Mikko Roth

Measurement platform for Instruction-Level Energy Modeling

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Finland - Espoo - Otaniemi

Thesis supervisor:

Prof. Lauri Savioja

Thesis advisor:

D.Sc. (Tech.) Vesa Hirvisalo

Aalto University
School of Science

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Mikko Roth

Title: Measurement platform for Instruction-Level Energy Modeling

Date: Language: English Number of pages: 55

Department of Media Technology

Professorship: Media Technology Code: IL3011

Supervisor: Prof. Lauri Savioja

Advisor: D.Sc. (Tech.) Vesa Hirvisalo

Energy efficiency is a rising concern in today's society and also affects information technology and software. This concern becomes highlighted in battery powered embedded systems like portable multimedia and smart devices as the available energy is limited. Energy efficiency in information technology is often thought as energy-efficient hardware although it should rather be thought as a joint effort between hardware and software. The energy consumption of hardware depends on what software is running on top of it and how that software uses the hardware. Reciprocally, the software energy consumption depends on the energy efficiency of the hardware. Instruction-level energy models are used for analysing and optimising the energy efficiency of software. The models in turn rely on accurate measurements of energy consumed by individual machine instructions.

Low-energy processors are usually used in embedded systems. Measuring their energy consumption is a challenging task, due to the small currents and noise. It is a challenge to develop a measurement platform for conducting accurate measurements as well as minimising the noise affecting the measurements.

In this thesis, four different circuits were examined to assess their suitability for a measurement platform. A printed circuit board was designed and constructed in order to identify and investigate solutions concerning the challenges in designing such a platform. Within this process a suitable processor was selected to act as the measurement target. A survey of low-energy processors targeted at embedded systems was conducted as part of this work. As a result a prototype measurement platform was constructed.

Keywords: Internet of Things, embedded systems, energy measurement, energy models

AALTO-YLIOPISTO
PERUSTIETEIDEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Mikko Roth

Työn nimi: Mittausalusta Konekäskytason Energiamallintamiseen

Päivämäärä: Kieli: Englanti Sivumäärä: 55

Mediatekniikan laitos

Professuuri: Mediatekniikka Koodi: IL3011

Valvoja: Prof. Lauri Savioja

Ohjaaja: TkT Vesa Hirvisalo

Energiatehokkuus on kasvava huolenaihe nyky-yhteiskunnassa ja koskettaa myös tietotekniikkaa ja ohjelmistoja. Tämä korostuu erityisesti akkukäyttöisissä sulautetuissa järjestelmissä, kuten kannettavissa medialaitteissa ja älylaitteissa, joissa käytettävän energian määrä on rajattu. Energiatehokkuuden tietotekniikassa ajatellaan yleisesti viittaavaan energiatehokkaaseen laitteistoon, vaikka itse asiassa pitäisi puhua pikemminkin laitteiston ja ohjelmiston yhteispelistä. Laitteiston energiankulutus riippuu laitteiston päällä ajettavasta ohjelmistosta ja tavasta, miten ohjelmisto laitteistoa käyttää. Vastavuoroisesti ohjelmiston energiankulutus riippuu laitteiston energiatehokkuudesta. Ohjelmiston energiatehokkuuden analysointiin ja optimointiin käytetään konekäskytason energiamalleja. Mallit puolestaan perustuvat tarkkoihin mittauksiin yksittäisten konekäskyjen käyttämästä energiasta.
Sulautetuissa järjestelmissä käytetään yleisesti energiapihejä prosessoreita. Niiden energiankulutuksen mittaaminen on haasteellista pienten virtojen ja ulkoisten häiriöiden takia. Haasteena on sellaisen mittausalustan kehittämien, jolla mahdollistetaan tarkat energiamittaukset ja minimoidaan mittauksiin vaikuttavat häiriötekijät.
Työssä tutkittiin neljän eri virtapiirin soveltumista mittausalustaksi. Tämän lisäksi suunniteltiin ja valmistettiin piirilevy suunnitteluun liittyvien ongelmien tunnistamiseksi ja ratkaisuvaihtoehtojen selvittämiseksi. Tähän liittyy myös oikeanlaisen mittauskohteen valinta. Samalla tehtiin kartoitus energiapiheistä prosessoreista, jotka on suunnattu sulautettuihin järjestelmiin. Lopputuloksena on rakennettu prototyyppi mittausalustasta.

Avainsanat: Esineiden Internet, sulautetut järjestelmät, energiamittaus, energiamallintaminen

Acknowledgements

I want to thank all the following people:

First and foremost, my parents for all their immense support, encouragement and patience throughout the years, as well as my brothers, relatives and friends for all their support.

My instructor Vesa Hirvisalo from the Embedded Software Group at Aalto University for offering me this opportunity in the first place and his guidance throughout the process, and Professor Lauri Savioja for supervising it.

My host Prof. Dr. Heiko Falk from the Institute of Embedded Systems/Real-time Systems at Ulm University and Nicolas Roeser with whom I had the pleasure to work with in close collaboration during my visit at their institute.

To my colleagues Jussi Hanhirova and Jukka Kommeri from the Department of Computer Science and Engineering at Aalto University for their support. I also want to express my gratitude to all my other colleagues at the department for the interesting and often entertaining coffee room conversations.

Egon Kirsch from the Institute of Embedded Systems at the Hamburg University of Technology for constructing an operational amplifier specially made for the project and his support during the testing of the Energyboard.

The following colleagues and individuals from Ulm University: Prof. Dr. Frank Slomka, Florian Hock, Arno Luppold, Tobias Bund, Victor Pollex, Benjamin Menhorn, Steffen Moser, Barbara Porada, Rajinder Sidhu, Klaus Mammel, Edgar 'Eddy' Zeilmeir and Heiko Ehret.

Dr.-Ing. Dejan Lazich for a long and an interesting discussion about measurement techniques used in the field of embedded security.

Marko Takala and Teemu Ronkka from Electroshop at the Aalto Design Factory for their support and guidance in using their soldering facilities.

I also want to thank TEKES for funding the ParallaX2 project under which this thesis was written and the European COST project TACLe under which the research visit to Ulm University was organised.

Helsinki, May 25, 2015

Mikko Julian Roth

Contents

Abstract 2

Abstract (in Finnish) 3

Acknowledgements 4

Contents 5

Abbreviations 7

1 Introduction 8
- 1.1 Problem statement . . . 8
- 1.2 Contribution . . . 8
- 1.3 Structure of the thesis . . . 9

2 Background 10
- 2.1 Internet of Things . . . 10
- 2.2 Networked embedded systems . . . 12
- 2.3 Heterogeneous System Architecture . . . 13
- 2.4 Energy and power consumption . . . 14
- 2.5 Power management . . . 15
- 2.6 Multi-criteria optimisation . . . 16

3 Modeling energy consumption 18
- 3.1 Classic instruction-level energy model . . . 18
- 3.2 Advanced energy model . . . 19

4 Energy measurement systems 23
- 4.1 Oscilloscopes and probes . . . 23
- 4.2 Operational amplifier . . . 25
- 4.3 Benchmarks . . . 26

5 Measurement circuits 28
- 5.1 Investigated measurement circuits . . . 28
  - 5.1.1 Shunt resistor circuit . . . 28
  - 5.1.2 Current mirror circuit . . . 29
  - 5.1.3 Automated measurement system . . . 30
  - 5.1.4 Advanced setup . . . 31

6 Measurement target 32
- 6.1 Selecting the measurement target . . . 32
- 6.2 ARM Cortex-M3 . . . 35
- 6.3 TMS470MF03107 microcontroller . . . 36

7 Energyboard construction and testing 39

7.1 General PCB design considerations 39
7.2 Design of the Energyboard prototype 40
7.3 Manufacture and assembly 45
7.4 Testing 47

8 Future work and summary **49**
8.1 Future work 49
8.2 Summary 49

References **50**

Appendices **54**

A Energyboard schematics **54**

B Energyboard PCB layout **56**

Abbreviations

ABS	Anti-lock braking system
ACPI	Advanced Configuration and Power Interface
ADC	Analog-digital converter
APU	Accelerated Processing Unit
CAN	Controller Area Network
CMOS	Complementary metal-oxide-semiconductor
CPU	Central Processing Unit
CRT	Cathode-ray tube
DC	Direct current
DSO	Digital Storage Oscilloscope
DUT	Device Under Test
ECC	Error Correcting Code
EPS	Electric Power Steering
FPGA	Field-programmable gate array
GPU	Graphics Processing Unit
HLIR	High-level intermediate representation
HSA	Heterogeneous System Architecture
HMMU	HSA Memory Management Unit
H-CU	HSA Computing Unit
IC	Integrated circuit
IoT	Internet of Things
IPv6	Internet Protocol version 6
ISA	Instruction Set Architecture
JTAG	Joint Test Action Group
LBIST	Logic built-in self-test
LLIR	Low-level intermediate representation
MCU	Microcontroller unit
NCD	Numerically controlled drill
NVIC	Nested Vectored Interrupt Controller
OSPM	Operating system-directed configuration and power management
PCB	Printed circuit board
RoHS	Restriction of Hazardous Substances Directive
SMD	Surface Mounted Device
UART	Universal asynchronous receiver/transmitter
USB	Universal Serial Bus
UV	Ultraviolet
UX	User experience
WCC	WCET Aware C compiler
WCET	Worst Case Execution Time

1 Introduction

Embedded systems are found everywhere in our society: portable multimedia and entertainment devices, intelligent home and office automation systems, industrial automation systems and automotive systems. In many cases, these systems form a network of sensors which themselves are a certain type of embedded devices collecting information on their surroundings. The most recent trend is to connect these systems to the Internet and make them exchange information autonomously creating the Internet of Things.

