Chapter 3

Measuring Energy

Steve Kerrison, Markus Buschhoff,

Jose Nunez-Yanez and Kerstin Eder

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/65989

Abstract

This chapter provides an introduction to quantifying the energy consumed by software. It is written for computer scientists, software engineers, embedded system developers and programmers who want to understand how to measure the energy consumed by the code they write in order to optimize for energy efficiency. We start with an overview of the electrical foundations of energy measurement and show how these are applied by reviewing the most commonly found energy sensing techniques. This is followed by a brief discussion of the signal processing required to obtain energy consumption data from sensing. We then present two energy measurement systems that are based on sensing techniques. Both can be used to directly measure the energy consumed by software running on embedded systems without the need to modify the hardware. As an alternative, regression-based techniques can be used to infer energy consumption based on monitoring events during program execution using counters monitors offered by the hardware. We introduce the foundations of regression analysis and illustrate how an energy model for an ARM processor can be built using linear regression. In the conclusion, we offer a wider discussion on what should be considered when selecting an energy measurement technique.

Keywords: energy measurement, power, energy sensing, energy measurement systems, regression analysis

1. Introduction

Energy is now the limiting factor in electronic system design. To develop more energy efficient systems, energy must be taken into account during system design at all levels of abstraction. Low-power design has been a focus for hardware developers for several decades with

impressive results in terms of low-power processors from embedded to high-performance systems.

However, beyond the hardware layer in the system stack there are further savings to be made. Experts expect these to be significantly higher than what the hardware can achieve on its own. For example, while dedicated low-power hardware can realize savings of 20% in multimedia processing, three to five times more could be saved with software support [1]. While energy efficient software development is only just emerging, it is clearly an essential part towards achieving energy efficiency of whole systems.

To support energy efficient software engineering, the energy consumed by software needs to be determined. Energy measurement is one way to achieve this, and this chapter provides the reader with an insight into techniques that can be used to measure the energy consumed by software running on embedded systems. The measurements can then be used to gain deeper understanding of how algorithms, languages, coding styles, data structures and compilers impact on the energy consumed during program execution; they also help engineers identify energy bugs in the software. Furthermore, energy consumption monitoring at runtime becomes feasible, and this enables dynamic adaptations to be introduced into systems to adjust their energy consumption in response to high or low levels of activity, or to accommodate varying levels of demand. Energy measurements also allow the creation of energy models. These can be used to estimate energy consumption either at design time or at runtime, without the need for measurements to be taken, thus saving the effort involved in setting up and measuring, often at the expense of accuracy, although acceptable error margins can be achieved with state-of-the-art modeling techniques.

Broadly speaking, energy consumption of electronic systems can be measured either directly or indirectly. The direct approach relies on sensing, which may require some instrumentation of the target hardware, often involving invasive procedures such as soldering, to install the components required for measurements to be taken. The knowledge and skills needed to accomplish this step are typically not within the repertoire of software developers, which can be a considerable barrier in practice. External energy measurement systems already come with sensing components built in and only require connection to the measurement points on the target hardware. This chapter introduces the fundamental concepts of direct energy measurement in Section 2 and presents two external measurement systems in Section 3.

The indirect approach infers energy consumption from other events that can be measured during program execution [2]. Modern architectures offer counters integrated into the hardware to collect statistics on the operation of the processor and memory system. Examples include the counters in the performance monitoring units on the Intel Xeon Phi [3] or the ARM Cortex A9 [4]. A variety of different events can be counted at runtime, such as the number of read or write misses at level 1 in the data cache or the number of data cache-dependent stall cycles in a pipeline. Regression analysis is applied to establish a correlation between these events and dynamic power dissipation. If this is successful, a predictive model can be built. Section 4 provides an introduction to regression analysis, which is illustrated with a worked example for the ARM Cortex A9 in Section 5.

http://dx.doi.org/10.5772/65989

Beyond direct measurement and inferring energy consumption indirectly through regression analysis, energy consumption information can also be obtained based on simulating a hardware design. This requires availability of the gate-level design description and layout information. A power estimation tool can then be used to obtain power data based on the amount of switching recorded during hardware simulation. This, together with performance data, provides energy consumption information. However, we will not cover this approach in more detail in this chapter, as we focus on energy measurement of real systems rather than designs.

2. Basics of direct measurement techniques

This section provides the necessary background to understand the processes by which a device's energy consumption can be directly sensed. The underlying physical principles defining electrical energy consumption are explained in detail, followed by a discussion of methods for observing and recording measurements.

2.1. Fundamental concepts

In the following sections we will briefly introduce some fundamental concepts and terminology of electrical engineering in the domain of energy measurement.

In general, measuring energy directly is infeasible, because energy is a virtual concept that quantifies the influence that things can have on their physical environment. Instead of measuring energy directly, we have to measure these physical effects and to deduce the energy that was involved in realizing these effects. The most important effect of energy in the realm of electricity is the ability to transport electrical charge, to build up electric and magnetic fields and to produce heat, light or other forms of radiation.

Electrical energy can be defined as the conversion of an electrical power, P, measured in watts, W, for a given amount of time. Since power changes continuously over a period of time $[t_0, t]$, energy is the integral of all converted power during this interval:

$$E(t) = \int_{t_0}^{t} P(t)\mathrm{d}t \tag{1}$$

Consequently, energy can be expressed in units of watt-seconds, Ws, which is equivalent to the unit joule, J, where 1 J = 1 Ws.

Electrical power is defined as the strength of an electric current, I, measured in amperes, A, caused by an electric "driving pressure", the potential difference, measured in volts, V. For a given point in time, the electrical power is defined as:

$$P(t) = V(t) \cdot I(t) \tag{2}$$

For certain classes of observed systems, assumptions can be made about voltage and current. For example, for many computational systems, the supply voltage is constant over a great

period of time. This reduces the problem of measuring energy to the measurement of current and time. Thus, a device's energy consumption can be calculated as:

$$E(t) = V_{const} \int_{t_0}^{t} I(t) \mathrm{d}t \tag{3}$$

More generally, electrical energy can be measured by measuring voltage and current over a known amount of time. In practice, several decisions need to be made when developing an energy measurement approach. Firstly, the precise measurement of either current or voltage (they can be converted, as shown later), second, the integration of measured values and finally the reduction of the energy required for the actual sensing. The latter is necessary to reduce the influence that measurements have on the observed system. In the following, the observed device is referred to as DUT (device under test).

We now address each of these options in detail. In Section 2.2, we describe approaches to sense the influence that current, voltage and energy have on the environment. In Section 2.3, we discuss the means to amplify the sensed values, so that we can increase sensitivity for the purpose of reducing the measurement equipment's influence on the DUT. We will further talk about analog to digital conversion and the problems that arise from the discretization of time imposed by this conversion.

2.2. Sensing

A sensing unit is the basic element in a measurement setup. There are a variety of approaches to implement the sensing of energy consumption. In the following, the most prominent sensing approaches will be discussed.

2.2.1. Voltage drop or shunt measurement

The term "shunt" refers to an electrical resistor that is used to convert electrical current into a voltage drop for the purpose of measurement. In practice, voltage drop sensing is very common and easy to implement with analog to digital converters or even standard measurement-equipment such as a multimeter or oscilloscope.

According to Ohm's law, $V = R \cdot I$, a current I through a resistor R is caused by a voltage V proportional to R and I. Having a shunt resistor in series with a DUT results in an equal current flow through both components, while the supply voltage is split among them. This circuit of a *voltage divider* is schematically shown in **Figure 1**.

To avoid that the current being restricted mainly by the shunt and to avoid as a consequence the majority of the voltage drop occurring at the shunt resistor—both having negative impact on the DUT—the shunt resistance value must be chosen to be smaller by several magnitudes compared to the asserted resistance of the DUT.

