Improved Sorting Methodology of Data-processing Instructions

Andrii Borovyi[1], Volodymyr Kochan[2], Theodore Laopoulos[1], Anatoly Sachenko[2]

[1]Electronics Lab, Physics Dept., Aristotle University, 54124, Thessaloniki, Greece
e-mail: aborovyi@physics.auth.gr, laopoulos@physics.auth.gr
[2]Research Institute of Intelligent Computer Systems, Ternopil National Economic University
3, Peremoha Sq., Ternopil, Ukraine
e-mail: oko@tneu.edu.ua, as@tneu.edu.ua

Abstract

An improved classification methodology for sorting data-processing instructions for ARM7TDMI CPU core is presented in this paper. Main discussion is related to the process of creating appropriate training sets for neural network (NN) based estimation of power consumption. Proposed instructions' sorting according to the binary instruction representation and according to the resources being used for the overall system model. Separate instructions groups are obtained for NN-based estimation of power consumption. Experimental results of the proposed method confirm successful usage of this sorting methodology for providing high-accuracy estimation about power consumption.

1 Introduction

One of the problems, which require detailed analysis during the design of embedded and autonomous systems, is power-aware design. Latest publications in this area [1], [2] describe the need for the accurate power consumption estimation not only during the design process, but as well as during the normal usage of the final devices. Important advantage of this research is possible power-aware design not only for the hardware, but for the software also. Proposed approach is based on that well-known fact, that power consumption of the CPU is determined by the currently running routines.

There are known power-optimization methods [3] [4] which are based on the fastest running of the routines and putting microprocessor into power-saving mode (usually, by disconnecting unused blocks of the CPU). These methods are highly effective when the program must run few times, and it doesn't involve detailed analysis of CPU's analysis of operation according to the power-consumption.

In a previous publication of the same group [5] different measurement issues have been presented, along with results of measurement data analysis [6]. The estimation methodology was mainly presented in these papers, while the explicit data representation of the instructions was not examined. Consequently the error level of the analysis remained higher that it was expected, mainly due to the simple representation approach for the instructions' information. By contrast, an improved data sorting and analysis methodology will be represented in this paper aiming to complete the approach of using neural networks for the software-related power consumption estimation [7].

2 Analysis

Artificial neural networks are proposed to be used in the present approach. The well known ARM7TDMI CPU core will be used as a test bench in this work. The power consumption for all types of instructions within its 32-bit set will be explored. Data-processing instructions are forming the biggest instruction group within the ARM7TDMI ARM Instruction Set, as well as the most-used during the regular routines usage [8]. That's why these instructions must be analysed in first-time order.

In order to minimize amount of real measurements, and avoid using only simulations, different data-estimation methods have been used during previous work of the Ukrainian group of authors [9]. The estimation of the accuracy of the calculations and testing is complex, because the information about the measurement methodology is not very clear [10]. According to the results of the previous work it was decided to use preliminary sorting methodology related to the maximum available data (maximum bits amount, that are used during the instruction forming, processing and execution) as well as grouping various

instructions within separate subgroups, according to their power consumption performance. Because of these reasons the total amount of required real measurements is expected to decrease drastically.

The term "base power consumption" has been proposed in [10] for each instruction for simplifying estimations of the mathematical model development. It refers to power consumed by CPU during the execution of the instruction with its minimum values (zero values). Therefore, the appropriate amount of "basic" instructions has been written, according to the [7]. These instructions either described zero immediate values within the instruction or operated with the registers, which internal values has been equal to zero.

We have been trying to estimate power consumption of the CPU, using the approach that has been described above for the present research as well. Also we are taking into account only operation code and addressing mode, because all other values remained the same and didn't change. Doing so, all instructions of interest obtained decimal values (from 0 to 15) according to their OpCode value. Addressing modes were enumerated in the same way, but according to the order as mentioned in [10]. Power consumption estimation has been implemented within the MATLAB environment using “nntoolbox package”, and neural networks correspondingly. Yet, the results became unstable (maximum estimation error rapidly changed) and unacceptable (error changed from 10% to 1000%). Thus, extra research effort on a more efficient representation of obtained data is required.

