Energy Consumption Estimation of Software Components based on Program Flowcharts

Patrick Heinrich, Hannes Bergler, Dirk Eilers
Fraunhofer Institute for Embedded Systems and Communication Technologies ESK, Munich, Germany
{forename.surname}@esk.fraunhofer.de

Abstract—This paper presents and evaluates a new approach of energy estimation for single software components based on program flowcharts. This estimation is designed to be applicable early in the design process, which enables system designer to evaluate different design variants with respect to the energy consumption of the later system. The energy estimation model is based on individual flowchart elements and execution probabilities for branches and iterations. The used flowchart elements are for arithmetical calculations, flow control and reads/writes, which are a selection of possible elements used to show the feasibility of the approach. The estimation model is verified in a first step by using three commercially available benchmarks. The flowcharts of these are utilized to estimate the energy consumption by using the presented model. The comparison between estimated and measured energy consumption of an exemplary embedded system results in an estimation error bandwidth between -11.9 % and +6.9 %. The main benefit of the presented approach is the applicability within the development phase “System Design” [1], i.e. previous to any software implementation. This is realized by using only available information of that development phase and generic elements to estimate the energy consumption.

Keywords—energy estimation, energy-efficiency, embedded, adaptive systems, automotive electronics

I. INTRODUCTION

As embedded systems become more prevalent and powerful, they are consuming more energy. This paper focuses on estimating energy consumption in embedded systems during the early stages of design. Our research concentrates on the area of embedded systems commonly found in automobiles, aircrafts and industrial systems. Embedded systems designers, such as those active in the automotive, aerospace or other industries, are frequently given energy consumption requirements for the finished product. This makes it necessary to estimate the energy consumption early in the design process to realize the evaluation of different designs with respect to energy consumption. Because most industrial domains such as the automotive industry have especially long development cycles, a lot of hardware and software choices must be made very early during the design process [2].

This paper presents an energy estimation model for single software components. This estimation uses information of program flowcharts, which are composed of generic flowchart elements. The focus of this work is to enable the energy consumption estimation within early phases of system development as suggested within [1], since most of the existing energy estimation models are not applicable. This is caused by too less available information within the early phases of system development processes.

The paper is structured as follows: Section II discusses different existing energy estimation models. The energy estimation approach is presented in Section III and evaluated in Section IV. The paper closes with conclusion and future work in Section V.

II. RELATED WORK

Energy estimation techniques are categorizable concerning their level of abstraction. Ibrahim [3] identifies two main abstraction levels for models that rely on the running system: The low-level power modeling with focus on hardware and the high-level techniques with focus on software. Models which do not rely on running systems are part of a further (higher) abstraction level, where the presented model is included.

The low-level abstraction level of [3] includes the circuit-, gate-, register-transfer- and the micro-architecture-level. These estimation techniques use detailed electrical descriptions of the processor to estimate the energy consumption. These techniques are used for example for processor development. The high-level abstraction level contains two categories: the Instruction Level Power Analysis (ILPA) and the Functional Level Power Analysis (FLPA). [4] introduced the ILPA, which is based on the energy consumption of assembler instructions including inter-instruction effects. The maximum estimation error rate of this technique is specified as 3 %. The estimation technique based on FLPA was introduced by [5], where the functional elements (e.g. processing unit, memory management unit) of processor cores are individually analyzed concerning the usage by assembler instructions. [6] extended the approach to the programming language “C”. The reached estimation errors of these approaches are 4.2 % analyzing “C” code and 1.8 % for assembler code. However, these estimation techniques require the existence of the final source code, which is normally not available within the development phase “System Design” [1]. Another missing aspect is the different possible program flows during execution which influence the energy consumption of the program, viewed as a whole.

The abstraction level of models, which do not rely on running systems, such as the presented approach, enables energy estimation without the presence of source code. Existing approaches use Petri nets to enable the modeling of program...
flows. [7] uses stochastic Petri nets transformed from UML\textsuperscript{1} models with MARTE profile to describe the program flow of software. At first sight this seems to be a similar approach, however, existing source code is divided into source code specific unbranched program blocks of software and provided with energy consumption and the transition times between those. However, this approach needs to know the energy consumption of the individual program blocks. These blocks are not generic and so individual measurements are necessary – previous to the estimation of the energy consumption of the system. [8] also uses Petri nets to simulate the program flow. The nets are automatically generated from assembler code. After that the execution probabilities of branches and iterations must be added by the system designer. This approach results in an estimation accuracy of 93 % for a program in whole.