As an example, we take a DUT that requires a constant voltage of about 4 V and a maximum current of 100 mA. This device has a theoretical maximum resistance of 40 Ω according to Ohm's law. With a badly selected shunt resistor of 4 Ω in series, the maximum current

http://dx.doi.org/10.5772/65989

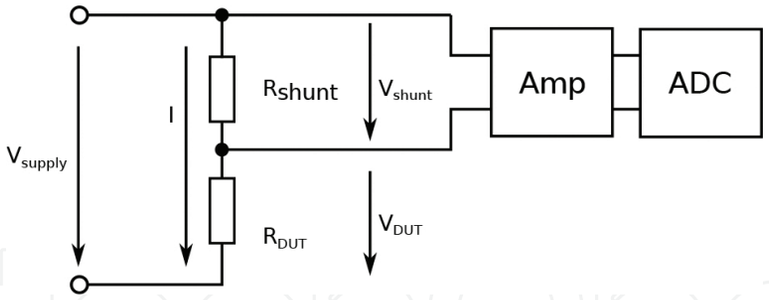


Figure 1. The concept of current measurement using a shunt in a voltage divider.

would be reduced to 91 mA. So, when the DUT is at maximum load, the voltage on the shunt resistor is 390 mV, whereas the supply voltage at the device drops to 3.7 V. This would not only distort the measurement results to a large degree but also could make the DUT malfunction.

On the other hand, using a shunt of 10 mΩ would just reduce the supply voltage to 3.999 V and the current to 99.98 mA. But, the maximum voltage to measure at the shunt would be below 1 mV.

In conclusion, the following assertions on shunts can be derived:

- The resistance value of the shunt is limited by the maximum acceptable voltage drop on the DUT's supply voltage and its maximum current consumption. This limits the measurement range.
- Usually, extremely small resistance values with low tolerances are preferable, which makes measurement shunts expensive.
- The small voltage drop at a shunt is prone to thermal noise.
- The small voltage drop at a shunt needs to be amplified to process the measurement signal.

To overcome these issues, there is a broad variety of high-quality (and high-cost) shunt resistors on the market to suit all measurement purposes. Additionally, there are technical approaches to enable precise or wide-range shunt measurements. To increase the measurement scale, some circuits use a shunt switching technology: whenever the voltage drop exceeds a threshold value, a lower Ohmic shunt is selected, thus enhancing the measurement range. This is not simple to implement because the transition from one shunt to the next must be exactly calibrated. Also, the threshold values for switching forward and backward must have a certain distance from each other (hysteresis) to prevent the system from oscillating between two shunts. Additionally, the current shunt selection must be communicated to successor circuits that process or use the measurements, since the evaluation of the measured voltage drop is dependent on the concrete value of the shunt.

Another approach often discussed is using exponential or logarithmic elements like diodes, instead of linear shunts. Usually these elements have technical issues such as strong temperature dependabilities and impractical tolerances that are hard to overcome.

2.2.2. Charge accumulation

Another method to measure energy is to use it to transfer an electrical charge into a capacitor. The charging capacitor will raise its voltage according to the following equation, with C being the capacity, measured in Farad, F.

$$E = \frac{1}{2} \cdot C \cdot V^2 \tag{4}$$

The energy stored in the capacitor can be derived from its voltage level. A good way to transport energy into a capacitor in proportion to the energy consumed by a DUT is the utilization of a *current mirror*. Current mirrors let a current pass through their input contacts while using an external power source to reproduce the same (or a proportional) current flow at their output side.

Assuming a constant voltage supply for the DUT, the *charge*, measured in Coulombs, C, which is equivalent to ampere seconds, As, transported to a connected capacitor is

$$Q(t) = Q(t_0) + \int_{t_0}^{t} I(t) \mathrm{d}t. \tag{5}$$

There are two suitable approaches to measure energy continuously: one is to sample and discharge the capacitor (or an alternating set of capacitors) with a constant frequency and the other approach is to wait until the capacitor voltage reaches a given limit to trigger a counter. **Figure 2** shows the corresponding circuit. In this approach, the frequency of the counter signal increases proportional to the energy consumption.

2.2.3. Charge transfer

Similar to accumulating charge, methods based on the transfer of charge into a capacitor can take the time into account that is required to transport energy into it. Capacitors are charged through a resistor. The time to increase the voltage of a capacitor is denoted by τ.

$$\tau = R \cdot C \tag{6}$$

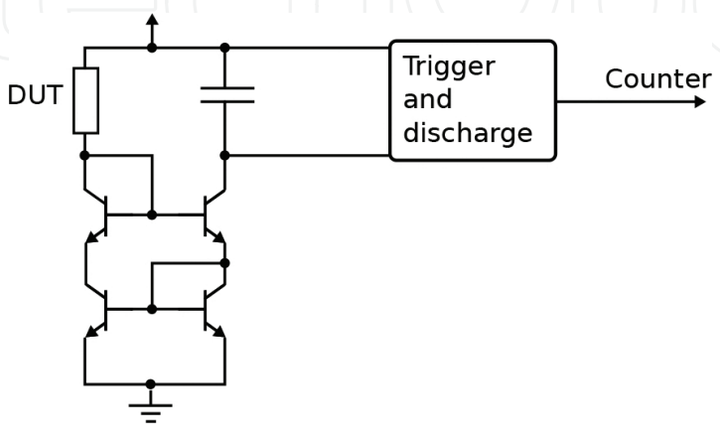


Figure 2. A Wilson current mirror used to charge a capacitor.

http://dx.doi.org/10.5772/65989

$$V(\tau) = V(t_0) + \left(V_{sup} - V(t_0)\right) \cdot \left(1 - e^{-1}\right) \approx V(t_0) + 0.632 \cdot \left(V_{sup} - V(t_0)\right) \quad (7)$$

Konstantakos et al. [5] have evaluated the usability of this measurement method for the in-situ power measurement of embedded systems. They implemented different circuits based on charge transfers using current mirrors and concluded that this method is accurate enough to serve as an in-situ measurement approach for embedded systems. Additionally, it can be implemented without dependencies to the clock frequency of the system.

One of the main advantages of using a capacitor to collect energy is the implicit integration of current over time. As capacitors are analog elements, there is no sampling involved, so that even very short energy peaks will be taken into account. A disadvantage is the additional analog circuitry required, which adds thermal noise and nonlinearities and thus is prone to reduce accuracy.

2.2.4. Magnetic field

A current flowing through a conductor creates a magnetic field around the conductor. This field can be sensed by devices like *Hall-effect sensors*. The Hall effect describes the occurrence of a voltage (Hall voltage) within a live conductor that is positioned perpendicular to an external magnetic field. That means, placing the live conductor to a DUT close-by and perpendicular to another live conductor (sensor) will induce a measurable Hall voltage into both conductors. Since both magnetic fields, that of the main conductor and that of the sensor, influence each other in the same way, the current through the sensor has to be smaller than the current through the examined conductor by several orders of magnitude.

The Hall effect has only a small impact on the voltage of the sensor, so a very sensitive amplifier has to be used to amplify its output signal. Hall-effect measurements are prone to errors, because many parameters influence the measurement. The conductor material, as well as its distance from the sensor and any insulating material between the two, all have a significant impact. Also, if there is air between the conductor and the sensor, air humidity and temperature might influence the measurement results.

The main advantage of Hall sensor measurements is the contact-free and nonintrusive way that a Hall sensor can be deployed. For these reasons, Hall sensors are mainly used to measure high currents in environments where invasive measurement is not desirable.

2.3. Signal processing

As shown in **Figure 1**, a typical measurement setup has to process the signal of the sensing unit to convert it into a human or machine readable form. This usually includes at least two stages: 1. A signal amplifier to adapt the output level of the sensing unit to satisfy the requirements of successor units. 2. Most often, the amplifier will be followed by an analog to digital conversion unit (ADC) to make the signal machine readable.