Analysis of the obtained data has shown that internal features of the data representation are not taken into account during data processing. These features should be considered during the analysis of the instructions' binary code that is processing by the CPU core. All the instructions within this research are existent and valid for the 32-bit mode of the CPU (ARM mode) and their length is 32-bits. Therefore the detailed analysis of instructions' internal representation has been provided (with the disassembler that accompanies the ARM Developer Suite) and common bits were marked.

The analysis [9] discovered that gathering instructions into separate groups is an effective way in terms of power consumption. Using this fact, the following three groups were created (according to the second operand value):

1. instructions with the second operand as immediate value;
2. instructions with the second operand as register or register shifted to the immediate value;
3. instructions with the second operand as register or register shifted to the register.

We should note that proposed method for sorting instructions excludes a description of the redundant data, allowing instructions definition in fixed format, and decreasing noises amount for NN (Table 1).

Table 1. Preliminary results of CPU power consumption during the data-processing instructions.

Parameter	**Value**
Architecture of neural network	6-8-1
Vectors amount in training set	12
Vectors amount in verification set	12
Mean square error	10^{-8}
Training epochs	20785
Average training error	0,0 %
Maximum training error	0,1 %
Average estimation error	2,7 %
Maximum estimation error	5,2 %

Despite of the acceptable value of NN's average estimation error, as in shown in Table 1, the maximum estimation error remains high, therefore training set must be analysed further in order to remove data with noise. The ARM Assembler syntax as well as data binary representation has been taken into account during the creation of the new training set. According to the syntax the three following subgroups were created:

1. instructions where all three operands are used for data processing;
2. instructions where both the destination and the second operands only are used for data processing;
3. instructions where both the first and the second operands only are used for data processing.

Thus, after the analysis nine subgroups for data-processing instructions were formed, according to the syntax and the maximum length of the second operand to the each of those has its own unique structure and requires separate NN for power estimation.

Such grouping may be seen in Table 2.

Table 2. Data-processing instructions, according to predefined criteria

		Registers used within the instructions		
		All registers	A destination and a second register	A first and a second register
Amount of used bits	96 bits	ADD, SUB, RSB, ADC, SBC, RSC, AND, ORR, EOR, BIC	MOV, MVN	CMP, CMN, TST, TEQ
	128 bits			
	160 bits			

3 Architecture of neural network

According to the above analysis, the appropriate architecture for the neural network is developed. The input layer has 6 neurons, the hidden one - 4 neurons and output layer has 1 neuron. Its output value will correspond to the power consumption of the CPU. Multilayer perceptron has been used as non-linear activity function. This model is very simple, as well as universal model for the estimations [11], [12]. Output value for the three-layer perceptron can be defined with the next equation:

$$y = F_3\left(\sum_{i=1}^{N} w_{i3} h_i - T\right),$$

where N — amount of the neurons at hidden layer, w_{i3} — weight of the synapse from hidden neuron i, to the output layer, h_i — output value of the neuron i, T — threshold for output neuron, and F_3 — activity function for the output neuron.

Output value for hidden neuron j can be estimated with the next equation:

$$h_j = F_2\left(\sum_{i=1}^{M} w_{ij} x_i - T_j\right),$$

where w_{ij} — weight coefficient for connection between input neuron i and hidden neuron j, x_i — input values, and T_j — threshold of hidden neuron j. Activity function for the hidden layer will be sigmoid, and activity function for the output layer will be linear function with the k coefficient [13].

