/~steinke/LOW_POWER/encc.html`
5. Motulsky, H.J., "Analyzing Data with GraphPad Prism", www.graphpad.com, GraphPad Software, Inc., San Diego CA, 1999
6. Simunic, T. and Benini, L. and De Micheli, G., "Energy-Efficient Design of Battery-Powered Embedded Systems", Proc. of International Symposium on Low Power Electronics and Design, 1999
7. Simunic, T. and Benini, L. and De Micheli, G., "Cycle-Accurate Simulation of Energy Consumption in Embedded Systems.", Design Automation Conference, pp. 876-872, 1999
8. Sinevriotis G. and Stouraitis, T., "Power Analysis of the ARM 7 Embedded Microprocessor", Proc. 9th Int. Workshop Power and Timing Modeling, Optimization and Simulation (PATMOS), Oct. 6-8 1999
9. Su, C.-L. and Tsui, C.-Y. and Despain, A., "Saving Power in the Control Path of Embedded Processors", IEEE Design & Test of Computers, Winter 1994
10. Tiwari, V. and Malik, S. And Wolfe, A., "Power Analysis of Embedded Software: A First Step towards Software Power Minimization", IEEE, Trans. On VLSI Systems, December, 1994
11. Tiwari, V. and Lee, M. T.-C., "Power Analysis of a 32-bit Embedded Microcontroller", VLSI Design Journal, Volume 7, No. 3, 1998