There is a broad range of measurement amplification circuits, and the quality and complexity of these circuits have an impact on the accuracy of the measurement and the effective measurement range. While simple units with low requirements may contain only a single bipolar

transistor and few passive elements, more sophisticated setups that use one or more high-quality OpAmps allow for better temperature stability, better signal-to-noise ratio, amplification of both positive and negative signals and fewer parasitic effects like unwanted filtering of peaks, thus allowing measurements of signals of higher frequencies.

Among ADCs, the diversity is no less. Every ADC will convert a signal level (voltage) to a digital value by comparing the signal voltage to a reference voltage and calculating the ratio as digital value. Two fundamental parameters are the resolution, which is the number of bits per converted signal, and the sample frequency. Further parameters define the accuracy of the conversion, usually described by a set of error measures like offset error, gain error and nonlinearity in an ADC's data sheet.

An important design consideration is imposed by the Nyquist-Shannon sampling theorem, which limits the maximum signal frequency that can be sampled to half of the sampling frequency. Higher signal frequencies may cause errors in the conversion results. This adds the requirement to insert a low-pass filter into the signal path, which cancels out frequencies above the frequency limit. These filters are often integrated in ADC circuits and thus not considered any further in the design of measurement equipment. Due to the undefined quality of these filters, the input frequency range allowed in the product descriptions for commercial measurement equipment is often very limited.

Better filter systems integrate the signal accurately before sampling. Although low-pass filters and analog integrators are equivalent from a conceptual perspective, sophisticated integration units are more accurate and can limit the integration interval exactly to the sampling time. In the case of energy measurements, this would guarantee that the measurement result is always correct, even if the signal contains peaks of a width that is only a fraction the sampling interval time. Although such a peak would not be visible as such in the sample data, it would correctly increase the ADC's next output value. This concept was used in the MIMOSA measurement tool presented in Section 3.

3. Example direct measurement systems

This section presents two measurement systems designed to allow the energy consumption of embedded systems to be captured. The two systems, *MAGEEC wand* and *MIMOSA*, have slightly different design goals and use different measurement techniques. Both measurement systems offer fully working solutions; describing their construction here is expected to aid in the development of future measurement devices.

3.1. MAGEEC wand

The MAchine Guided Energy Efficient Compilation (MAGEEC) project[1] sought to find compiler optimizations that reduce energy. As part of this work, real hardware measurements were used, rather than model-based estimations. To that end, the MAGEEC wand, shown

[1] http://mageec.org/

http://dx.doi.org/10.5772/65989

in **Figure 3**, was created. Complementing the open source nature of the compiler work that was performed in the project, the wand's custom hardware and software is also open source. A number of MAGEEC wand kits were produced and both sold and given away at workshops. Using the published designs, anybody can commission the manufacture of their own boards.

3.1.1. Features

The MAGEEC wand is designed to be flexible in how it is used to suit various energy measurement needs. Its key features include:

- Three measurement channels that can be monitored simultaneously.
- Sample rates of up to 2 million samples per second.
- The measurement board attaches to a widely available, low-cost embedded system to capture and transmit collected data.
- It can use one of the available channels for self-monitoring of the capture device's energy consumption.
- Each channel can be easily adjusted to measure a wide range of power supplies and power ranges.
- Supplies with pre-installed shunt-resistors can also be monitored.
- The measurement firmware provides various capture methods, including streaming data, triggered capture and a live GUI.

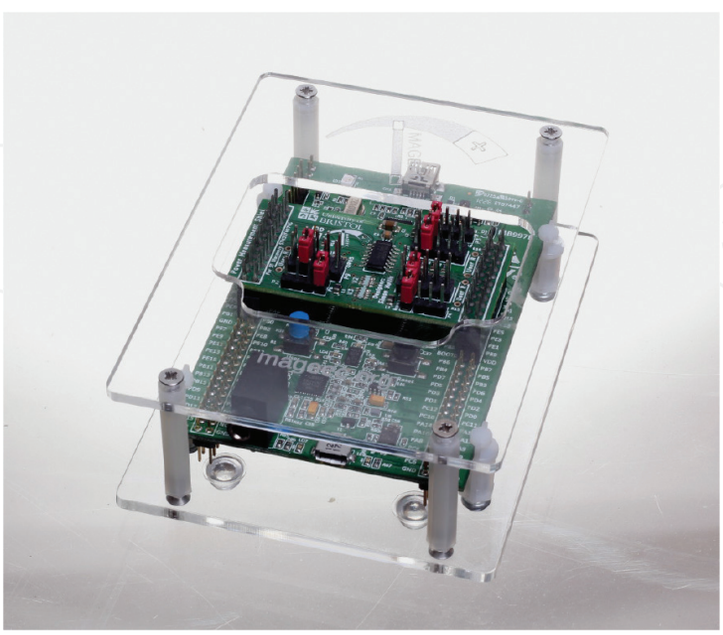


Figure 3. MAGEEC wand attached to an STM32F4 discovery board.

3.1.2. Construction

The MAGEEC wand is a small add-on board designed to connect to an ST *discovery board*, which is a low-cost MCU evaluation board from ST that includes an ARM Cortex-M series processor. A block diagram of the relevant components is shown in **Figure 4**. The wand uses the common shunt-resistor based measurement method, as described in Section 2.2. It features a selection of shunt resistors per channel, probe points for connecting the DUT, inductors and an array of current sense amplifiers in a MAX4378FASD chip. The discovery board acts as the controller and data acquisition device. The on-chip ADCs are used to sample the voltage drop that was amplified on the wand. Sample data are delivered through USB to a connected PC.

Devices that use up to a 12 V power supply can be safely monitored with the wand. The power supply to the DUT may need to be modified to allow sensing, typically by splicing a cable. However, many devices, particularly evaluation boards of embedded systems, feature shunt resistors and probe points, removing the need for any hardware alterations. In the former case, an appropriate resistor value must be chosen, such that the voltage drop is sufficiently large to observe with minimal noise. Additionally, if the splicing is done by removing an inductor on the target board, an inductor on the wand can be used in its place. In the latter case, the inductor and resistors on the wand can be bypassed, although the shunt resistor value on the target device must be noted in order to correctly scale the measurements that are obtained.

The on-board resistors for each channel are 0.05 Ω, 0.5 Ω, 1 Ω and 5 Ω, with a header and space for a custom resistor per channel. Jumpers select the type of resistor to be used. A trigger pin can be assigned on the discovery board to allow sampling to be controlled by an external event, such as the toggling of a GPIO port on the DUT.

The design of the MAGEEC wand means that it does not provide power to a device. However, MIMOSA does, as described in Section 3.2, thus providing an alternative where this is preferred.

