Measuring and Modeling Energy Consumption of Embedded Systems for Optimizing Compilers

Mikko Roth, and Arno Luppold, and Heiko Falk
Institute of Embedded Systems
Hamburg University of Technology
Germany
mikko.roth|arno.luppold|heiko.falk@tuhh.de

ABSTRACT

Estimating energy consumption already during development as precisely as possible is crucial for many embedded system designs. These energy estimates should be expressed such that they can be used by subsequent automated optimizations during the compilation phase in order to minimize the expected energy consumption. In this paper we present our current approach on measuring and modeling, and subsequently using the derived energy estimates. Our model is implemented within an optimizing compiler, allowing for future energy focused compiler optimizations.

CCS Concepts

•**Hardware** → **Power estimation and optimization;** •**General and reference** → *Measurement;* •**Software and its engineering** → *Compilers;*

Keywords

Compiler, Energy, Analysis, Measuring

1. INTRODUCTION

Many embedded systems only operate with a restricted energy budget. This naturally includes any battery driven devices, but also ECUs in larger systems like cars, where the combined power consumption of dozens of computing units leads to massive issues regarding power distribution.

Energy consumption can be determined very precisely at a device's gate level. However, the analysis techniques are way too fine grained and complex to be used to express the power consumption of the overall device. Additionally, detailed knowledge on the internal structure of a processing unit must be known in order to analyze the device's power consumption on such a low level. Unfortunately, hardware producing companies regard this information as strictly confidential intellectual property, and subsequently deny any access to this data. On the other hand, power consumption given in the final processing unit's data sheet are usually just highly over-pessimistic upper bounds which are completely useless when it comes to tight estimates.

As a result, the power consumption of any processing unit must be measured manually. Depending on the target system, power consumption usually consists of several components like base costs depending on the executed instruction, costs of memory accesses, etc. These values form the so-called energy model. Due to the fact that hardware architectures differ massively, the relevant key properties of the energy model may also differ a lot.

In this paper we aim at tackling several of these issues. The key contributions of this paper are:

- We propose a setup to measure the energy consumption of common microcontroller devices used in the embedded systems domain.
- We propose an approach to integrate arbitrary energy models and measured data into an analysis framework.
- We integrated the approach into the WCET-Aware C Compiler Framework (WCC)
- We outline, how this framework can also be used for energy focused compiler optimizations.

This paper is organized as follows: Section 2 will first give a brief introduction into related work. Section 3 explains our methodology to precisely measure energy consumption of an embedded system. Section 4 briefly introduces the compiler framework which is going to be used as a basis for future energy oriented optimizations. Section 5 describes how the measured data is represented in the compiler framework as basis for analyses and future optimizations. Section 6 closes with a short description of our future plans.

2. RELATED WORK

For over 20 years, there has been considerable effort to estimate energy consumption on the instruction-level. An early approach is by profiling a program in terms of so called base- and inter-instruction costs [7]. The usual method of measuring is to observe the voltage drop over a shunt resistor using an off the shelf ammeter, when executing loops of repeated instruction patterns to derive the base and inter-instructions costs. While the described measurement and modeling technique is the basis for many research efforts in this area, including our work, the model and earlier measurement setup lacks precision as it relies on using an off-the-shelf ammeter.

SCOPES '18, May 28–30, 2018, Sankt Goar, Germany

DOI: https://doi.org/10.1145/3207719.3207729

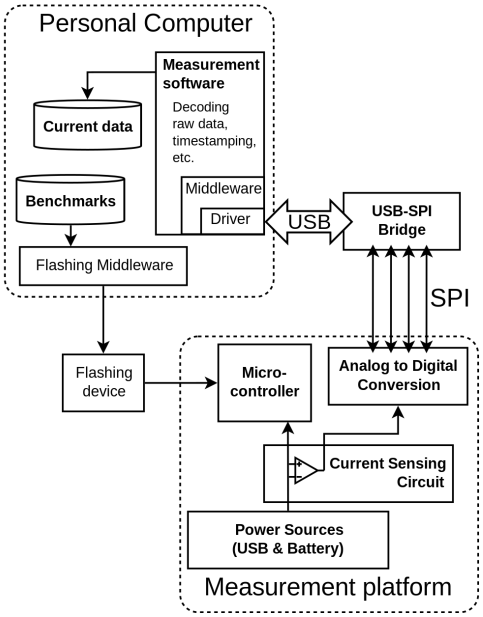

Figure 1: Measurement setup.