The raw information collected by the sensors may include a lot of unnecessary information and it would be a waste of resources to send it as such to the network. The earlier the information can be processed the more efficient the exchange of information will be. The sensor nodes are the leaf nodes and generate the data in the first place. The nodes close to the leaf nodes are more likely to run on limited battery power and therefore, they have to be highly energy-efficient. Not only the hardware has to be energy-efficient but the software has to use it wisely beside other restrictions.

Energy models for processors predict the energy consumption of software. The energy models are based on measurements. The quality and accuracy of those measurements lay the foundation for a model and its accuracy, and in the end for the creation of an energy-efficient embedded system.

The focus in this thesis is on performing those measurements for processors used in modern low-energy embedded systems. Because there is no platform specifically designed to function as a measurement platform, this thesis explores the issues of designing such a platform.

1.1 Problem statement

Measuring the energy consumption of single machine instructions is not trivial as the measurements are easily disturbed by noise creating errors that propagate into the energy models. A measurement platform is required for performing these kinds of measurements, and has to be designed and constructed. The biggest challenge is to minimise unwanted noise affecting the measurements.

1.2 Contribution

This thesis presents the results from an investigation concerning different circuits suitable for creating this kind of measurement platform for low-energy processors. This investigation described in section 5 was executed in close collaboration with

Nicolas Roeser from the Institute of Embedded Systems/Real-time Systems at Ulm University.

A prototype was designed, manufactured and assembled to explore the design issues concerning the construction of this kind of platform. The process revealed many aspects that have to be taken into consideration and the findings and results are described in sections 6 and 7.

In the end two prototypes of the measurement platform, nicknamed Energyboard, were built and tested.

1.3 Structure of the thesis

The thesis is divided into 8 sections. A brief description of the overall structure of this thesis is given here:

- **Section 1** is the introduction to this thesis.
- **Section 2** describes the context and background for this thesis.
- **Section 3** explains two different instruction-level energy models in detail. The models are quite different from each other but the basis is the same: both rely on accurate measurements for the energy consumption of single instructions.
- **Section 4** describes measurement systems needed for performing instruction-level energy measurements. Oscilloscopes and a custom-made operational amplifier are described here.
- **Section 5** describes four different circuits which were investigated. The circuits described are designed with the goal to reduce noise in order to perform good quality energy measurements.
- **Section 6** presents a survey of current low-energy processors used in networked embedded systems. Selecting a suitable processor for the prototype platform is not an easy task and the criteria that steered our choice are presented here.
- **Section 7** focuses on the printed circuit board design of the prototype platform nicknamed Energyboard. The layout is divided in five different functional blocks and a description is provided accordingly. A short description of the manufacturing and assembly process is also included in this section. During the testing of the Energyboard it was noticed that the microcontroller does not start up properly. Some measurements were performed and the results are analysed at the end of this section.
- **Section 8** presents the future research possibilities and the summary.

2 Background

This section provides the context and necessary background to this thesis. The emerging Internet of Things is described to provide a context for networked embedded systems. Modern hardware is moving towards heterogeneous computing systems and therefore the heterogeneous system architecture is introduced. Basic physic formulas for energy and energy management techniques are described as well.

2.1 Internet of Things

To begin with, there was the Internet of Computers in which computers are connected to each other and exchange information with each other. This forms the foundation on which the World Wide Web was built in the 1990s. In the first decade of the 2000s, the Internet of Computers evolved into the Network of People which indicates that instead of just consuming content, people actively create content themselves and exchange information with other people. Buzzwords, such as the "Social Web" or "Web 2.0", emerged during this time. The next step is towards the Internet of Things. [8]

Defining the Internet of Things (IoT) paradigm is not easy and still a bit fuzzy. As a result, there are many definitions for it which are influenced by the perspective from which the issue is approached. If emphasizing the functionality of the concept of IoT one could define the Internet of Things as "Things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts" [18]. This means that, for example, a piece of furniture is able to connect to the Internet and exchange information with any other device and react intelligently according to the information, if necessary. The only prerequisite to make the piece of furniture or indeed *any thing* connectable to the Internet is to uniquely identify it from any other device on the Internet and this is not a problem in the foreseeable future with IPv6 becoming more common. [18]

McEwen and Cassimally [28] approach the Internet of Things from a "Thing oriented" perspective with the following equation:

```
Physical Object + Controller, Sensor and Actuators + Internet
= Internet of Things
```

The physical object – the 'Thing' – is worn, carried around or placed at a relevant place – e.g. at home, at work or in a car – for the user. It acts as the container for the embedded system that consists of the "Controller, Sensor and Actuator" part of the equation. The Controller is an "intelligently programmed computer processor" [28]. It can collect data on its surroundings using built-in sensors.

This data is then processed by combining it with information from the Internet. The result can then trigger an output through actuators conveying information to the real world. The data can also be sent over the Internet to a server or service that processes and combines the data with information collected from the Internet. The key element in IoT-devices is that there is some information flow that "connects the defining characteristics of the Thing with the world of data and processing represented by the Internet" [28]. Hence, the processing does not necessarily happen on the device itself. It might even be impractical if the processing is very intensive and requires a lot of processing power as another characteristic of IoT is that there are hundreds or even thousands of these devices. These devices are low-cost and low-power compared to modern personal computers, smartphones or tablets. [28]

This definition resembles an older idea of ubiquitous computing as described in [28]. Basically, the difference is that ubiquitous computing does not include the Internet aspect. As an example the book mentions an air freshener that outputs scent when its sensors notice movement in the room.

The International Telecommunication Union (ITU) mentions the aspect of the size of the Things in the four dimensions it has identified in its report [19] as described in [8]:

- **Item identification:** Tagging things, the ability to identify items through unique addressing systems.
- **Sensors and wireless sensor networks:** Generate data from the real world.
- **Embedded systems:** Provide the capabilities for programming intelligent behaviour to create Thinking Things.
- **Nano-technology:** Refers to shrinking the Things which makes it possible to embed intelligence into almost any object whatsoever.

A prerequisite for the Internet of Things is obviously having access to the Internet. Broadband Internet is affordable and widespread in many countries, especially in Europe and North America. According to Eurostat [10], in the European Union the level of Internet access was 79 % in 2013.

The scale of the IoT is expected to be huge. Billions of Things that connect to the Internet and do intelligent data processing to make people's lives easier. Because of the vast number of Things, the price for the sensors and embedded systems, as well as the energy consumption of a Thing have to be low. In the report [18], low-power microchips and energy harvesting are identified as enabler technologies for IoT. Sensor networks will likely be operating on batteries for long periods of time.

Also, the low-cost requirement supports the idea of low-power and low-energy consumption of the Thing. With billions of Things, as envisioned, even a small energy improvement in each Thing accumulates to significant energy savings.

2.2 Networked embedded systems

Embedded systems are similar to general purpose computing systems and it is not straightforward to differentiate them from each other. Holt and Huang [17] define some characteristics found in but not limited to embedded systems. Differentiating between a general purpose computing system and an embedded system is somewhat fuzzy. Holt and Huang identify a few characteristics associated with embedded systems in their book. These characteristics are not exclusive to embedded systems as they point out.

- *Subsystem of a device or machine:* Embedded systems are often found integrated into larger mechanical or electrical systems. An example would be the different subsystems of a car or a microprocessor inside a household device. Counter-examples of non-subsystems would be network routers or set-top boxes.
- *Dedicated application:* Embedded systems are generally meant for fulfilling a specialised task. In cars, for example, the different subsystems specialise to fulfil complicated tasks like the anti-locking braking system (ABS). A simpler example would be a temperature monitor in a server room that controls an on/off-switch to protect the servers from overheating.
- *Small footprint:* Embedded systems are small. The size is driven by the requirements of the intended task and by the environmental restrictions where the system operates.
- *Low power consumption:* It is typical that embedded systems are operating on limited battery power. Therefore they need to conserve energy to stay functional for as long as possible. In case of non-rechargeable batteries, this determines the lifespan of the system.
- *Real-time systems:* Many of the tasks the embedded systems are designed for have strict time constraints. The airbags in a car need to be launched within a certain time window to protect the passengers inside the car in case of an accident. In this case, taking any longer is likely to be fatal for the passengers.

The trend in the past three decades has been to build networks of embedded systems, also called distributed embedded systems. One of the most important drivers for this trend has largely been the need to replace wired point-to-point connections with a single communications bus. The network may consist of either wired or wireless communication bridges or a mix of both. A typical example of wired networked embedded systems is from the automotive industry. A car has a collection of embedded systems ranging from ABS and engine control to the audio system and window lift controls which are all networked by in-vehicle network interfaces like Controller Area network (CAN). [25]

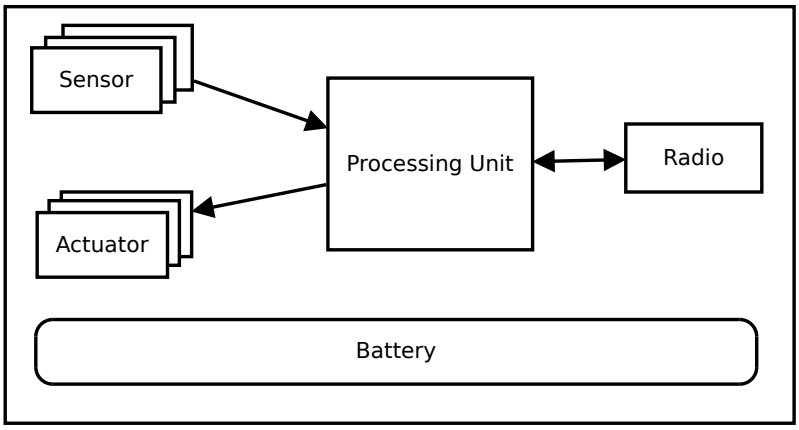


Figure 1: Sensor node architecture.