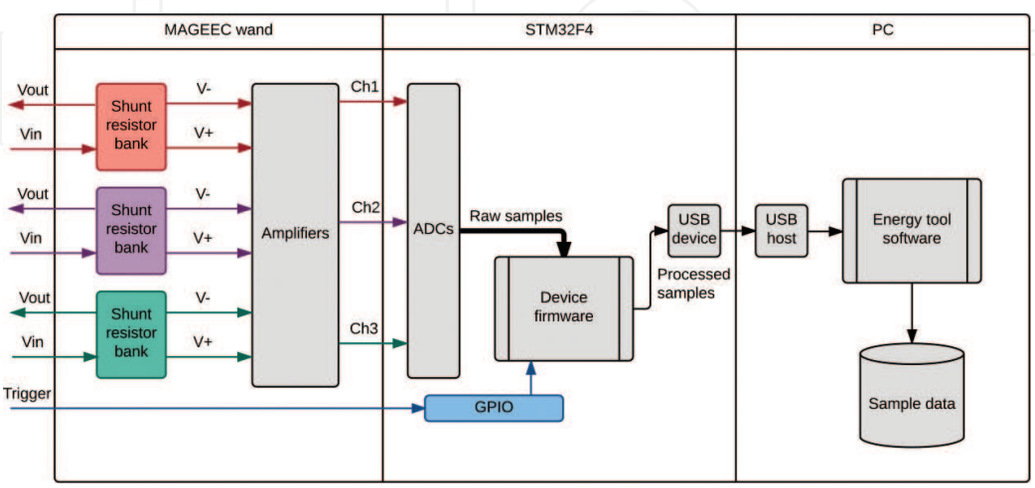


Figure 4. Block level depiction of MAGEEC wand and relevant STM32F4 components for taking energy measurements.

http://dx.doi.org/10.5772/65989

The device firmware and PC-side library allow data to be collected and presented in several ways. The *energy tool* program [6] is written in Python and can be run standalone or used as a library to integrate with other tools. It supports three main modes of acquisition:

1. In triggered mode, the device firmware samples power during the triggering period of the assigned GPIO. At the end of the triggering period, the duration, average and peak power, as well as total energy, are provided to the host PC.
2. In continuous mode, data are continuously sampled. The firmware aggregates samples to provide data to the USB host at a reasonable data rate.
3. In interactive mode, continuously sampled data are provided in a real-time graph. Parameters such as channel selection and resistor values can be interactively configured. This provides easy experimentation and observation prior to programmatically configuring these parameters for automated data collection.

An additional bundled tool, *platformrun*, combines the energy tool with the ability to run programs on a DUT, coupling the compilation and execution of a program of interest with the data collection process. This allows automated collection of program energy consumption under changing parameters, such as the compiler parameters explored by the MAGEEC project.

3.1.3. Uses

The MAGEEC wand has been used in various contexts to date. In 2014, a FOSDEM workshop was held where attendees could set up a wand to measure the energy of a variety of embedded devices.[2]

In research, the wands have been used to collect energy data for processor and communication modelling [7], data-dependent energy modelling [8, 9] and exploration of compiler optimizations for energy efficiency. Finally, the MAGEEC wand was of course pivotal in the research output of the MAGEEC energy efficient compiler optimization research.

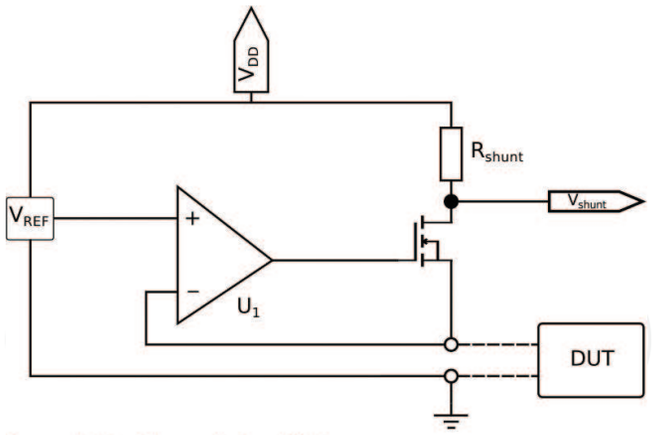
3.2. MIMOSA

Buschhoff et al. [10] proposed MIMOSA[3], a measurement device for creating high-accuracy energy models of embedded system components. MIMOSA combines different measurement approaches. It acts as the power source for the DUT, and by that it measures the energy that it delivers to the DUT. This is achieved by implementing a constant voltage source using a feedback loop on an operational amplifier (OpAmp) to create the output voltage from a high-impedance reference voltage. Such a circuit is sometimes referred to as *voltage follower*.

Compared to a typical voltage follower, which feeds back the OpAmp's output to one of its inputs directly, MIMOSA breaks the feedback loop with a transistor, as shown in **Figure 5**. Since the OpAmp strives to cancel out the voltage difference on its input pins, the transistor

[2]MAGEEC FOSDEM workshop: http://mageec.org/wiki/Workshop

[3]"**M**essgerät zur **i**ntegrativen **M**essung **o**hne **S**pannungs**a**bfall," German for "Measurement device for integrative measurements without voltage drop."



Source: [10], with permission of Springer

Figure 5. MIMOSA voltage regulator circuit.

will be forced to create the reference voltage at the DUT side by the OpAmp, whereas the current supply for the DUT actually has to come over a shunt from MIMOSA's own voltage supply. For that to work, the MIMOSA supply voltage must be greater than that of the DUT. Contrary to normal shunt measurement, the shunt used here can be high ohmic because the voltage drop of the shunt is compensated by the regulatory circuit.

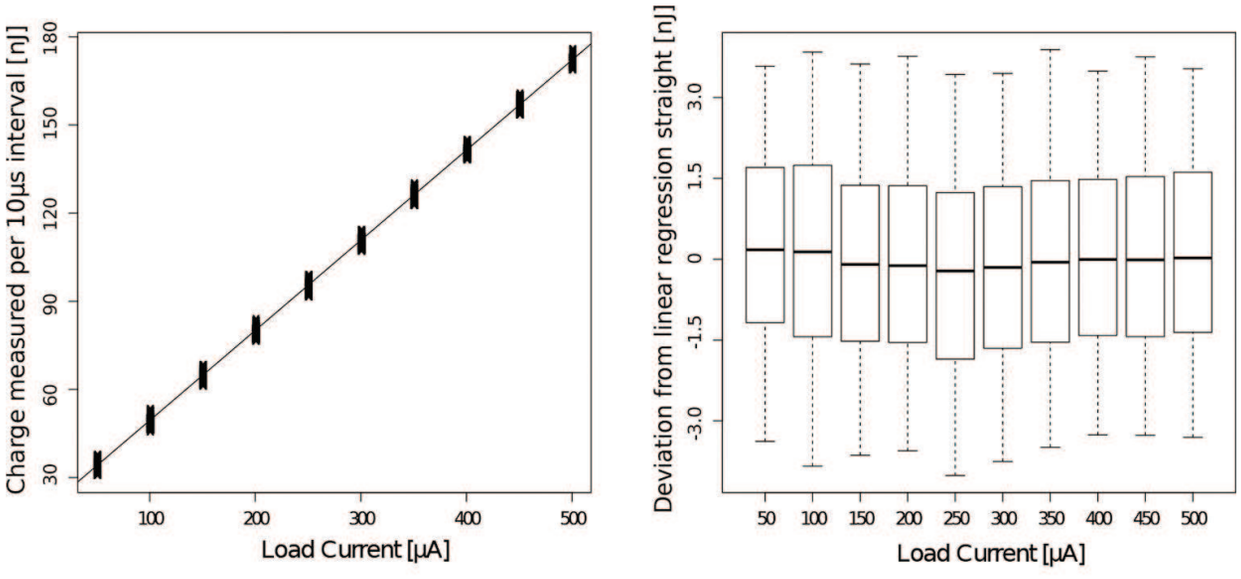
Using a high-ohmic shunt here has some advantages over the usual "shunt plus amplifier" approach. As an example, a current of 1 mA would create a voltage drop of 1 V at a 1 kΩ shunt. That means, no further amplification is necessary, and a standard ADC connected to the shunt can sense currents down to the micro ampere region. Due to the fewer components required, thermal noise is less of an issue, while precise high-ohmic resistors are far less expensive than precise measurement shunts in the milliohm-range. **Figure 6a** shows a reference measurement in a DC situation. Ten high precision ohmic resistors were used as load. For each resistor, 300,000 measurements were taken. The figure shows the measured range as a vertical bar for each resistor. **Figure 6b** shows the distribution of the measurement results (normalized average to the linear regression straight).