The presented energy estimation model of this paper is designed to estimate the energy consumption based on generic flowchart elements. The energy consumption of the generic elements is measurable independent from the specific application and is given, for example, by the processor manufacturer. Details are presented and analyzed within the next section.

III. ENERGY ESTIMATION IN EARLY DESIGN PHASES

Energy estimation during early stages of the development process enables system designers to evaluate their design choices with respect to the energy consumption of the later system. The challenge of early energy estimation is the database on which the energy estimation is done. To obtain precise estimations, most models need information, which are not available during the early phases of system development as discussed within Section II. Information commonly available previously to the final source code are program flowcharts, which represent algorithms or processes of software components. Flowcharts and the used symbols were standardized within ISO 5807 [9] and were designed to be independent from hardware and software. The presented energy estimation approach uses these flowchart elements specified by generic programming language elements. These elements and the estimation approach are described within the following subsections.

A. Program Flowcharts for Energy Estimation

The presented energy estimation approach focuses on single software components, which are executed on embedded systems and written in the programming language “C”. The basic idea of the approach is to use input, which is abstracted from the source code. Flowcharts are able to represent the software structure (program flow) and the individual flowchart elements encapsulate the execution of different programming elements. In the first step this approach is realized using elements very close to the programming language to demonstrate the feasibility of the approach. These elements consist of a set of generic flowchart elements specified by generic programming language elements (cf. Table I). Intentionally not all available constructs of the programming language “C” were used, because having all available constructs and creating flowcharts with these would be equivalent to writing source code. The selected elements are generic enough to be usable for program flowcharts in the development phase “System Design” and are commonly known previous to the software implementation. However, this leads to three main challenges:

- Estimating energy consumption at an abstracted level (e.g. flowcharts) means that a lot of elements are not represented. For example where data is stored, background processes, etc. This means that this kind of energy estimation normally results in too small energy estimations, which must be compensated.
- The second challenge is the mostly unknown number of iterations or the result of conditional branches, caused by unknown input data. This means the program flow varies and therefore the energy consumption of the software component in total. Large estimation errors are possible through that.
- The energy consumption of the selected flowchart element is not constant. It is still influenced by different factors, e.g. the energy for mathematical operations is influenced by the selected data type or even by the number of zeros and ones within the operands [4]. This results in estimation errors, even when flowchart and source code are a perfect match.

The energy consumption of the different flowchart elements depends on the used hardware. This makes it necessary to know the energy consumption of the generic flowchart elements to enable the energy consumption estimation. This could be realized by an own database or the values are provided by the hardware manufacturer. It is not necessary to measure these values on the final hardware, this would reduce the benefit of this estimation approach. The following subsection depicts the individual elements of the energy estimation approach.

B. Energy Estimation Model

The energy estimation is based on individual energy estimation elements as described within subsection III-A. These elements are used within Equation 2 to estimate the energy consumption of programs represented by program flowcharts (symbols described within Table I). The final energy estimation of a software component is shown in Equation 1, where $E_{flow}$ is the energy consumption of the program flowchart corrected by the corrective factor $\tau_{corr}$. This factor represents all neglected energy consumers, e.g. not analyzed operations such as bit shifting or even energy consumption by caching, pipelining, etc., which is commonly not determinable within the development phase “System Design”. For instance the value of this factor is application-specific, i.e. the characteristics of the software such as data-, calculation- or control-intensive, and is determinable after a training phase of the energy estimation algorithm for different kinds of applications.

\begin{equation}
E_{SW_{comp}} = E_{flow} \cdot \tau_{corr} \tag{1}
\end{equation}

\begin{equation}
E_{flow} = \sum_{i=0}^{n_{add\_sub}} E_{add\_sub,i} + \sum_{i=0}^{n_{mult}} E_{mult,i} + \sum_{i=0}^{n_{div}} E_{div,i} + n_{inc} \cdot E_{inc} + \sum_{i=0}^{n_{f\_if}} E_{f\_if,i} + \sum_{i=0}^{n_{loop}} E_{loop,i} + \sum_{i=0}^{n_{rw}} E_{rw,i} \tag{2}
\end{equation}

\textsuperscript{1}Unified Modeling Language
In the following, the energy consumption symbols used within Equation 2 are detailed. Intentionally not all available constructs of the programming language “C” were used, because having all available constructs and creating flowcharts with these would be equivalent to writing source code.