The back-propagation algorithm [14] has been used as training algorithm. This algorithm is based on gradient decreasing method and provides iterating procedure for renewal of weights and thresholds for each vector p from the training set:

$$\Delta w_{i,j} = -\alpha \frac{\partial E^p(t)}{\partial w_{i,j}(t)},$$

$$\Delta T_j(t) = -\alpha \frac{\partial E^p(t)}{\partial T_j(t)},$$

where α training coefficient, $\frac{\partial E^p(t)}{\partial w_{i,j}(t)}$ and $\frac{\partial E^p(t)}{\partial T_j(t)}$ — function error gradients at iteration t for training vector p, $p \in \{1,...,P\}$, where P — size of the training set.

Mean-square error for iteration t is estimating according to the next equation:

$$E^p(t) = \frac{1}{2}\left(y^p(t) - d^p(t)\right)^2,$$

where $y^p(t)$ is outputting value for iteration t, and $d^p(t)$ — target outputting value for the training vector p.

During the training process, overall error is estimated as:

$$E(t) = \sum_{p=1}^{P} E^p(t)$$

The steepest descent method for calculating the learning rate [13] is used for removing the classical disadvantages of the back propagation error algorithm. Thus, the adaptive learning rates for the logistic and linear activation functions are given, respectively, by:

$$\alpha(t) = \frac{4}{\left(1 + \left(x_i^p(t)\right)^2\right)}$$

$$\frac{\sum_{j=1}^{N} \left(\gamma_j^p(t)\right)^2 h_j^p(t)\left(1 - h_j^p(t)\right)}{\left(\sum_{i=1}^{N} \left(\left(\gamma_j^p(t)\right)^2 h_j^p(t)\right)\right)^2 \left(1 - h_j^p(t)\right)^2}$$

$$\alpha(t) = \frac{1}{\sum_{i=1}^{N} \left(h_i^p(t)\right)^2 + 1},$$

where, for the training vector p and iteration t, $\gamma_j^p(t)$ — is the error of neuron j and h_j^p — is the input signal of the linear neuron.

The error of neuron i with logistic activation function can be determined by the expression:

$$\gamma_j^p = \sum_{j=1}^{N} \gamma_3^p(t) w_{i3}(t) h_i^p(t)\left(1 - h_j^p(t)\right),$$

where $\gamma_3^p(t) = y^p(t) - d^p(t)$ is the error of the output neuron, w_{i3} — is the weight of the synapses between the neurons of the hidden layer and the output neuron.

The described algorithms of NN have been implemented in routine for estimating power consumption of the CPU.

4 NN verification results

The next step in the described procedure is to form groups according to the proposed method of the sorting instructions and taking into account available data about CPU's "basic" power consumption during the data-processing instructions execution:

1. Arithmetic-logic instructions (10 instructions):
   1.1.96 bits - 10 vectors;
   1.2.128 bits - 60 vectors;
   1.3.160 bits - 40 vectors.
2. Movement instructions (2 instructions):
   2.1.64 bits - 2 vectors;
   2.2.96 bits - 12 vectors;
   2.3.128 bits - 8 vectors.
3. Comparing and testing instructions (4 instructions):
   3.1.64 bits - 4 vectors;
   3.2.96 bits - 24 vectors;
   3.3.128 bits - 16 vectors.

Taking into account, that maximum available data for creating training and verification sets have been obtained for subgroup 1.3, it has been decided to perform power analysis for this subgroup.

Research results are represented in Table 3.

Table 3. NN-prediction of power consumption

Architecture of neural network	20
Vectors amount in training set	20
Vectors amount in verification set	2
Mean square error	13
Training epochs	10^{-6}
Average training error	11444
Maximum training error	0,20%
Average estimation error	0,60%
Maximum estimation error	1,80%
Architecture of neural network	5,30%

Comparing results, provided in Tables 1 and 3, it is observed, that proposed sorting methodology provides decreasing of average predicting error up to 1,8% that is much better than in the previous results.

Partial output of the NN verification is provided in Table 4.