In the next stage, MIMOSA integrates the voltage measured at the shunt by using a set of three analog integrators (each consisting of an OpAmp and a capacitor). This can be seen in the overview of the system in **Figure 7**. The integrators function in a rotational manner: while one integrator is connected to the shunt, another one's output value is sampled by the ADC and the third integrator is reset. The sampled value is then sent to a PC using a USB connection.

MIMOSA additionally features a digital input that can be used to tag the collected samples, bookmarking important events. The DUT can use this connection to signal important events like the start and the end of a program sequence to analyze. The state of the digital input is encoded within the data stream.

On the connected PC, data can be evaluated with a graphical user interface. The user can display and record measurements, select and cut interesting sections and export data for further evaluation (see **Figure 8**).

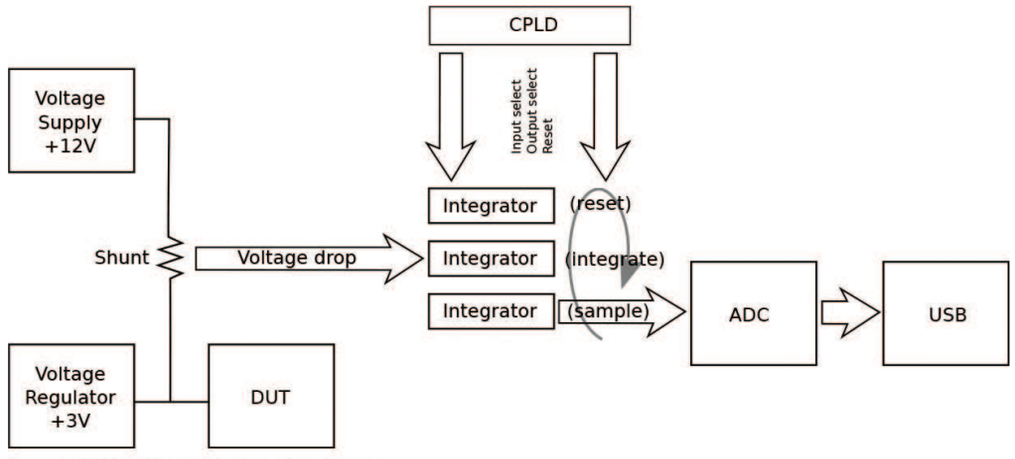
http://dx.doi.org/10.5772/65989



(a) DC linearity (b) Noise

Source: [10], with permission of Springer

Figure 6. DC measurement with 10 precision resistors [10]. (a) DC linearity and (b) noise.



Source: [10], with permission of Springer

Figure 7. Overview of the MIMOSA architecture [10].

Figure 9a shows measurements of rectangular signals that are shorter than the actual sampling period of 10 µs at a sampling frequency of 100 kHz. Throughout the measurement range, MIMOSA shows results close to linear. This is depicted more precisely by the box plot in **Figure 9b**, where the deviation from the regression line and the distribution of measurement values is shown.

In conclusion, MIMOSA aims at the precise measurement of deeply embedded systems, as it allows for the sensing of current in the lower μA region, while still having good time precision through its sample rate of 100 kHz. Due to the integration circuits, MIMOSA guarantees a good energy measurement accuracy; even small peaks far below the sample period will be accounted for. The sample rate can be raised by using higher value sampling

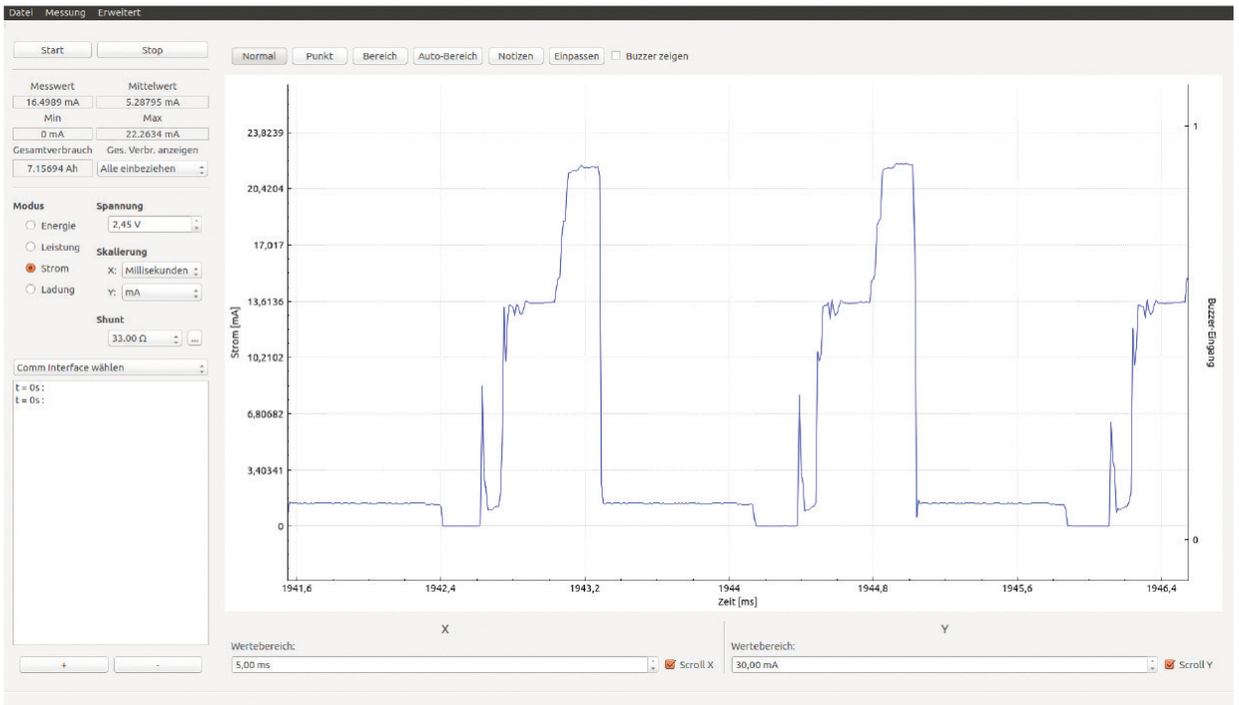


Figure 8. The MIMOSA GUI application.