Another example is a wireless sensor network that allows flexible deployment and maintenance. Figure 1 shows a generalised architecture of a wireless sensor node. Sensors are nodes in the network expected to collect information on their surroundings using one or more sensors and forward the collected information more or less preprocessed to a central node for further use. A sensor could be simple as a push-button or a complex system like a video camera. The node could also include one or more actuators. An actuator transforms an electric signal into an observable event. For example, turning on or off a LED or moving a robotic arm.

The sensor node operates on limited battery power that is either rechargeable or non-rechargeable depending on the application. Wireless communication uses more energy than local computation [31] which encourages to aggregate and preprocess the collected data to send only essential information over the radio link. This makes sense in cases where a lot of data is generated by the sensor or sensors and it is possible to do some preprocessing already on the sensor node itself. The processing unit is likely to have specialised hardware to do the preprocessing and a general purpose processor to handle communication and controlling of the sensor node.

For instance, if the situation is following: The sensor is a videocamera and we are only interested in objects identified in the videostream. Instead of sending the raw videostream through the network, the stream is processed on the sensor node video processing hardware and just the data objects are sent further.

2.3 Heterogeneous System Architecture

The heterogeneous system architecture (HSA) is addressing four challenges the computing industry is facing today: power, performance, programmability and portability. Reducing power consumption is important across all computing segments: data centres are faced with rising power and cooling costs. Consumers expect improved

battery life for their portable devices. At the same time the continuous improvement in computing performance is expected to enable new user experiences (UX) and features. Programmability enables programmer productivity by using familiar programming models. Portability is important as it means the software can run on a variety of different hardware without the need of rewriting the code as is often the case today. [23]

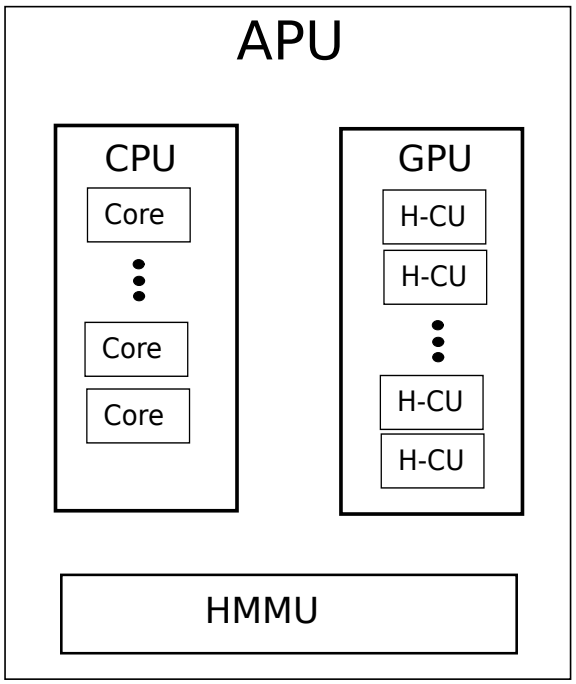


Figure 2: HSA Accelerated Processing Unit.

Instead of having a separate central processing unit (CPU) for general purpose computing and a separate graphics processing unit (GPU), the HSA Accelerated Processing Unit (APU) combines them into one platform with shared coherent memory. Figure 2 shows a simple APU that consists of a multi-core CPU for general purpose computation and a GPU with multiple HSA Computing Units (H-CU) for parallel processing. The HSA Memory Management Unit (HMMU) controls and manages the access to system memory to keep it coherent and shared between the CPU and GPU. This makes the memory management transparent to the application software running on the APU. [23]

2.4 Energy and power consumption

Energy is a physical quantity that describes how much work a physical system can do to another system. Both energy and work is measured in joules (J) after the physicist James Prescott Joule in the International System of Units (SI). The work

performed or energy consumed in a circuit in the time interval $[t_1, t_2]$ is therefore:

$$E = \int_{t_1}^{t_2} (I(t) \times V(t)) \mathrm{d}t \tag{1}$$

The focus of this thesis is on the electromagnetic energy consumed in integrated circuits, especially digital CMOS circuits such as single-core microcontrollers. To find out the energy consumption of a circuit, the current I in equation 1 needs to be solved.

$$I = \frac{V}{R} \tag{2}$$

In equation 2, resistance R denotes an objects property which describes its opposition to the flow of current through it. The SI unit of resistance is ohm (Ω) after physicist Georg Ohm.

If an electric potential difference or voltage V is imposed across a resistor, a current flows through it. Resistors are conductive objects that have a specific resistance. The quantity V equals the *energy gained per unit charge as charge "falls" through the potential difference* [45]. In case of a resistor, this energy is transformed into heat. Therefore, the rate of doing work or power can be expressed in multiple forms by applying equation 2:

$$P = I \times V = I^2 \times R = \frac{V^2}{R} \tag{3}$$

These fundamental laws of physics essential to energy modeling and measurements are described in many good quality physics books, such as [45].

2.5 Power management

Software consumes energy during its execution by affecting the state of the CMOS circuit. The energy consumption of the CMOS circuit can be divided in two parts: dynamic energy and static energy. Dynamic energy consumption represents the processing of the software as it changes the circuit state. A circuit state change is the transition from logic '1' to logic '0' or vice versa.

Static energy consumption is the result of leakage currents. Leakage currents are small undesired current flows through diodes and transistors. In ideal circuits no leakage would happen but in reality they cannot be avoided and are dependent on

the manufacturing process. Current leakage increases as CMOS planar circuit size is scaled down. [36]

Advanced Configuration and Power Interface (ACPI) [16] is a collection of interfaces for power management functions to establish industry common Operating System-directed configuration and Power Management (OSPM). The goal is to establish a common way for all operating systems to configure the motherboard and manage power related issues.

There are two common technologies used: dynamic voltage and frequency scaling (DVFS) and low-power states. DVFS trades performance against energy savings. Changing the frequency and voltage of the processor affects the execution speed of instructions in exchange for energy consumption. The processor is unavailable to execute tasks during the frequency transition as the voltages and clocks must first stabilise to adapt to the new operating frequency. Switching from higher to lower frequencies is faster than the other way round. [36, 20]

Low-power states are an efficient way to gain energy savings as the systems unused parts are shut down by cutting off their power supply. The energy savings are larger when compared to reducing the operating frequency and voltage like in DVFS as both the dynamic and static consumption are removed.

2.6 Multi-criteria optimisation

Embedded systems usually have to meet several design constraints like energy consumption, code size and worst-case execution time (WCET). Because it is difficult to determine strict upper bounds for the energy consumption, energy minimisation is usually chosen as the multi-criteria optimisation target. The WCET and code size on the other hand are often hard constraints: Maximal code size constraints are given by the physical memory available on embedded systems. [35]

The current approach of manually analysing the compiled binary and tweaking the compiler input iteratively is tedious and time consuming. Automating the procedure and letting the compiler do the optimisations would seem a desired alternative. Different optimisation goals require expertise knowledge in different areas and the WCC compiler provides a noteworthy option. Also as noted in [44], energy consumption follows closely the execution time so that optimising the code improves the execution time which can result in reducing energy consumption.

The WCET Aware C Compiler WCC automates the WCET minimisation procedure. It achieves this by integrating the WCET analysis into the compiler and utilises the results to reduce the WCET. A simplified overview of this procedure is presented in figure 3. It is the first and currently the only fully functional automated WCET aware compiler.

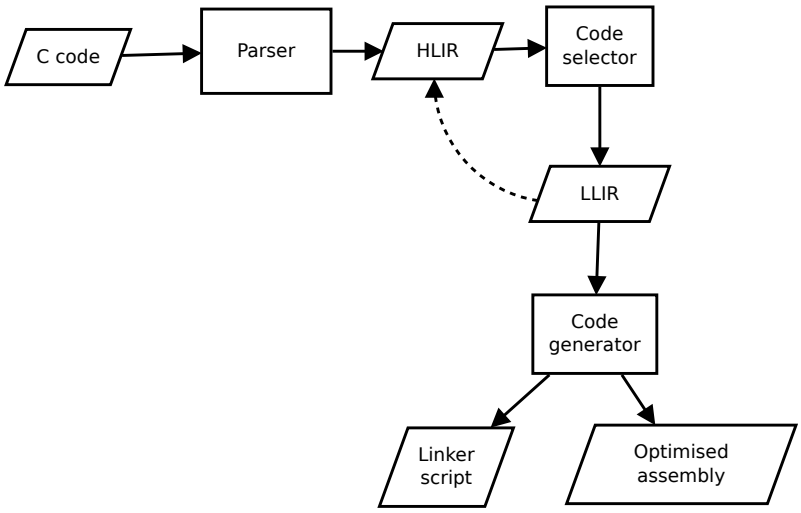


Figure 3: WCC framework.

The parser accepts C code files and generates a high-level intermediate representation (HLIR). The HLIR is used for loop analysis and other high-level WCET optimisation purposes. This optimised version is then translated into a low-level intermediate representation (LLIR). The LLIR is analysed by a static WCET analyser tool integrated into WCC and the results are imported into LLIR-objects which can store arbitrary data for different optimisation purposes. Various optimisations, such as loop unrolling, procedure cloning etc., are used for minimising the WCET. Some of the results are communicated via back-annotation to HLIR to exploit high-level optimisation methods. This is represented by the dashed arrow in figure 3.