Table 4. Partial verification results for the NN

Number	Real	Predicted	Abs. Err.	Rel. Err.
22	0,926	0,894	-0,032	3,50%
23	1,01	1,066	0,054	5,30%
24	0,82	0,810	-0,005	0,60%
25	1,12	1,118	0,002	0,01%

5 Conclusions and future works

Improved data processing methodology has been proposed. This methodology is based on accounting Assembler syntax of the instruction as well as used resources. Gathering instructions into groups provided exclude extra "noises" from the data, describing only required resources. NN-based estimation of power consumption confirmed decreasing of average estimation error. Increasing of maximum estimation error for 0,1% can be explained with the absence of exact description of all the fields of the instructions (in terms of data).

Future research will be focused on estimating power consumption while the instructions will be in different states, not only in "basic" one. It may implement deeper usage of NN, providing more accurate estimation.

References

1. Nikolaidis S., Chatzigeorgiou A., and Laopoulos Th., "Developing an Environment for Embedded Software Energy Estimation", Computers, Standards and Interfaces, Vol.28, N.2, 2005
2. Kavvadias N., Neofotistos P., Nikolaidis S., Kosmatopoulos C., and Laopoulos Th., "Measurements Analysis of the Software-Related Power Consumption of Microprocessors", IEEE Transactions on Instrumentation and Measurement, Vol.53, N. 4, 2004
3. Carlo Brandolese, William Fornaciari, and Fabio Salice. Ultra Low-Power Electronics and Design, chapter Source-Level Models for Software Power Optimization, pages 156–171. Politecnico di Torino, Italy, 2004.
4. M. F. Jacome, A. Ramachandran. Power Aware Embedded Computing // Embedded Systems Handbook Zurawski, R. (ed.) CRC Taylor & Francis, 2006, P. 16-1 - 16-17
5. A. Borovyi, V. Konstantakos, V. Kochan et al. Analysis of CPU's instructions energy consumption device circuits // The fourth IEEE international workshop on Intelligent Data Acquisition and Advancing Computing Systems (IDAACS 2007): Proceedings. — Dortmund, Germany, September 9–11, 2007. —P. 42—47. — ISBN 978-1-4244-1347-8
6. A. Borovyi, V. Konstantakos, V. Kochan et al. Using Neural Network for the Evaluation of Power Consumption of Instructions Execution // The Fifth International Instrumentation and Measurement Technology Conference (I2MTC'2008): Proceedings. — Vancouver Island, Victoria, British Columbia, Canada, May 12—15, 2008. — P. 676—681.
7. ARM Limited, editor. ARM Architecture Reference Manual. Number ARM DDI 0100I. ARM Limited, 2007.
8. S. Segars. ARM7TDMI Power Consumption. IEEE MICRO, 17(4):12–19, July– Aug. 1997.
9. A. Borovyi, O. Havryshok, V. Kochan, Z. Dombrovsky Development Problems of The CPU Power Consumptionm Model // Proceedings of the 10th International Scientific Conference "Modern Information and Electronic Technologies", May 18-22 2009, Ukraine. Vol. 1, P. 157 *(in Ukrainian)*
10. Nikolaidis S., Kavvadias N., Laopoulos T., Bisdounis L., Blionas S., "Instruction-level energy modeling for pipelined processors", Journal of Embedded Computing (special issue on Low-Power Design), Cambridge International Science Publishing (CISP), N.3, 2004
11. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators// Neural Networks.- 1989.- #2, P.359-366.
12. Simon Haykin. Neural Networks and Learning Machines, 3rd Edition, Prentice Hall, 2008. 936 pages.
13. V. Golovko, Neural Networks: training, models and applications. Radiotechnika. Moscow, 2001, P. 256. *(In Russian)*.
14. D. Rumelhart, G. Hinton, R. Williams. Learning representation by back-propagation errors // Nature.- #323.- 1986.- P. 533-536.