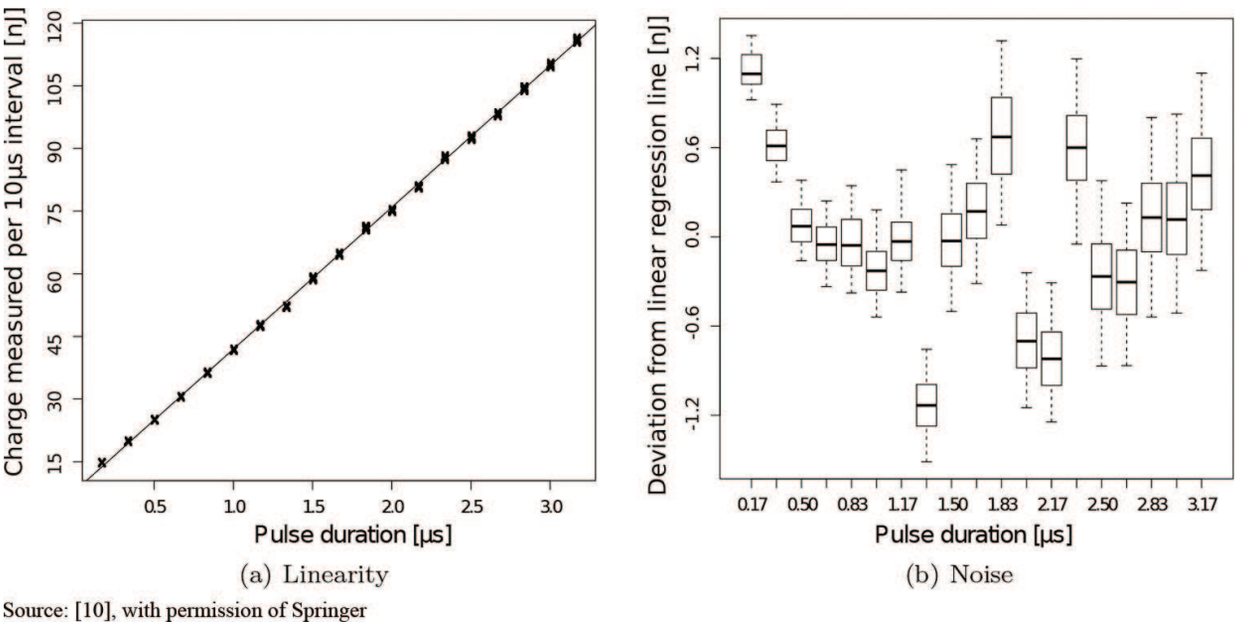


Figure 9. Integration of peaks smaller than the sample period (10 μs) [10]. (a) Linearity and (b) noise.

devices like oscilloscopes right behind the sensing unit. In comparison to existing commercial devices for energy measurement in embedded systems, MIMOSA is able to represent the signal form more precisely without loss of overall accuracy [10]. On the downside, MIMOSA requires a more sophisticated setup as it is necessary to replace the constant voltage source of the DUT.

http://dx.doi.org/10.5772/65989

4. Basics of regression-based techniques

Modifying and instrumenting hardware to measure its energy consumption may be difficult or undesirable in some circumstances. A certain level of skill is required, and it may not be practical to provide the measurement apparatus to all instances of a device. An alternative is to construct a model that provides measurements using alternative data sources as a proxy. This still yields run-time energy samples, but they are sourced indirectly.

This section explains how regression-based techniques can be used to establish the parameters necessary to extract energy consumption from other metrics. The subsequent section then demonstrates an application of this.

4.1. What is regression analysis?

Regression is a method to investigate the functional relationship among variables. The relationship is expressed in the form of an equation or a model connecting the *response* (or *dependent*) variable and one or more *predictor* (or *explanatory*) variables.

The response variable is denoted using Y and the set of predictor variables by X_1, X_2,…, X_p, where p denotes the number of predictor variables. The relationship between Y and the set of predictor variables X can be expressed by a general regression model,

$$Y = f\left(X_1,\ X_2,\ldots,\ X_p\right) + \epsilon, \tag{8}$$

where ε is assumed to be a random error representing the discrepancy in the approximation. The function f describes the relationship between Y and the predictor variables in X. An example of a linear regression model is.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon, \tag{9}$$

where β variables are called regression parameters or regression coefficients, which are unknown constants to be determined (estimated) from the data.

We can decompose the problem of practical regression analysis into the following six steps:

1. Statement of the problem.

2. Selection of potentially relevant variables.

3. Data collection.

4. Model specification.

5. Model fitting.

6. Model validation.

The first three steps require a good understanding of the problem so that a good selection of the predictor variables can be made. These should be variables with a strong relation with the response variable. For example, if trying to model power consumption in a CPU, the operat-

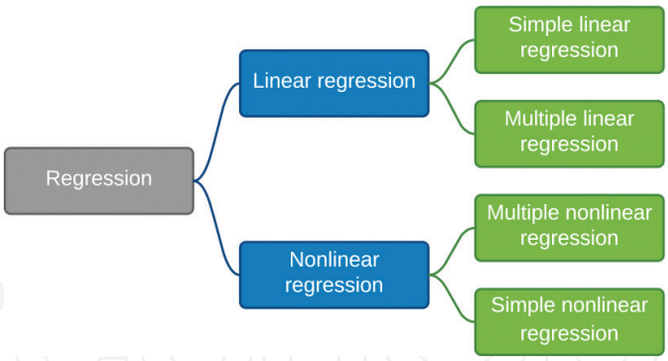


Figure 10. Various categorizations of regression approaches.

ing frequency and voltage will be good predictors to start with together with the number of cycles spent in different processor states, such as idle, active, etc. Others could include the type of instructions (floating point, arithmetic, load/store, etc.), or data on micro architectural events such as the cache miss rate. A good understanding of the problem is important because the effects of some predictors could be unexpected. For example, a high cache miss rate is not desirable, but it will result in an instantaneous power reduction since the core will spend more cycles doing nothing while waiting for data. The data collection step should exercise the predictor variables as thoroughly as possible so that enough data is collected to link the predictors and response variables. Once this data has been collected, the following steps can be performed.

4.2. Model specification

To specify the model, we need to decide what type of regression we are going to use. Regression is divided into two basic types, linear regression and nonlinear regression (see **Figure 10**). Linear regression can be performed with simple linear regression or multiple linear regression and the same applies to the nonlinear case. To solve linear regression, the least squares method is most often used. However, to solve nonlinear regression, variable transformation must be applied on the nonlinear model at first [11, pp. 1405–1411] to obtain a linearized model and then the linear methods can be used with the new linear model.

For example, each of the four following models is linear:

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{10a}$$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \tag{10b}$$

$$Y = \beta_0 + \beta_1 \log X + \epsilon \tag{10c}$$

$$Y = \beta_0 + \beta_1 \sqrt{X} + \epsilon \tag{10d}$$

http://dx.doi.org/10.5772/65989

because the model parameters β are related linearly to the response variable Y. On the other hand,

$$Y = \beta_0 + e^{\beta_1 X} + \epsilon \tag{11}$$

is not a linear model because the coefficient β_1 does not enter the model linearly and the relationship between Y and X is not linear either. To satisfy the assumption of the standard linear regression model, it is sometimes possible to apply an appropriate transformation of the variables to the equation so that the relationship between the transformed variables and the new response variable becomes linear. Instead of working with the original variables, working with the new transformed variables linearizes the model and thus simplifies the approach. Taking Eq. (11) as an example, after applying logarithm on both sides, the equation turns into:

$$\ln Y = \ln \beta_0 + \beta_1 X + \epsilon \tag{12}$$

Now the response variable, ln Y, has a linear relationship with the predictor variable, X, and coefficient, β_1.

As an example, the model equation in a power model could be:

$$P = aV^2 f + bV^3 + cV^5 \tag{13}$$

In Eq. (13), the first term represents dynamic power; the second term is subthreshold leakage and the third term is gate leakage. The gate leakage tends to be very small when compared with the prior two terms and tends to be disregarded. The equation is linear with multiple nonlinear variables, but it does not violate the assumption of a standard linear regression model because coefficients have already entered the equation linearly. Hence, the simplest way is to regard the terms V^2f and V^3 as predictor variables X_1 and X_2, respectively, and then the original model in Eq. (13) becomes a multiple linear regression model, formally described by Eq. (14).

$$P = aX_1 + bX_2 \tag{14}$$

4.3. Model fitting

Model fitting estimates the regression parameters or, in other words, fits the model to the collected data using the chosen estimation method. The estimates of coefficients β_0, β_1, …, β_p are denoted by $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$. In the case of linear regression, the estimated regression equation then becomes

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \ldots + \hat{\beta}_p X_p. \tag{15}$$

The value $\hat{Y}$ is called the fitted value [12] computed using estimated parameters and the values of predictor variables in the observation. Using Eq. (15), we can compute n fitted values of Y for n observations in the data set. Hence, the fitted value $\hat{Y}_i$ in the ith observation is

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \ldots + \hat{\beta}_p x_{ip}, \quad i = 1, 2, \ldots, n. \tag{16}$$

where x_{i1}, x_{i2}, …, x_{ip} are the values of the p predictor variables for the ith observation. These fitted values are the quantities needed for computing *correlation* which can evaluate the goodness of the model. In addition, we can use them to compare with the real value of the response variable y_i and compute the errors (or residuals) in order to evaluate the goodness of the fitting in a different way. This provides the average error which should be within tolerable bounds for the regression to be successful.