The optimised code is then passed on to the code generator which outputs assembly code and linker script to create the WCET optimised binary. The modular structure of WCC and the extensible LLIR-object structure are intended for implementing future extensions to support other optimisation goals like energy consumption. [11, 12]

3 Modeling energy consumption

For a successful integration of energy consumption optimisation at compiler level, the energy consumption needs to be modeled. The term 'energy cost' is used to describe the energy consumed by something. One of the simplest models would be the multiplying of an averaged energy cost E_{avg} by the number of instructions N in the program to obtain the total energy cost E_P.

$$E_P = E_{avg} \times N \tag{4}$$

However, this model would very likely hold true only for the program where the average was derived from and thus useless to model arbitrary programs. Different instructions tend to have different energy costs depending on active parts in the processor and also other aspects, like state switches, pipeline stalls, memory accesses and so on, affect the total consumption of a program.

The models presented in this section are intended especially for energy consumption modeling. However, there are models created for so called power attacks in the field of embedded security and they could potentially be exploited for use in energy modeling meant for energy optimisations [26].

This section presents the first instruction-level model that was introduced in the mid-90s and a newer one from 2013.

3.1 Classic instruction-level energy model

Software energy consumption can be modeled by instruction level power analysis. The first instruction-level model described in this section was presented in [43, 44]. The measurement setup used was basically the same as the circuit described in section 5.1.1.

The total energy consumption of a program E_p is expressed with the following equation:

$$E_p = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) + \sum_k E_k \tag{5}$$

The following factors contribute towards the energy cost of a program:

- B_i - Base cost for each instruction i.
- N_i - Number of execution times for each instruction i.

- $O_{i,j}$ - Circuit state change overhead for each instruction pair (i,j).
- $N_{i,j}$ - Number of execution times for each instruction pair (i,j).
- E_k - Energy cost for other inter-instruction effects (stalls, cache misses etc.).

Instruction base cost B_i which is the cost associated with the basic processing needed to execute the instruction. This can vary depending on the value and address of the operands but is in practice negligible, below 10 %. The base cost is then multiplied by the number of times each instruction is executed. Operand values affect the base energy cost but the difficulty is that the exact values are usually not known before the execution time and thus the base cost has to be an average.

The circuit state overhead $O_{i,j}$ takes the energy contribution of circuit state changes into account. Circuit state changes occur when executing two different instructions consecutively. Without this, the energy cost of the whole program would be underestimated.

For finding out the overhead the basic energy costs of the two consecutive instructions, i and j are subtracted from the total energy cost of executing instructions i and j consecutively:

$$O_{i,j} = |E_{i,j} - (B_i + B_j)| \tag{6}$$

Other inter-instruction effects E_k may result during the execution of the program like cache misses and pipeline stalls and are added to the final result. However, the penalty caused by e.g. a pipe stall is hard to predict and causes varying errors in the predictions. In [43], the predictions had generally a 3 %-error in test cases with no pipe stalls or cache misses. Such cases are not realistic in real world scenarios.

3.2 Advanced energy model

The energy model proposed in [6] and described in this section aims to model the energy consumption of the CPU, Flash memory, SRAM and the memory controller which should provide more accurate estimates than the classic model introduced in section 3.1. The memory model is still quite simple as no cache hierarchy is modeled and does not include the energy consumption of read-only memory (ROM) or peripherals either.

A typical program for embedded systems has two parts: the initialisation phase and the main phase. The initialisation phase is only executed once at startup. The main part is basically often just an endless loop. The implication is that the energy consumption for the initialisation phase is negligible in regard to the relatively long

running main phase. The initialisation phase can safely be discarded from the energy analysis without losing much accuracy.

When an instruction is executed, energy is consumed during the different stages of the CPU pipeline. Because of that, the total energy cost of the instruction can be expressed as a sum of the energy consumed in the different pipeline stages. The processor used in [6] is an ARM7TDMI which has a three-stage pipeline: instruction fetch, instruction decode and instruction execute. [4]

Most modern processors make use of instruction parallelism by pipelining instruction execution. Instead of executing one instruction fully during one clock cycle of a processor, multiple instructions can be in execution at the same time. This increases the processor throughput and clock frequency. A basic pipeline consists usually of three stages: instruction fetch (IF), instruction decode (ID) and instruction execution (EX). In practice, there are often more stages like memory, mathematical or other intermediate stages inside a processor. [33]

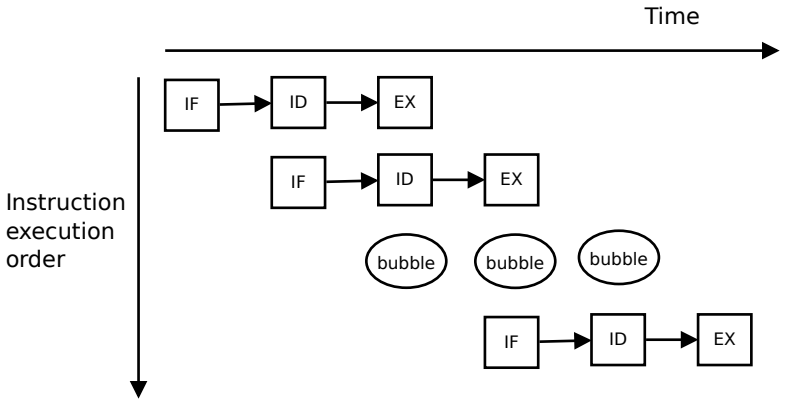


Figure 4: Pipelined instruction execution in a three-stage pipeline.

Figure 4 illustrates how a pipelined execution works featuring a three-stage pipeline matching with the ARM7TDMI pipeline mentioned before. Each horizontal line corresponds to one instruction going through each stage of the pipeline. Each stage takes one clock cycle and only one instruction can be in a stage at a time. The third instruction, marked as a bubble moving through the pipeline stages, indicates a pipeline stall when basically a no-operation instruction moves through the pipeline [33]. Each stage consumes energy and the total energy consumption of an instruction is the sum of the energy consumption used in each of the stages. In [6], the memory and memory controller related energy is added to the relevant stages as explained next.

The energy cost of the fetch stage consists of the CPU consuming energy E_{IF} calculating the memory address of the next instruction. The memory controller is then ordered to fetch the next instruction of the program code. The memory controller in turn orders the Flash memory to read the instruction from the specified address. The energy cost associated with the memory controller is denoted as $E_{ctrl(code)}$ and

with the Flash memory as $E_{Flash(data)}$.

The energy cost of the decode stage consists of the CPU being active. The energy consumed is marked as E_{ID} which can be assumed to be constant in case of the ARM7TDMI processor as it uses the fixed length 32-bit ARM instructions.

The execute stage is the most complex one. In the processor the execution may activate different hardware depending on the type of instruction. The execution stage either solves an arithmetic or a logic operation or triggers a memory operation. The energy consumed is marked as E_{EX}. In case of an operation that accesses the memory, the memory controller is activated and either the SRAM or Flash memory is accessed to read or write data. The associated energy costs are $E_{ctrl(data)}$, E_{SRAM} and $E_{Flash(data)}$. Pipeline stalls are possible during memory operations when the processor needs to wait for the memory operation to finish and the energy cost ensuing from this is denoted as E_{stall}.

$$\begin{aligned} E_{process} &= E_{ctrl(code)} + E_{Flash(code)} + E_{IF} + E_{ID} + E_{EX} + E_{stall} \\ E_{memory} &= E_{ctrl(data)} + E_{Flash(data)} + E_{SRAM} \end{aligned} \tag{7}$$

In equation 7, the different energy costs are divided into three different types: The energy cost associated with fetching and processing an instruction $E_{process}$, the energy cost associated with memory data accesses E_{memory} and static energy consumption E_{static}. Static energy is the energy consumed by hardware that is supplied with current but inactive as explained in section 2.5.

$$\begin{aligned} E_{memory} = N_{Flash(read)} \times \lambda_{Flash(read)} \\ + N_{SRAM(read)} \times \lambda_{SRAM(read)} \\ + N_{SRAM(write)} \times \lambda_{SRAM(write)} \end{aligned} \tag{8}$$

The E_{memory} is rewritten in equation 8 to count the number of read and write operations. The write operations to Flash are omitted from the equation as write operations are rare in the normal operation of embedded systems.

$$E_{core} = E_{IF} + E_{ID} + E_{EX} \tag{9}$$

In equation 9, the CPU pipeline stages found in $E_{process}$ equation 7 have been combined into one E_{core} energy cost which is comparable to equation 5 in the classic model described in section 3.1. Because the inter-instruction costs are only about 5 % of the pure base cost as shown in [22], the inter-instruction costs are ignored in [6] and instead, the E_{core} energy consumption is estimated by the following parameters:

- Instruction word Hamming distances $I_{Hamming}$.

- Instruction word weight I_{weight}.
- Number of shift operations N_{shift}.
- Register bank bit flips $R_{bitflip}$.
- Pipeline stalls (both fixed and variable length).

When comparing two arbitrary strings, the Hamming Distance is the number of positions in which the two strings differ from each other. In the instruction word context it means how many bit positions have to be flipped to the opposite – '1' to '0' or vice versa – in order to transform the instruction into the other one. Flipping a bit costs more energy than not needing to flip it.

The instruction word weight is basically the count of 1's in the binary representation. It costs more energy to have more bits equalling '1' than '0' in an instruction. The number of shift operations equals the count of shift operations in the program.

The register bank bit flips are related to the Hamming distance as this parameter measures the effect of flipping bit in the registers as a result of executing instructions.

The pipeline stall parameter measures the energy cost of pipeline stalls. In [6], this parameter is approximated by an average value gotten from a benchmark suite.