The energy consumptions of the arithmetical operations $E_{\text{add,sub}}$, $E_{\text{mult}}$ and $E_{\text{div}}$ are distinguished between the data types used for the operation. Caused by the different calculation complexities, the energy consumption is a multiple, for example comparing “integer” and “floating point” calculations. In the context of this paper the elements are subdivided into operations with data types “integer” and “floating point” (here: double). Equation 6 shows exemplary the calculation of the energy consumption of $E_{\text{add,sub}}$, $E_{\text{mult}}$ and $E_{\text{div}}$. The symbols $n_i$, specified by different indexes, contain the number of the respective elements. The energy consumption for incrementing integer values is included in the energy estimation equation, because this is a very common operation within embedded system software. In particular, this operation is used within “for” and “while” loops as counter. The energy consumption for increment operations $E_{\text{inc}}$ is lower than for addition and subtraction, because just one variable is manipulated.

$$\sum_{i=0}^{n_{\text{add,sub}}} E_{\text{add,sub},i} = n_{\text{add,sub},\text{int}} \cdot E_{\text{add,sub},\text{int}} + n_{\text{add,sub,dbl}} \cdot E_{\text{add,sub,dbl}} \quad (3)$$

One of the main influences concerning the energy consumption of software components in total is the execution flow, which is influenced by conditional branches (e.g. “if”) and iterations (e.g. “for” and “while”). The challenge is the necessarily to concern all relevant branches and iterations within the energy estimation. The presented estimation equation solves this by introducing the factor $\omega$. This factor is used in the context of “if” branches to define the probability of every branch to be executed ($\omega_{if}$). For “for” and “while” iterations the factor is used to give the probable number of iterations ($\omega_{\text{loop}}$). It is possible to mark uncertain values with an interval of error (or confidence value), which is then propagated to the final estimate to indicate the error bandwidth resulted by the $\omega$ factors.

The $\omega$ factors must be defined by the system designer during the creation of the program flowchart or at least previous to the energy estimation of the software component. These factors induce the risk of large estimation errors caused by incorrect values, which requires a careful choice by the designer. Equation 5 represents the energy estimation of “if” branches, which include the energy consumption of the condition check ($E_{\text{if,head}}$), the “if” branch itself ($E_{\text{if,branch}}$) and the “else” branch ($E_{\text{else,branch}}$). Within every branch the full set of possible flowchart elements may be represented, as defined in Equation 2. The energy consumption of $E_{\text{if,head}}$ is given by energy consumption necessary for comparing values. Equation 5 shows the estimation of loops like “for” and “while” as discussed above. The content of the variables $E_{\text{loop,if}}$, $E_{\text{loop,head}}$ and $E_{\text{loop,body}}$ depends on the used energy estimation elements by the system designer.

$$\sum_{i=0}^{n_{\text{if}}} E_{\text{if,},i} = \sum_{i=0}^{n_{\text{if}}} E_{\text{if,head},i} + \omega_{if,i} \cdot E_{\text{if,branch},i} + (1 - \omega_{if,i}) \cdot E_{\text{else,branch},i} \quad (4)$$

$$\sum_{i=0}^{n_{\text{loop}}} E_{\text{loop},i} = \sum_{i=0}^{n_{\text{loop}}} E_{\text{loop,head},i} + \omega_{\text{loop},i} \cdot (E_{\text{loop,head}} + E_{\text{loop,body}}) \quad (5)$$

Every software component consumes energy by reading and writing variables. The energy consumption differs depending on the data type, caused by different representation of data and for different number of necessary bytes, e.g. for variables of type “double”. The equation for this estimation is structured such as Equation 3. (Note: Only reads and writes to internal memory are analyzed within this paper.)

The following section evaluates the presented energy estimation approach using an exemplary embedded system.