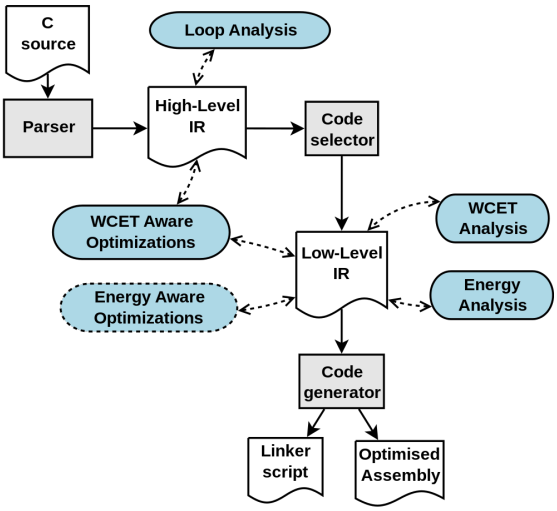

Figure 2: Overall structure of WCC.

Another research group has also gathered power-data with the same low-precision ammeter approach, but the model accuracy is increased by extending base- and inter-instruction costs with consideration of their Hamming weights and distances [6]. The Hamming weight corresponds to the number of '1' in a bit sequence, and Hamming distance is equal to the number of bit-flips to transform a bit sequence to another one. Their model includes effects of data transfers between the processor and bit-toggling on data buses, so they could examine the effects of different allocation schemes of memory objects when multiple memories are available. For this they included another shunt-measurement point at the power supply pin of the external memory. Additionally to the low-precision measurement setup, the energy model seems to have some degree of overlap in their extension of Hamming weights and distances with the base cost definitions.

Various attempts have been made to replace the low-precision ammeter approach by building custom hardware for power measurement purposes, that base on the same idea of measuring the voltage drop over a shunt resistor. We have experimented with a few of them, such as the platform designed by the MAGEEC project [1], as well as a couple of commercially available development boards. The MAGEEC power measurement platform was specifically designed to measure microcontroller power consumption, but unfortunately we ran into technical issues and were unable to fix them within our lab. The commercial solutions targeted at measuring system level or specific feature related consumption, e.g. wireless communication, for application developers, rather then to specifically profile microcontrollers on the instruction level.

The measured data then needs to be imported into a developer's toolchain for utilization. In a recent study [3], a technique to make a program's energy consumption visible at different abstraction levels in the LLVM compiler toolchain was presented. The machine-specific instructions are mapped to the LLVM intermediate representation level as a 1 to m mapping, which is then used to hoist energy information from an instruction set architecture model to the intermediate representation level. They also present an energy profiling technique based on that knowledge at the intermediate level they gained using their mapping technique.

There are a variety of ways estimating power consumption as well as models that go along with them. To have reliable values from power measurements as the basis for energy models, we designed an integrated power measurement platform for modern low-power microcontrollers. As we should be able to handle a variety of different models, we need a method of communicating values for energy models used within compilers.

3. MEASUREMENT SETUP

To derive good quality models for the energy consumption of a processor, one has to have knowledge about a device's power consumption. Companies treat detailed information about the processors' power and energy consumption as confidential. In the best case, there is a global average based on running a full benchmark suite like CoreMark given in the data sheets. To get detailed information to base an instruction-level model on, more fine grained information is needed.

Our goal was to create a design for a power and energy measurement platform with high precision and low noise, while at the same time keeping the design and production low-cost and relatively simple as a blueprint for new measurement targets. The platform should be easy to configure for specific measurement purposes by the end user: Keeping the platform quite small, allows easy modification or switching of the measuring circuit or target device.

Additionally, it reduces the risk of possible unwanted noise sources and leakage currents. To reduce the risk of error noise influencing measurements, the analog signal should not be transferred over long distances and be converted into digital values as soon as possible. For this reason, we decided to integrate a dedicated analog-to-digital converter (ADC) onto the board design itself. The board design consists in

Listing 1: XML definition of the energy consumption for an imaginary platform.

```
<instructions>
  <case>
    <id>OPCODE</id>
    <opcode>sub32</opcode>
    <energy>1</energy>
    <case>
      <id>PREDECESSOR</id>
      <opcode>sub32</opcode>
      <case>
       <id>FALLBACK</id>
       <energy>0.5</energy>
      </case>
    </case>
    <case>
      <id>FALLBACK</id>
      <energy>1</energy>
    </case>
  </case>
<!-- ... Further cases ... -->

  <case>
    <id>FALLBACK</id>
    <energy>2</energy>
  </case>
</instructions>
```

essence of the target device, the power supply regulation circuitry and a measurement point including the ADC.