Finally, E_{static} is considered to be a constant value in the model and is added in the base cost of an instruction. Thus, the final form of the equation is as follows:

$$E_{program} = \sum_{p \in P} N_p \times \text{coeff}(p) \tag{10}$$

where $p \in P = \{$Processor instruction set, $\lambda_{SRAM(R)}$, $\lambda_{SRAM(W)}$, $\lambda_{Flash(R)}$, $I_{Hamming}$, I_{weight}, $N_{shift}\}$.

The coefficients are computed using regression analysis. It is a statistical method where the coefficients are adjusted using training data. In this case it would mean that the training data are test programs for which the total energy consumption is known and the coefficients of the model are adjusted to produce sufficient estimates for the total energy consumption of the test programs. The more test programs that stress different parameters, the better will the model be adjusted. [6]

4 Energy measurement systems

First, an overview of oscilloscopes and probes is presented as they are crucial for conducting meaningful and accurate measurements. Furthermore, an operational amplifier custom-made for this project is described in this section. Lastly, both microbenchmarks and application benchmarks, and their use in these kinds of measurements are described.

4.1 Oscilloscopes and probes

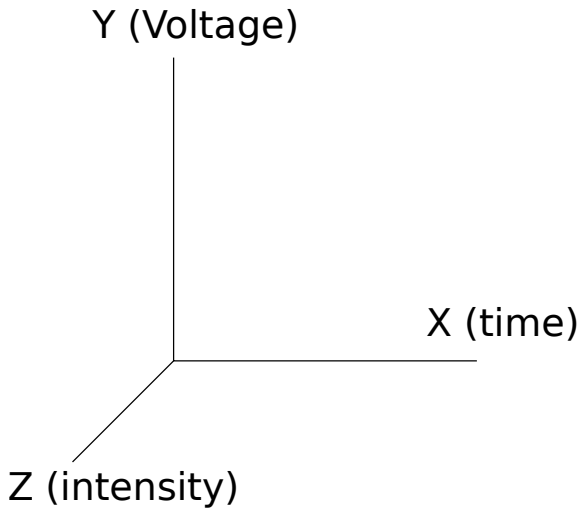


Figure 5: Three dimensions of an oscilloscope.

An oscilloscope is a device that draws a graph of an electrical signal onto a display. Its usual application is to show signal change over time. The components of the displayed waveform are the X-axis representing the time dimension while the Y-axis represents the voltage dimension. The intensity of the displayed waveform is considered to be the Z-axis. The dimensions are illustrated in figure 5. [39]

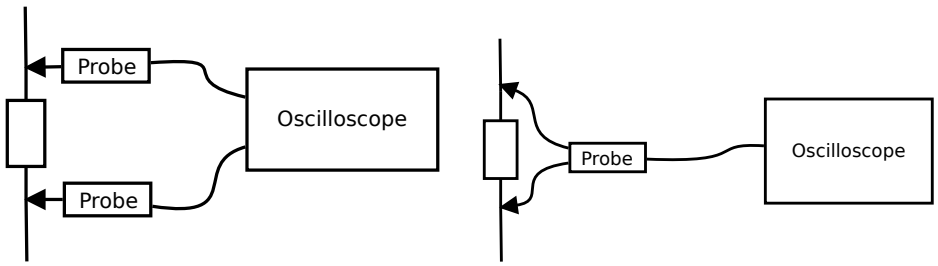
There are two types of oscilloscopes: analog and digital. An analog oscilloscope works with a continuously variable signal by applying it directly to the electron beam of the cathode-ray tube (CRT) display. This makes analog oscilloscopes ideal for tracing rapidly varying signals that need to be followed real-time. Analog oscilloscopes use a phosphor-based display that features a characteristic known as "intensity grading" which relates to the Z dimension in figure 5. It makes signal details more easily distinguishable by displaying the trace brighter where signal features occur more often. [39]

Digitising oscilloscopes on the other hand digitize the continuous analog signal through an analog-digital converter (ADC) into discrete binary values. The conventional digital oscilloscope is known as a Digital Storage Oscilloscope (DSO).

The DSO constructs the displayed waveform by sampling the signal using various methods. Real-time sampling works for signal frequencies up to a half of the oscilloscope's maximum sampling rate. It captures a sufficient amount of samples in one cycle to reconstruct the waveform accurately enough. It is the only method to capture transient events.

Transient events are signals that happen once or only rarely. The other methods that are used for signal frequencies larger than half of the oscilloscope's sampling rate miss those transient events. Interpolated real-time capture method interpolates the gaps in between the samples to make an estimation of the real signal. Lastly there is the method of equivalent-time sampling where samples are collected from successive cycle until enough samples have been obtained to reconstruct a repeating signal. [39]

A fundamental part of an oscilloscope is the probe of which there are many different types. Two major categories are passive and active voltage probes. By far, the most common probes used are passive ones as they are the most durable and simplest type of probes. Compared to active probes they have a higher source loading effect. The probe becomes a part of the circuit and draws some of the current from the circuit to create a signal for the oscilloscope. The probe tip causes both resistive and capacitive load on the circuit. The capacitive load is usually of a greater concern as it affects the maximum bandwidth and capacitive reactance. [38]



(a) Two single-ended probes. (b) Differential probe.

Figure 6: Methods of measuring a differential signal.

The most common probes are for measuring single-ended signals; i.e., signals that are referenced to the ground. When, however, two signals are referenced to each other we talk about differential signals. There are two ways of measuring a differential signal illustrated in figure 6. The first way as shown in figure 6a is to use two single-ended probes and subtract second signal (signal A – signal B) from the first one to obtain the differential signal. The use of this setup is however discouraged because of problems such as time skewing of the two signals and poor common-mode signal rejection. [38]

Time skewing happens if one of the signals has any delay in relation to the other signal which can produce amplitude and timing errors in the computed differential signal shown on the oscilloscope. The probe signals are affected by noise as any

other signal. Because the two signals are not combined until in the oscilloscope, the computed differential signal is a result of subtracting a noisy signal from another noisy signal.

Both problems mentioned above are minimised by using a differential probe as shown in figure 6b in which the combining of the two signals is already done in the probe. This generally improves the common-mode signal rejection rate. Most of the measurement circuits described later in section 5 rely on measuring a voltage drop over a resistor which in essence is measuring a differential signal. [38]

As a side note, there are methods developed in the field of embedded security to measure and model the energy consumption of processors. Some methods do not require a probe to be physically connected to the circuit, like e.g. electromagnetic emissions that can be observed from a short distance. The positive aspect is that these kinds of methods do not require any special test pins or other hardware support from the device which is to be measured. However, the measurements are easily much noisier and require special measurement systems that are hard to obtain. For these reasons, it is not possible to use these kinds of measurement systems in context of this thesis. [26]

4.2 Operational amplifier

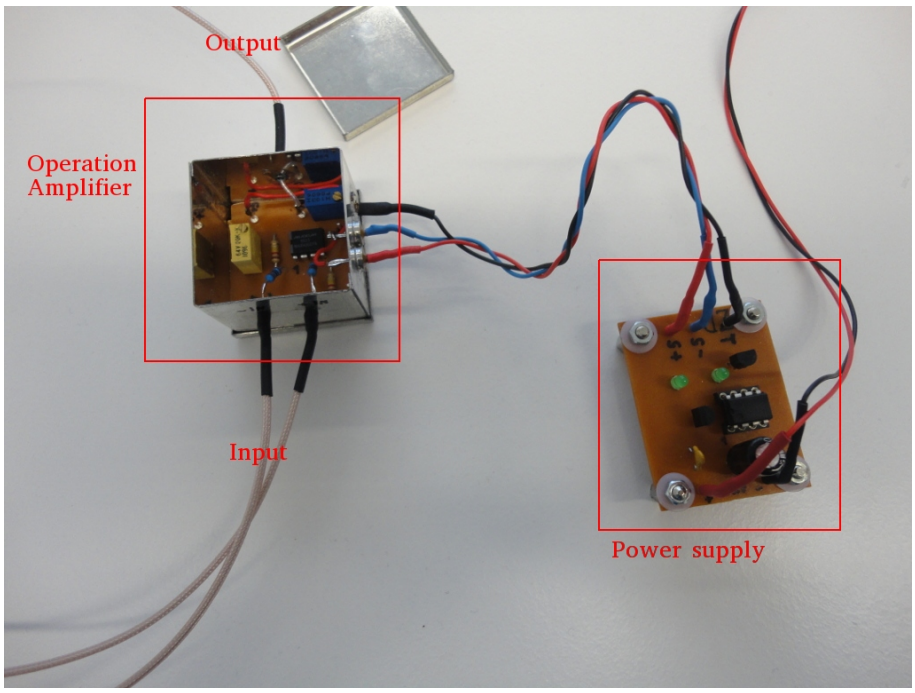


Figure 7: Operational amplifier by Egon Kirsch.

An operational amplifier is an amplifier for direct current circuits. It is used for amplifying the potential difference of a differential input to a higher potential difference

in the output. [42] The operation amplifier in figure 7 was custom-made for use in this project by Egon Kirsch from the Institute of Embedded Systems at Hamburg University of Technology.

It is designed to amplify the potential difference over a shunt resistor without influencing the energy consumption to be measured by having a completely separate power supply. It is connected between the measurement pins and the oscilloscope. Usually operational amplifiers are integrated into the circuit in which they are used. An operational amplifier with a separate power supply reduces unwanted noise produced by the operational amplifier. An initial test with a signal generator indicated that it is able to amplify signals of a frequency up to 1 MHz.