To solve simple linear regression, the least squares method is commonly used. It is possible to compute the regression coefficients with scripts for Matlab or Octave or even with a spreadsheet application. Based on the available data, we aim to estimate the value of the regression parameters and find a straight line (or surface for multiple linear regression) that gives the best fit. A best fit means that this fitting could give the smallest sum of squares of errors (residuals). The smallest sum of square of errors is obtained by minimizing the sum of squares of the *vertical distances* from each point to the line. These vertical distances represent the errors between the estimated response variable and the real response variable. Rearranging the equation of a standard linear model as given in Eq. (9), the errors are represented as

$$\epsilon_i = y_i - \beta_0 - \beta_1 x_i, \quad i = 1, 2, \ldots, n \tag{17}$$

The sum of squares of errors (SSEs) can then be written as

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 \tag{18}$$

The values of the two coefficients that minimize the SSE value are given by using the least squares method [13]

$$\hat{\beta}_1 = \frac{\sum (y_i - \overline{y})(x_i - \overline{x})}{\sum (x_i - \overline{x})^2} \tag{19}$$

and

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x} \tag{20}$$

where

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad \text{and} \quad \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{21}$$

http://dx.doi.org/10.5772/65989

are the mean of the response variable and predictor variable, respectively. The estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are called least squares estimates of β_0 and β_1, respectively. They represent the intercept and slope of the line that gives the minimum sum of squares of the vertical distance from each observation point to the line. This line is called the least squares regression line because it is the solution to the least squares method. The least squares regression line is formalized as

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X \tag{22}$$

To compute each fitted value in each observation:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i, \quad i = 1, 2, \ldots, n \tag{23}$$

The vertical distance corresponding to the *i*th observation is

$$\epsilon_i = y_i = \hat{y}_i, \quad i = 1, 2, \ldots, n \tag{24}$$

This kind of vertical distances is called *ordinary least squares residual*. These residuals have satisfied the property that their sum is zero, which means that the sum of distances of the points above the line is equal to the sum of the distances below the line.

4.4. Model evaluation

After fitting a regression model relating the response variable with the predictor variables, it is important to determine the quality of the fit. Covariance and correlation measure the direction and strength of the linear relationship between the two variables (response variable and predictor variable). The quantity correlation is key to evaluate the goodness of the fit. An additional useful measure of the quality of the fit is the R-square value. To obtain the R-square value, there are three quantities that need to be computed:

$$\text{SST} = \sum \left(y_i - \overline{y} \right)^2 \tag{25}$$

$$\text{SSR} = \sum \left(\hat{y}_i - \overline{y} \right)^2 \tag{26}$$

$$\text{SSE} = \sum \left(y_i - \hat{y}_i \right)^2 \tag{27}$$

where SST denotes the sum of squares of the deviations in Y from its mean, SSR represents the sum of the squares due to the regression and SSE stands for the sum of squares of residuals (errors). To understand the significance of these measurements, we can write the following simple relation between observed values and estimated values:

$$\begin{array}{rccc} y_i = & \hat{y}_i & + & \left(y_i - \hat{y}_i \right) \\ \text{Observed} = & \text{Fit} & + & \text{Deviation from fit,} \end{array} \tag{28}$$

then subtracting $\overline{y}$ from both sides of Eq. (28), it becomes

$$\begin{aligned} y_i - \overline{y} &= \left(\hat{y}_i - \overline{y}\right) + \left(y_i - \hat{y}_i\right) \\ \text{Deviation from mean} &= \text{Deviation due to fit} + \text{Residuals} \end{aligned} \tag{29}$$

It is obvious that the total sum of squared deviations in Y, SST, is decomposed into two parts: one is the SSR, which measures the quality of X as a predictor of Y, and the other one is SSE that measures the error in the estimation. The quantity R-square value is the ratio of SSR to SST which tests how accurate the fit is; it removes the contribution of residuals in the SST. Hence, the R-square value, R^2, is written as

$$R^2 = \frac{\text{SSR}}{\text{SST}} = 1 - \frac{\text{SSE}}{\text{SST}}. \tag{30}$$

If the residuals corresponding to the SSE are 0, then the estimation is perfect and the R-square value is 1. Therefore, the closer it is to 1, the stronger the fit.

5. Applying linear regression to estimate power requirements of an ARM processor

The evaluation board uses an ARM Cortex A9 processor equipped with a PMU (performance monitoring unit) that can monitor six performance counters out of a maximum of 58 available events. The event profiling is based in the Linux utility *perf* configured to only monitor events corresponding to the application being executed. We focus on activities related to cache misses, instruction execution and CPU states that have been shown to have a strong influence on power. However, the hardware limitation means that the model is limited to a maximum of six coefficients. The benchmark selected is Mibench [14] and the whole benchmark is divided into two groups so that one group is used for model training and the other group is used for model verification.

The instructions that can be monitored by the PMU are integer instructions, load/store instructions and floating-point instructions. Our experiments show that the integer clock enable state and the data engine clock enable estate have a strong correlation with the integer instructions and offer more accuracy in the model than directly counting the number of integer instructions. On

Predictor variables	Coefficient values
Instruction cache miss	-5.4683×10^{-8}
Data cache miss	-1.4589×10^{-8}
Load/store instructions	4.7787×10^{-11}
Floating-point unit instructions	2.5745×10^{-9}
Integer clock enabled	3.6552×10^{-10}
Data engine clock enabled	3.7001×10^{-11}

Table 1. Coefficients for the final model.

http://dx.doi.org/10.5772/65989

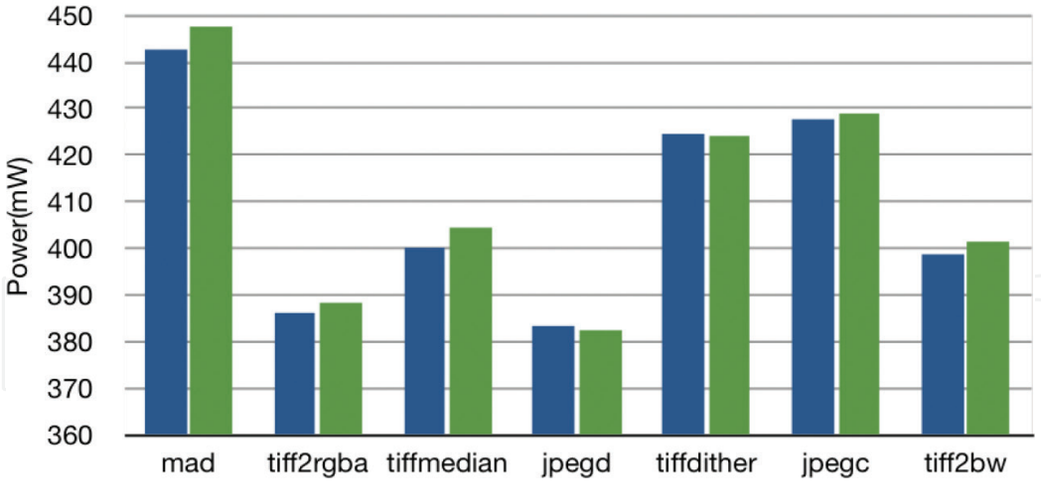


Figure 11. Estimated (blue) and measured (green) average power for the model with Mibench applications.

the other hand, the floating-point instructions and the load/store instructions show a high correlation between the estimated power and the measured power and for this reason are selected.