The ADC outputs the recorded values over SPI to a USB-SPI bridge device, that repackages the raw measurement data for transfer to the PC. Using a commercial SPI-to-USB bridge accompanied by driver and support libraries that handle the low-level USB communication, we reduce development efforts, while keeping high flexibility as we can control the data acquisition by simply modifying application code on the PC. This way , we are also able to use user-friendly libraries and up-to-date driver to handle the low-level USB communication.

The measurement software on the PC is a console driven application written in C, and to keep it concise and maintainable, we make use of the aforementioned libraries and drivers. Of course, we also need to be able to reprogram the target device with benchmark binaries. We use a widely used commercial debug and flashing device to reprogram the target device . For running a long series of benchmarks over an extended period of time and to be able to easily repeat them, the user can write scripts to orchestrate complex measurement runs of a long series of benchmarks.

We have produced a prototype of the measurement board, on which the current is measured by a precision current-sense amplifier that amplifies the voltage drop across the shunt resistor. The output signal of the current-sense amplifier is sampled by an 12 Bit analog-to-digital converter. The ADC is controlled by the PC driver via an USB-to-SPI interface device controller bridge. Due to technical limitations, the USB-to-SPI interface device controller is only able to operate the ADC with 3 MHz clock which sets a maximum sampling rate of 187.5 ksps. Sampling is controlled by a measurement driver on the PC, which requests samples in 65.5 kbyte chunks in bit banging mode. This causes a small break after every 32 767 samples, but on average we achieve a sampling rate of 170 ksps, based on our initial observations.

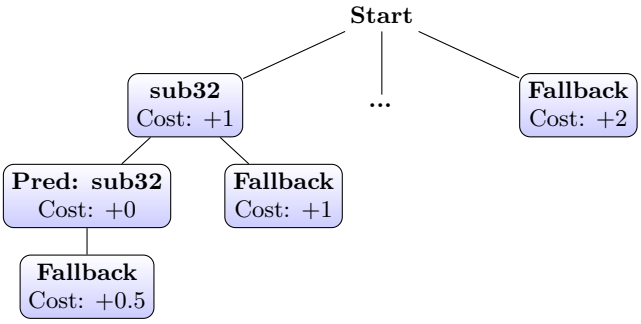

Figure 3: Exemplary structure of the energy analysis model for an imaginary platform.

Drawbacks of this setup are the initial investment in the design and production effort of the measurement platform. Another drawback we want to tackle in future, is the intrusive nature of the currently used measurement circuit based on placing a shunt resistor into the power supply rail.

With this measurement setup, we are able to instantiate the data gathered as an energy model into the WCET-Aware C Compiler WCC [2] developed in our group. This way, our compiler will be able to not only analyze and optimize a program's worst-case timing behavior, but also return a good estimate on its energy consumption.

4. WCC FRAMEWORK

The WCET-aware C Compiler (WCC) is an optimizing C compiler for embedded systems [2]. WCC allows to integrate and to apply optimizations for worst-case execution time (WCET) minimization into the compilation process. Currently, it supports the ARM7TDMI architecture, Infineon TriCore TC1796 and TC1797 and we have basic support for the Infineon Aurix microcontroller family. Additionally, we are currently extending the compiler framework to support the Leon3 microprocessor.

The overall structure of the WCC is shown in Figure 2: The first stage is the parser, that accepts C source files and generates a machine independent high-level Intermediate representation (High-level IR). It includes source code analyses and optimizations, for example on loop structures, and generates an optimized version of the code to be passed on further down to the next module.

The optimized high-level IR from the previous step is fed into the code-selector that generates a low-level intermediate representation (Low-level IR). Various standard assembly-level analyses and optimizations are performed at this stage. For timing optimizations, a WCET analysis tool is integrated at this stage to attach timing information to the Low-level IR of the program and to extract the worst-case execution path. The timing information is exploited by various optimizations to reduce the WCET of the input program, and specialized optimizations can be applied only to these code portions to minimize WCET aggressively.

After the various analyses and optimizations have been

concluded, the optimized code is passed down to the code generator to emits assembly code and appropriate linker scripts to allow producing a valid executable.

Our aim is to add analyses and optimizations that target the energy consumption of embedded software. The low-level IR is designed to be flexible for attaching new information to be used for specialized optimizations. We have already implemented the energy analysis module that attaches energy consumption information at the low-level IR according to the model and input specification as described in the next section.