4.3 Benchmarks

Benchmarks are used for calibrating and evaluating energy measurements. Microbenchmarks are used for evaluating characteristics of the hardware's microarchitecture, such as energy consumption. In [43], the microbenchmarks stress continuously a single architectural component of the processor while measuring the energy consumption. The microbenchmarks for measuring the base cost of individual machine instructions consist of a short startup code for the hardware followed by a loop of 120 instances of the same instruction. The branching code for looping causes a small error in the measured averaged base cost of an instruction. Loops of two alternating instructions were created in a similar fashion for measuring inter-instruction effects. The startup code is executed only once so that it does not affect the measurement. [43]

Microbenchmarks are well suitable for measuring the energy consumption for individual instructions and inter-instruction effects in order to create energy models. The models should be evaluated with application benchmarks that represent much more realistic real-world software. MediaBench is a benchmark suite designed for multimedia and communication applications running on embedded systems. MediaBench was introduced in 1997 [27]. The MediaBench aims to represent accurately realistic multimedia and communication workloads intended for embedded systems. MediaBench is first benchmarking suite stressing complete applications that were written in high-level languages and was published back in 1997. The 11 components of MediaBench are:

- JPEG - Still image compression encoding and decoding.
- MPEG - Video compression encoding and decoding.
- GSM - Speech transcoding in 2G mobile networks.
- G.721 - Voice compression encoding and decoding.

- PGP - IDEA/RSA public-key encryption algorithm for encrypting, decrypting and signing messages.
- PEGWIT - A public-key encryption and authentication program.
- Ghostscript - PostScript interpreter. Test file I/O.
- Mesa - A clone of the OpenGL graphic library. Includes the following applications: mipmap, osdemo and texgen.
- RASTA - A speech recognition program using various techniques.
- EPIC - Experimental image encryption.
- ADPCM - A classic audio coding technique: adaptive differential pulse code modulation.

MediaBench has been extensively used for almost 20 years but multimedia related technology has advanced since. MediaBench II is the next generation benchmark and beside just using more recent test programs it divides the benchmark into 6 different benchmark suites for different application areas: video and image, audio, speech, security, graphics and analysis. A seventh suite is the composite suite which will consist of two representative applications from the other suites. Furthermore, there is a kernel suite which does not consist of full applications but kernels are often found in multimedia applications. So far only the video and image suite has been completed. [13, 29]

5 Measurement circuits

The basic building blocks of software are machine instructions. The instructions trigger circuit state changes which are the source of energy consumption. If software uses hardware non-optimally, the result may be significant for the software energy consumption. For example, an unfortunate order of the machine instructions – e.g. having vastly different instruction types ordered one after each other – can result in expensive state changes in the circuit. Reordering the instructions by minimizing expensive state changes, without altering the software program's logic, would improve the energy efficiency of the program up to 40 %. [44]

This section presents the results of an investigation concerning different measurement circuits. The investigation was conducted in collaboration with Nicolas Roeser from the Institute of Embedded Systems/Real-time Systems at Ulm University.

5.1 Investigated measurement circuits

According to the feedback we received about the plan to use a development board as a measurement platform was regarded as unqualified for our purposes. The differentiation of the processor energy consumption from other hardware would most probably not succeed. For example, the on-board USB voltage regulator would cause a lot of noise. Based on this feedback, we conducted a survey of different measurement circuits which we could use for constructing our own measurement platform tailored for measuring processor energy consumption.

The measurement circuits presented in this section were our proposals for measuring instruction-level power consumption. The circuits were presented in a position paper and distributed to colleagues mainly at Ulm University to receive feedback and collect suggestions for the most suitable measurement approach. Most of the feedback was collected in face-to-face conversations.

5.1.1 Shunt resistor circuit

The first attempt to model the energy consumption of a processor was made in the mid-90s by Tiwari et. al [43]. Their technique was based on measurements performed on real hardware executing different instructions. They measured the average current drawn from the power supply using an ammeter. This is equivalent to measuring the voltage drop over an accurately known resistor a.k.a. a shunt placed in series with the processor. The shunt resistance is usually very small so that the influence on the circuit is kept as small as possible.

This kind of measurement circuit is visualized in figure 8. 'R' is the shunt resistor

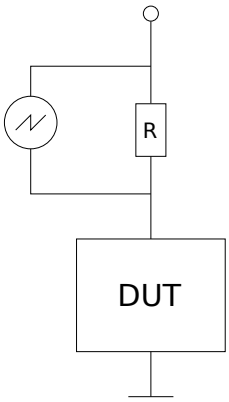


Figure 8: Classic shunt resistor measurement circuit.

and 'DUT' is the Device Under Test which in this case would be the processor.

Measurement circuits that insert a current sensing device – e.g. an ammeter or shunt resistor – into the device's power supply path affect the current to be measured. According to Nikolaidis and Laopoulos [32], the voltage drop degrades the operating performance of the device due to voltage variations.

5.1.2 Current mirror circuit

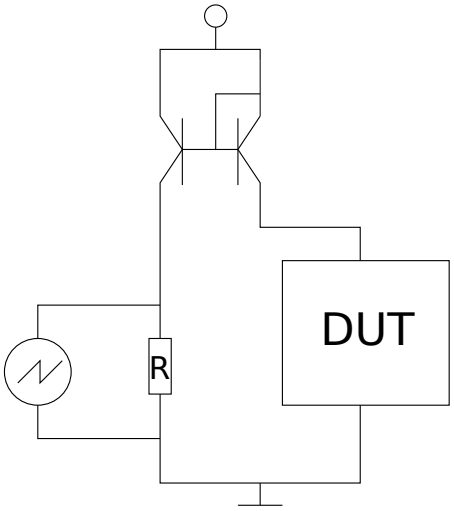


Figure 9: Current mirror measurement circuit.

Laopoulos et al [24] propose a measurement circuit that is based on a current mirror for creating a current sensing circuit. The current mirror creates a copy or mirrors of the instantaneous current drawn by the processor. Placing the shunt resistor on the mirrored side minimises the influence on the supply current for the processor. Figure 9 shows an example of this kind of circuit.

A current mirror is a circuit block that produces ideally an exact copy of a current. The high output resistance of the current mirror keeps the output current constant regardless of the load. [42]

5.1.3 Automated measurement system

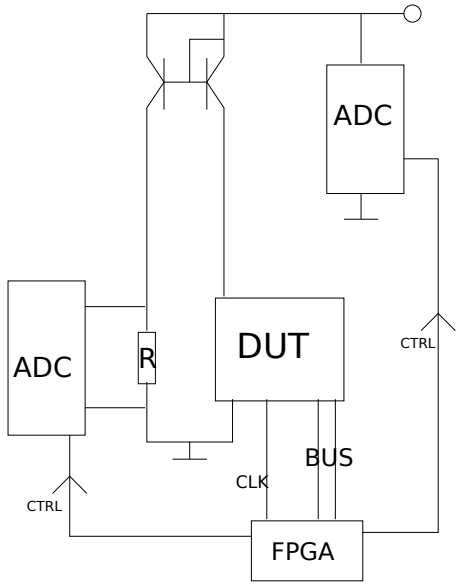


Figure 10: Automated measurement system by Prof. Slomka from Ulm University.

The circuit shown in figure 10 is a sketch for an automated cycle-accurate measurement circuit based on a current mirror. This circuit was conceptualised by Prof. Slomka from the Institute of Embedded Systems / Real-Time Systems of Ulm University. The analog digital converter (ADC) measures the voltage drop over the shunt resistor 'R'. The second ADC in the top-right corner in the schematic is for measuring the supply voltage which can also fluctuate and is added for achieving better accuracy. This measurement circuit is able to evaluate the energy consumption of arbitrary software.

The automation is accomplished by the field programmable gate array (FPGA) that shares the same system clock with the processor to be able to conduct cycle-accurate energy measurements. The FPGA implements the control logic for the measurements. It reads the output of the two ADCs at each cycle and evaluates them to get the current for each instruction executed.

The FPGA has to store a copy of the program and snoop the bus or feed instructions to the processor itself in order to know which instruction is being executed. It also stores a table of the number of clock-cycles that each instruction requires for its execution.

It takes time to read and evaluate the output from ADC. The processor has either to

stop execution until the FPGA is ready to evaluate the next instruction or leave the following instructions unevaluated. The solution is straightforward by duplicating the ADCs. Then the FPGA can switch ADCs between instructions allowing for uninterrupted execution of the software executing on the DUT.

5.1.4 Advanced setup

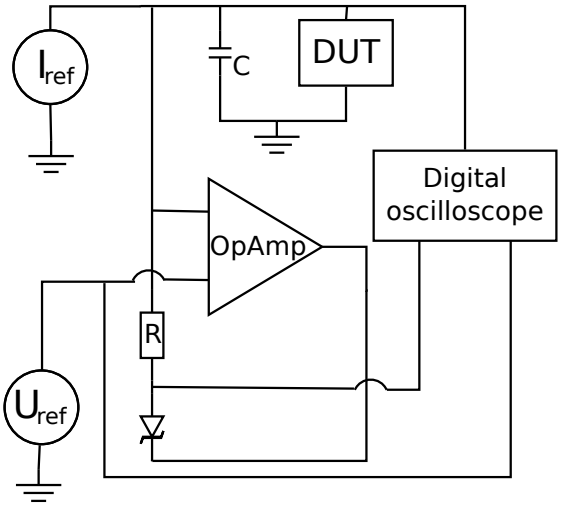


Figure 11: Advanced circuit by [7].

The circuit shown in figure 11 is a circuit for measuring the instantaneous current of a device that consumes current in a pulsating fashion like processors or microcontrollers. The circuit uses a current stabiliser as its current source. Basically, the current is kept constant while the voltage changes. The device needs a stable voltage in order to operate normally. Sudden spikes in the current drawn by the device changes the voltage however, and this needs to be stabilised using a capacitor as a local current source. These are called decoupling capacitors.