Both the instruction cache misses and the data cache misses significantly influence power usage since they result in stalls in the pipeline while the data is fetched from main memory. For this reason, the model coefficients shown in **Table 1** have negative values associated with the cache misses. This means that these cache misses result in a reduction in processor power due to additional stalling. Notice that overall, this is not a positive effect since the cache misses will increase power in the memory subsystem and also increase execution time resulting in an overall increase in energy usage.

The values shown in **Table 1** correspond to the coefficients $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_6$ according to Eq. (16) in Section 4.3, while the six events shown in **Table 1** are the predictors. The final constant value $\hat{\beta}_0$ represents idle power or power that is not due to execution of the application. We have measured idle power at a value of 356 mW, and this includes leakage, clock network power and some overhead power due to the Linux OS. **Figure 11** evaluates the goodness of the model with Mibench applications showing that a relative simpler linear model can achieve useful accuracy within 5–10% of measured values.

6. Summary

To reduce energy consumption, it is necessary to understand how much energy a device consumes. Measurement methods are therefore essential not just to help designers and developers understand the behavior of their devices, but also to help them gauge the success of any energy-saving efforts that they make. This chapter has presented several measurement approaches, each with a unique set of properties and potential use cases.

The first general approach, direct measurement, was presented in Section 2. This involves hardware that can detect energy consumption, for which there are several methods, including

current sensing, charge accumulation or transfer measurement and magnetic field sensing. Each measurement circuit has its own level of complexity, precision, range and level of invasiveness with respect to the target hardware. Two example measurement systems are presented in Section 3 and their respective capabilities discussed.

Beyond direct methods of measurement, there is model-assisted measurement. In Section 4, regression analysis methods are presented, which can be used to estimate a device's energy consumption through other properties that can be noninvasively observed, such as performance counters or other events that take place during program execution. Successful measurement of this kind requires that the parameters of the model are carefully selected and understood. Following a demonstration of the construction of a model, an example of a linear regression model, applied to an ARM Cortex A9 processor, is given in Section 5.

Choosing the best approach is dependent upon the device or devices to be measured, as well as the use case and intended outcome. For example, deeply embedded devices may have fewer sources of data to inform a linear regression model, thus direct measurement may be preferred. Or, the deeply embedded processor may be sufficiently simple that an equally simple linear regression model is acceptable. In a more complex system, there may be a desire to know exactly where energy is being consumed, for example, in RAM, buses or a particular type of computation. Without sufficient prior knowledge, this may be difficult to understand without multiple direct measurement points.

Regardless of the measurement method, care must be taken to control, or account for, external factors. An example of this is temperature. The temperature of a device affects its leakage current and therefore its energy consumption. Its temperature will be governed by how active the device is, as well as the ambient temperature, and the ability of the system to remove heat from the device, be it passively or actively. The efficiency of power supplies is also governed in part by temperature as well as load level. Thus, over time, and in different environmental conditions, energy measurements may change for an otherwise unchanged use case.

The permanence, transferability and side-effects of the measurement setup must also be considered. A noninvasive approach can be used on any instance of a device, maximizing transferability, allowing run-time monitoring of a device in a broad set of scenarios. However, this may come with a loss of accuracy, and if the data collection is introspective—collected and processed on the device under test—then the overhead of this effort must also be accounted for. Higher precision typically requires higher effort, as well as additional supporting hardware, removing processing overheads from the device under test, but placing an overhead on the developer in terms of additional equipment, tooling, data collection and analysis. A high sampling rate may seem desirable, but the increase in data collected may be excessive. Devices that are permanently tooled for energy measurement may not be desirable, as in some scenarios, the energy consumption of monitoring will itself be a problem.

Key questions in selecting a measurement method

To summarize, the following questions should be posed in order to guide the selection of an appropriate energy measurement setup.

http://dx.doi.org/10.5772/65989

How many devices need measuring? A one-off tooling versus one that needs replicating many times will influence the preferred method.

What level of detail is required? Establish whether multiple measurement points are necessary, or if a single total energy is sufficient, and what level of precision and accuracy are needed.

What is the overhead? Can the device under test tolerate additional processing burden, or must the data be collected and processed externally? The effort required by the software engineer must also be considered.

Where will the data be used? Run-time decision making requires always-available data, whereas data used to improve a system in development is no longer needed once the product is shipped.

Are there uncontrollable factors? Environmental conditions such as temperature can affect energy, and if they are not controlled, useful data cannot be obtained. Similarly, if the typical operating environment is volatile, then it is more difficult to make concrete assumptions based on data collected in a more controlled (for example, lab or workbench) environment.

Acknowledgements

This chapter is partly based on work performed in the EU 7th Framework Programme project ENTRA: Whole-Systems ENergy TRAnsparency (318337) and the MAGEEC project, supported by the Technology Strategy Board of the UK government under its Energy Efficient Computing Initiative. This work was also partly supported by the German Research Council (DFG) within the Collaborative Research Center SFB 876, project A4.

Author details

Steve Kerrison[1], Markus Buschhoff[2], Jose Nunez-Yanez[1] and Kerstin Eder[1,*]

*Corresponding author E-mail: kerstin.eder@bristol.ac.uk

1 University of Bristol, UK

2 TU-Dortmund University, Germany

References

[1] C. Edwards. Lack of Software Support marks the Low Power Scorecard at DAC. Electronics Weekly, 2011, No. 2472, p 6.

[2] R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu. A study on the use of performance counters to estimate power in microprocessors. IEEE Transactions on Circuits and Systems II: Express Briefs, 60(12):882–886, 2013.

[3] Intel Corporation. Intel Xeon Phi Coprocessor (codename: Knights Corner) Performance Monitoring Units. Intel Corporation, Report number: 327357-001, 2012.

[4] ARM Holdings. Cortex-A9 Technical Reference Manual, revision r2p0. ARM Holdings. Report number: DDI0388E, 2009.

[5] V. Konstantakos, K. Kosmatopoulos, S. Nikolaidis, and T. Laopoulos. Measurement of power consumption in digital systems. IEEE Transactions on Instrumentation and Measurement, 55(5):1662–1670, 2006.

[6] J. Pallister. STM32F4 Energy Monitor. Available from: https://github.com/jpallister/stm32f4-energy-monitor [Accessed Nov. 2016].

[7] S. Kerrison and K. Eder. Modeling and visualizing networked multi-core embedded software energy consumption. Computing Research Repository (CoRR). Report number: abs/1509.02830, 2015.

[8] J. Morse, S. Kerrison, and K. Eder. On the infeasibility of analysing worst-case dynamic energy. Computing Research Repository (CoRR). Report number: abs/1603.02580, 2016.

[9] J. Pallister, S. Kerrison, J. Morse and K. Eder. Data dependent energy modelling: a worstcase perspective. Computing Research Repository (CoRR). Report number: abs/1505.03374, 2015.

[10] M. Buschhoff, C. Günter and O. Spinczyk. MIMOSA, a highly sensitive and accurate power measurement technique for low-power systems. In Real-World Wireless Sensor Networks, Lecture Notes in Computer Science. Springer, Berlin Heidelberg, 2013.

[11] G.K. Smyth. Nonlinear regression. In: Gudmund Høst (ed.), Statistical and Numerical Computing, Encyclopedia of Environmetrics, 4, John Wiley & Sons Ltd, 2006.

[12] S. Chatterjee and A.S. Hadi. Regression analysis by example. John Wiley & Sons, 2015.

[13] E.W. Weisstein. Least squares fitting. Available from: http://mathworld.wolfram.com/LeastSquaresFitting.html [Accessed Nov. 2016]

[14] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown. Mibench. A free, commercially representative embedded benchmark suite. In Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop, WWC '01, pp. 3–14, Washington, DC, USA, 2001. IEEE Computer Society.