5. COMPILER MODEL

Our goal is to instantiate the data gathered by the energy measurements as an energy model into the WCET-Aware C Compiler WCC [2] developed in our group. This way, our compiler will be able to not only analyze and optimize a program's worst-case timing behavior, but also return a good estimate on its energy consumption.

One of the key challenges with this is, that the compiler features different target architectures. Despite the fact that all target platforms are RISC architectures, they differ substantially in their internal structure. At the time being, we do not yet know for sure each and every core component which has to be accounted for when it comes to tight energy estimates on these concrete targets.

As a result, we aimed at designing an analysis framework which is arbitrarily customizable and extensible to allow a flexible way model different target platforms as easy as possible. Consequently, we decided to describe the energy behavior of the system as structured XML data, which is then parsed into our compiler as a tree structure. The energy analyzing framework within the compiler will then use a depth-first algorithm to match every instruction with an energy behavior.

A very simple example XML specification for an imaginary system is shown in listing 1. The resulting analysis tree after being parsed by WCC is shown in Fig. 3. An instruction is matched from left to right, summing up all costs until the deepest match. The physical unit does not play any role here but will, most likely, be something like nJ. The flexibility of the model comes from the possibility to include logical expressions in the tree. In the given example, the cost for a *sub32* will be $1+0+0.5 = 1.5$ if the previously executed instruction has also been a *sub32*. Otherwise, the energy consumption of that instruction will be $1 + 1 = 2$. The top-level *Fallback* node finally simply assigns all other instructions a cost of 2.

Due to a very flexible object-oriented approach, it is very simple to add new types of logical blocks as needed. This way, inter-instruction costs (cf. *sub32-sub32* in Fig. 3) can easily be modeled. So far, we also support modeling accesses to different memories and costs depending on the core an instruction is executed on in case of a multi-core setup. We also have a block to match several given instructions at once. Analysis blocks to model bit-level Hamming distances between different instructions are currently developed.

One big benefit of this approach is that, due to the XML syntax, the user may simply plug the base blocks and annotate them with costs as needed, without even changing anything at the compiler's code base. Another benefit is the fact, that the tree-like structure does not enforce the user to annotate each and every energy consumption behavior in the same detailed level. E.g., many logic instructions might be "boring" from an energy analysis point of view, requiring only very coarse-grained analysis, while a few instructions might require detailed and time-consuming analysis (e.g., value analysis, calculating Hamming distances, etc.). Using our model, the user can decide at run-time, which instructions should be analyzed in which detail, allowing a trade-off between a small energy consumption description in XML, analysis run-time and analysis quality.

6. SUMMARY AND FUTURE WORK

We have produced a functioning prototype of our proposed measurement setup, and work on addressing the identified drawbacks and plan to produce a revised version of the measurement platform. For example, placing a shunt resistor directly into the power supply path is a intrusive method. We plan investigating into other, less intrusive current sensing circuits. We are already looking at building a version based on a current mirror setup that should offer higher measurement resolutions [4]. In our blueprint design, this change requires only replacing the current sensing circuitry and adjusting some configuration parameters within our measurement software for decoding the raw data.

In the long run we, we aim at extending the compiler framework towards multi-criteria optimizations to simultaneously optimize both WCET and energy consumption [5].

Acknowledgments

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779882.

This work received funding from Deutsche Forschungsgemeinschaft (DFG) under grant FA 1017/3-1.

7. REFERENCES

[1] MAGEEC. http://www.mageec.org/. Referenced 2018-04-12.

[2] H. Falk and P. Lokuciejewski. A Compiler Framework for the Reduction of Worst-Case Execution Times. *Real-Time Systems*, 46(2):251–298, 2010.

[3] K. Georgiou, S. Kerrison, Z. Chamski, and K. Eder. Energy transparency for deeply embedded programs. *ACM Trans. Archit. Code Optim.*, 14(1):8:1–8:26, Mar. 2017.

[4] T. Laopoulos, , P. Neofotistos, C. A. Kosmatopoulos, and S. Nikolaidis. Measurement of current variations for the estimation of software-related power consumption. *Instrumentation and Measurement, IEEE Transactions on*, 52(4):1206–1212, 2003.

[5] K. Muts, A. Luppold, and H. Falk. Multi-criteria compiler-based optimizations of hard real-time systems. In *In Proc. of SCOPES*. ACM, May 2018.

[6] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel. An accurate and fine grain instruction-level energy model supporting software optimizations. In *In Proc. of PATMOS*, 2001.

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