When measuring a voltage it is crucial to have the correct reference potential a.k.a. ground. The resistor and Zener-diode are added to the circuit to stabilise the reference voltage so that the digital oscilloscope gets correct readings. The operational amplifier is an addition to compensate inaccuracies caused by the voltage and temperature sensitivity of the Zener-diode. This setup offers accurate measurements of instant current values without affecting the device operation. [7]

6 Measurement target

This section describes the selection process of the measurement target. A Cortex-M3 based microcontroller was selected, and the architecture is described in more detail in order to understand what is going to be measured.

6.1 Selecting the measurement target

A sensible measurement target has to be selected in order to perform meaningful measurements. For this project, there are some restrictions that narrow down the possible candidates for our measurement platform:

- A modern low-energy or low-power ARM processor suitable for embedded systems like a sensor node such as described in section 2.2.
- A single-core processor as it is adequate for a first prototype of a measurement platform. This also means that it is possible to use a simple energy model at first.
- An ARM processor architecture that is supported by the WCC to aid the development towards a multi-criteria optimising compiler as envisioned in [35].

WCC supports compilation and code analysis for ARM processors. ARM processors are widely used in microcontrollers and in our survey we did not encounter microprocessors based on them. A primary difference between a microcontroller and microprocessor is that the microcontroller has an on-chip flash memory on which the program code is to be stored. This allows short startup times as the program does not reside on external memory as is the case with microprocessors. The downside is the limited memory size.

Another aspect is the power supply: a microcontroller usually has an embedded power supply to regulate its own internal voltage levels and thus needs only one voltage power rail. A microprocessor on the other hand needs to be supplied with different voltages for the different parts of the chip as it has no embedded regulation hardware inside the chip. [14]

Figure 12 shows different ARM architectures included in the survey of current low-energy processors for this project. The ARM Cortex-M processors are meant to be used in microcontrollers targeted at low-cost and low-energy embedded systems, whereas the Cortex-R series processors focus on high-performance real-time applications. The older versions of aiT used in WCC also support the classic ARM7TDMI processor architecture which can be considered to be the predecessor of the Cortex-M series. Even though the ARM7 processors were introduced in 1994, there are still

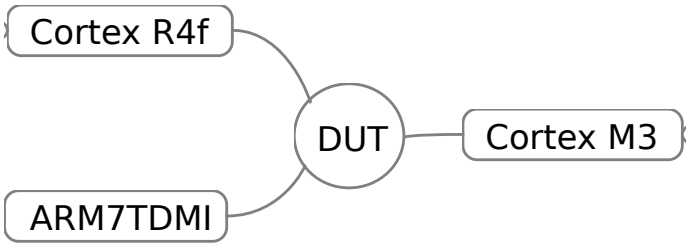


Figure 12: Three ARM architectures investigated.

microcontrollers based on this processor architecture on the market. It is included in this survey in case the more recent ones would have not been suitable for the measurement platform prototype. [3]

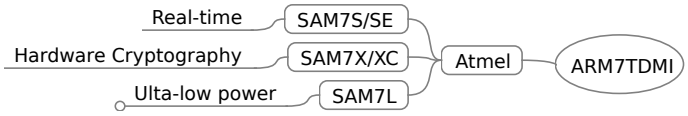


Figure 13: ARM7TDMI processors by Atmel.

The ARM7TDMI has been a widely used ARM core in many applications. Currently there are not many microcontrollers that use this core and it is being replaced by more current microprocessors. Atmel produces microcontroller series using this core, the SAM7S/SE, SAM7X/XC and SAM7L series, each of them targeted at different application uses.

The SAM7S/SE series is targeted at real-time applications while the SAM7X/XC is targeted at applications with special needs for hardware cryptography. The SAM7L is targeted at applications with very low power requirements featuring a 100 nA power-down mode.

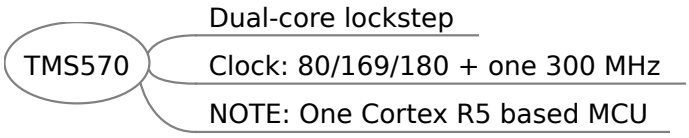


Figure 14: Cortex-R4 processors by Texas Instruments.

The ARM Cortex-R4 architecture is targeted at real-time systems. There are currently two microcontroller families from Texas Instruments based on the Cortex-R4 and Cortex-R4f. The 'f' in Cortex-R4f indicate that the ARM core has hardware floating point processing capabilities. Both microcontroller families belong to the Hercules Safety microcontroller family. The TMS570 family is targeted at transportation applications while the RMxxx families targeted at safety-critical medical and industrial applications.

Both the TMS570 and RM microcontroller families implement the Lock-step CPU Safety Architecture from Texas Instruments. The Lock-step CPU Safety Architec-

ture is based on the idea of having a duplicate microprocessor executing the same instructions one or more cycle after the other microprocessor. This physical and temporal differentiation allows the hardware with the help of discrepancies in outcomes to spot possible errors in execution. This feature makes the measurement of an instruction energy cost difficult as there are two cores executing a different instruction at the same time. It collides with our single-core requirement and is therefore not a suitable option for us.

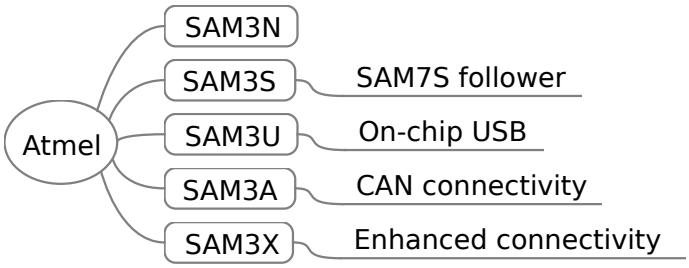


Figure 15: Cortex-M3 processors offered by Atmel.

Atmel has a range of MCU series based on the Cortex-M3 microprocessor. The SAM3N series is a basic model with the lowest clock speed and targeted at multiple applications. The SAM3S focuses on low power consumption and is intended to be the follower of the SAM7S series from the ARM7TDMI architecture. The SAM3U focuses on USB connectivity and SAM3A on CAN connectivity. SAM3X is meant for networking applications by providing Ethernet connectivity in addition to CAN and USB. The SAM3N and SAM3S would be suitable candidates for the prototype platform.

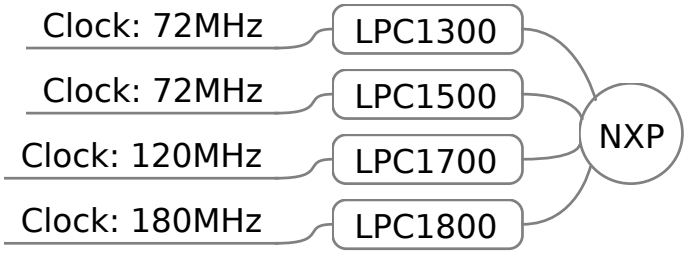


Figure 16: Cortex-M3 processors offered by NXP.

The NXP uses the Cortex-M3 in four of its microcontroller series: LPC1300, LPC1500, LPC1700 and LPC1800. The LPC1300 is the basic model offering mid-range performance and basic connectivity with each of the other series offering more performance and advanced connectivity and features. For example, the LPC1700 and LPC1800 have LCD-controllers. This is unnecessary hardware that could create unwanted noise, and therefore the two microcontrollers were not considered further to be an option. As a relatively simple processor is sufficient for the prototype, the LPC1300 would be a suitable choice.

Texas Instruments produces two MCU families which are based on the Cortex-M3: The F28M3x and TMS470M families. The F28M3x microcontroller is a dual-core

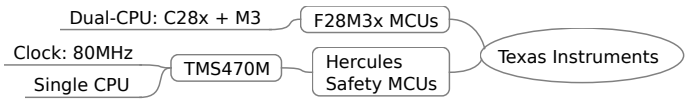


Figure 17: Cortex-M3 processors offered by Texas Instruments.

system with a C28x and Cortex-M3 integrated into a single chip. The C28x core is intended for the real-time and loop controller tasks while the Cortex-M3 works as the host subsystem taking control of communications, monitoring and other system functions. The dual-core structure collides with the requirement of a single-core processor.

The TMS470M series does not implement the lock-step CPU Safety Architecture mentioned before. Thus, it has only one Cortex-M3 core which fulfils the requirement of a single-core processor. As a bonus, the Flash memory module has its own supply pin. This could be used for measuring and modeling the energy consumption of the memory module separately from the core. The microcontroller is targeted at embedded systems used in transportation applications like Electric Power Steering (EPS) and safety related automotive applications.

The most likely candidates for the prototype were the NXP LPC1300, Atmel SAM3N and SAM3S, and the Texas Instruments TMS470M. All of them would suite the prototype well and fulfil the criteria listed at the beginning of this section: they all use a single-core Cortex-M3 processor supported by WCC and are suitable for being used in a sensor node. In the end, the TMS470MF03107 microcontroller from the TMS470M family was selected for its additional Flash memory supply voltage pin. However, the pin was not eventually exploited in the prototype as it would have complicated the hardware design.

6.2 ARM Cortex-M3

The ARM Cortex-M3 architecture is a modified Harvard architecture. The most distinctive feature of the Harvard architecture is that instructions and data are accessed via separate buses. The pure Harvard architecture also requires the instruction and data memories to reside in physically different memories but the Cortex-M3 mitigates this requirement which turns it to a modified Harvard architecture. It supports the Thumb and Thumb-2 Instruction Set Architecture (ISA).

Figure 18 provides a quick overview at the features provided by the ARM Cortex-M3 processor.