### IV. Evaluation

The microcontroller used for the evaluation has a MIPS32 architecture and is equipped with six processor cores. Every core is equipped with L1 instruction cache and L1 data cache of 16 kB. All cores share a L2 cache of 256 kB. The software executed for evaluation is running on just one processor core. The microcontroller has four power supplies for analog core (1.5 V), digital core (1.5 V), DDR2 SDRAM phy (1.8 V) and pads (3.3 V). A DC Power Analyzer N6705B of Agilent Technologies, Inc. is used to measure the energy consumption. The accuracy is 0.016 % + 15 mV and 0.04 % + 15 μA. The maximum sampling rate is 48.9 kHz. The source code of the benchmarks is compiled using the GNU Compiler (GCC)\(^2\) in version 4.5.2 with deactivated optimization options.

\(^2\)http://gcc.gnu.org
Benchmark Consortium EEMBC. The benchmark “a2time” is control-intensive and mainly consist of basic arithmetic operations and conditional branches. The set of data used for the calculations is composed of integer values. The benchmark “rspeed” uses also a set of integer values as input data, but the number of arithmetic operations is less than half of the benchmark “a2time”. Most of the operations of the benchmark “basefp” are floating point instructions (double), i.e. the benchmark is calculation-intensive. The number of instructions is about five times larger compared to “rspeed”, and the set of data is about four times larger.

The measured values of the energy consumption parameters of the system are presented within Table II. Within this paper the value of $\tau_{corr}$ is 1.23, which results from the measurement results of Section IV-A to correct unrepresented energy consumers. As discussed within Section III-B this factor is usually given e.g. after a training phase to learn the estimation error or is previously known by the kind of application.

Table II. PARAMETERS OF THE ENERGY ESTIMATION FOR THE EVALUATED SYSTEM

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{rw,int}$</td>
<td>22.714 nJ</td>
<td>$E_{mult,int}$</td>
<td>56.436 nJ</td>
</tr>
<tr>
<td>$E_{rw,dbl}$</td>
<td>32.197 nJ</td>
<td>$E_{mult,dbl}$</td>
<td>650.138 nJ</td>
</tr>
<tr>
<td>$E_{add,int}$</td>
<td>42.574 nJ</td>
<td>$E_{add,int}$</td>
<td>140.453 nJ</td>
</tr>
<tr>
<td>$E_{add,dbl}$</td>
<td>390.438 nJ</td>
<td>$E_{add,dbl}$</td>
<td>1557.194 nJ</td>
</tr>
<tr>
<td>$E_{inc}$</td>
<td>31.658 nJ</td>
<td>$E_{rs}$</td>
<td>40.401 nJ</td>
</tr>
</tbody>
</table>

Table III shows the number of energy parameters within the three benchmarks including the real energy consumption and the estimation error. The bandwidth of error is between -11.9 % and +6.9 %. Obviously the benchmark “a2time” shows a significantly deviation, i.e. the estimation is too less than for the other benchmarks. “rspeed” and “basefp” are relatively close together concerning the estimation error. Evaluating the characteristics of the benchmarks shows that the actual code size of “a2time” is more than double compared to the other benchmarks. (Note: This is not necessarily derivable from the number of energy estimation elements, because of loops and iterations.) A larger code size results in more energy consumption caused by the instruction cache, which needs to load more instructions. Another differing characteristic is a significant other ratio between local and global variables. Compared to the other benchmarks “a2time” uses 60 % more global variables, which consume more energy and resulting in more energy consumption than estimated.

V. CONCLUSION AND FUTURE WORK

This paper presented and evaluated a new approach of energy estimation for single software components based on program flowcharts without the presence of source code. This enables an energy consumption estimation within early design phases. The aim is to enable system designers the possibility to compare different design variants of the later system with respect to the energy consumption earlier in the development process. One main incentive is to use just available information of this development phase.

The presented estimation model was applied to the flowcharts of three commercially available software benchmarks to show the generally feasibility of this approach. The comparison between estimated and measured energy consumption results in an error bandwidth between -11.9 % and +6.9 %. Using flowcharts enables the use of estimation input which is abstracted from the source code. Flowcharts are able to represent the software structure (program flow) and the individual flowchart elements encapsulate the execution of different programming elements – both abstracted from source code.

Future work will include the evaluation of further benchmarks and also different hardware platforms. Furthermore the determination of the correction factor $\tau_{corr}$ and the probability factor $\omega$ need further evaluation. This includes the study of other influences on the energy consumption such as interaction between software components.

REFERENCES