- Nested Vectored Interrupt Controller (NVIC) - an embedded interrupt controller that supports low-latency interrupt processing. It supports 240 interrupts with 256 priorities that can be changed dynamically.

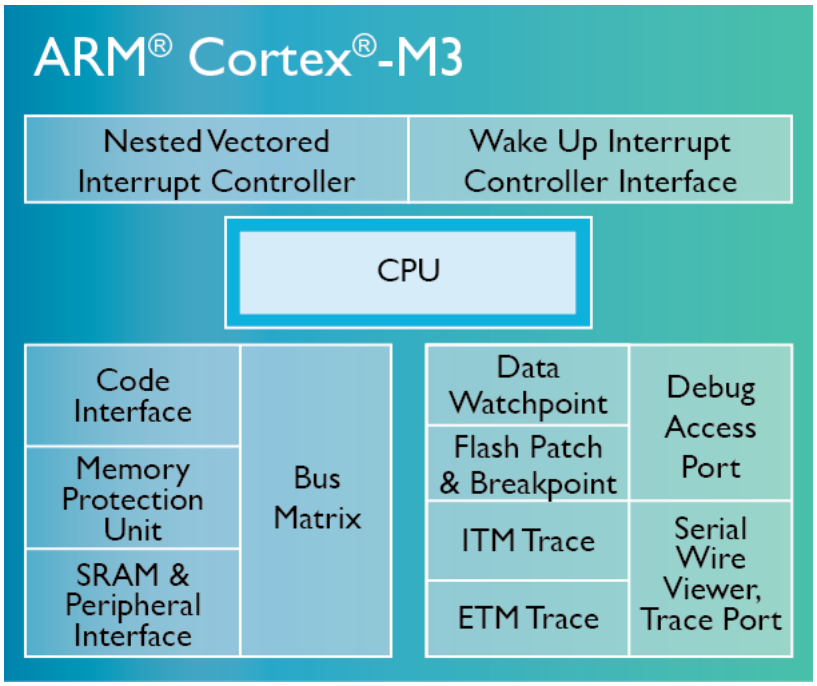


Figure 18: ARM Cortex-M3 chip architecture [2].

- CoreSight debug and trace interfaces which offer multiple choices of debug communication protocols (JTAG and Serial Wire Debug) and tracing capabilities.
- An optional Memory Protection Unit.
- Code interface which supports both Thumb and Thumb-2 Instruction Set Architectures (ISA)

The pipeline consists of three stages: Instruction Fetch, Instruction Decode and Instruction Execute. The decode stage also executes a speculative instruction fetch if a branch instruction is encountered.

6.3 TMS470MF03107 microcontroller

The Hercules TMS470M microcontroller series from Texas Instruments is targeted at safety-related automotive applications. The microcontroller is based on the ARM Cortex-M3 CPU running at 80MHz. The chip has built-in safety features like CPU and RAM self test engines, Error Correcting Code (ECC) and parity checks.

The upper half of the block diagram in figure 19 shows the microcontroller system, debug and power supply modules which have to be operational to allow basic operations on the chip. The core part is the ARM Cortex-M3 processor with a 'logic

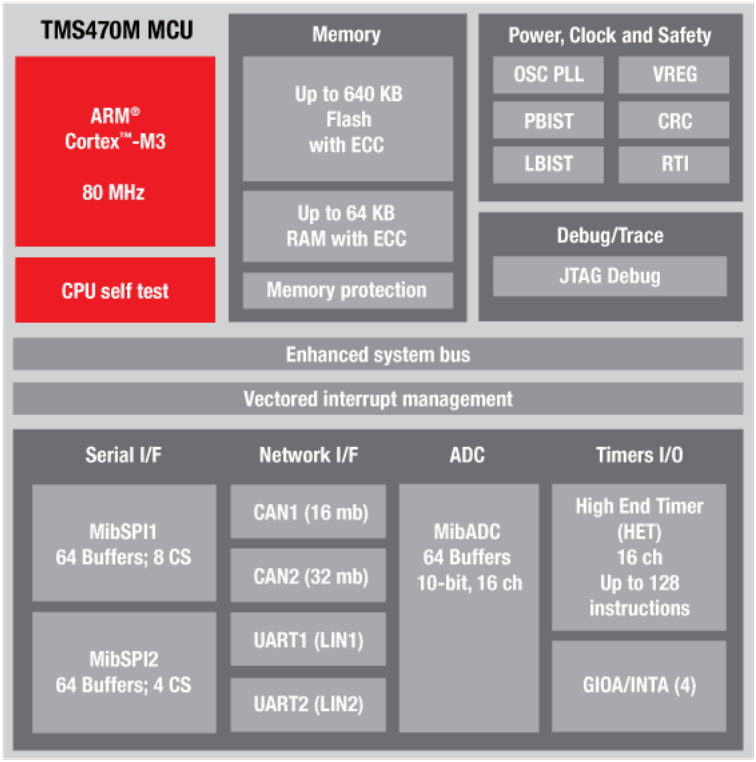


Figure 19: TMS470M block diagram [41].

built-in self-test' (LBIST) module. It allows the processor to verify the correct operation without external test equipment.

The microcontroller has two types of memory: SRAM and Flash. Both memory types have ECC capabilities to detect and correct data corruption. There is also an additional Memory Protection Unit to prevent access to protected parts of the memory.

The Power, Clock and Safety -block includes the clock logic, internal power regulation and debug functionality. The debug interface is the industry standard IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture (JTAG). [21]

The middle-part consists of the microcontroller internal structures like the system bus and interrupt management system. The lower part of the diagram shows the different interfaces to the outside. There are serial, network (CAN and UART) and timer interfaces. These are of little interest in this project as they are not going to be used extensively.

7 Energyboard construction and testing

The Energyboard is a printed circuit board (PCB) built to create a prototype of a measurement system. The board layout was designed using a PCB design software called EAGLE. [1] The Energyboard was manufactured at the faculty for Printed Circuit Board Technology of Ulm University. In total two Energyboards have been manufactured, assembled and tested.

This section describes the design and functional parts of the Energyboard. Various people at Ulm University offered their insight and improvement suggestions during the design process which influenced the overall design of the Energyboard.

7.1 General PCB design considerations

The physical aspects and features have also to be considered when constructing a piece of hardware. When measuring the microcontroller energy consumption, one consideration point is the PCB board itself. The main concern in this project was the transmission lines – or more commonly known as traces – and the various sources of unwanted interference that could create noise and impact the measurements. The aim is to minimize this noise so that it is possible to measure the small changes in the currents.

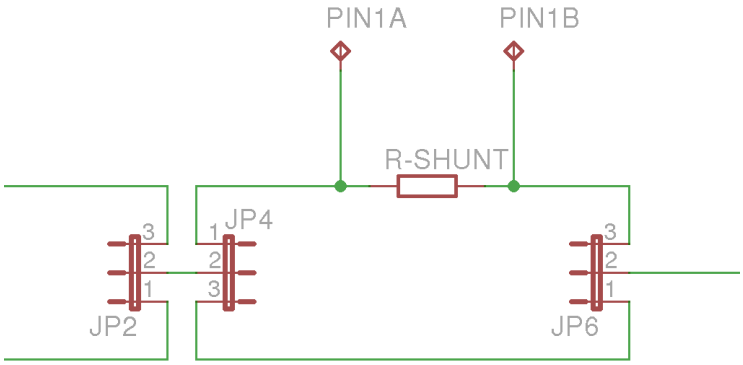


Figure 20: Parallel traces interfering with each other like antennas.

The measurements are going to be performed by connecting to the traces directly but one source of error is actually the traces themselves. The traces act as antennas – albeit very small – and pick up interference over the air or even from each other. One of the early board designs includes an example of a scenario where the antenna effect could take place as shown in figure 20. The traces in question are the ones between jumpers JP4 and JP6. In this case, the effect comes from the fact that only one of the traces would be connected to the circuit at a time. In this design stage,

the focus was to create a platform with adjustable configuration once it would have been built.

The upper route in figure 20 is the on-board energy measurement interface with the pins acting as points to which an oscilloscope connects. The lower route was to bypass the on-board measurement pins in case the power supply has a built-in current integrator for measuring the consumed current of the microcontroller. Bypassing the shunt resistor would minimize the influence on the measurements as discussed in section 5.1.1. Thus, only one of the routes would be connected to the circuit at a time. The route not connected to the rest of the circuit would be prone to behave like an antenna as there is no current to resist it.

In addition, the two signals from the left corner of figure 20 come from two different power sources: one on-board regulated and one externally regulated, like with a high-end laboratory power supply. The jumper labelled JP2 is for choosing either of these power sources as only one can be in use at a time.

A rule of thumb in PCB design is to avoid 90 degree corners as they may cause reflections of the signals moving along the traces. Also, the angles cannot be below 15 degrees because the etching agent, when manufacturing the PCB, may get trapped inside these pockets and cause corrosion [15]. On the Energyboard the thick main power rails have right angles whereas the thinner traces use 45 degree angles as seen in the layout image in appendix B. This is purely an aesthetic choice as the reflections get relevant in high-speed designs [30]. Frequencies of 1 GHz and above are considered to be high-speed and the microcontroller system clock is at 80 MHz or lower, so the reflections should cause no problems in the prototype.

7.2 Design of the Energyboard prototype

The Energyboard prototype shown in figure 21 is based on the measurement setup described in section 5.1.1. As figure 21 highlights, the Energyboard can be divided into five different functional blocks:

- Power supply.
- Voltage regulation.
- Measurement point.
- Device Under Test or DUT.
- JTAG connector.

The schematic and layout figures in this subsection are details from the full schematic and layout found in appendix A and appendix B, respectively.

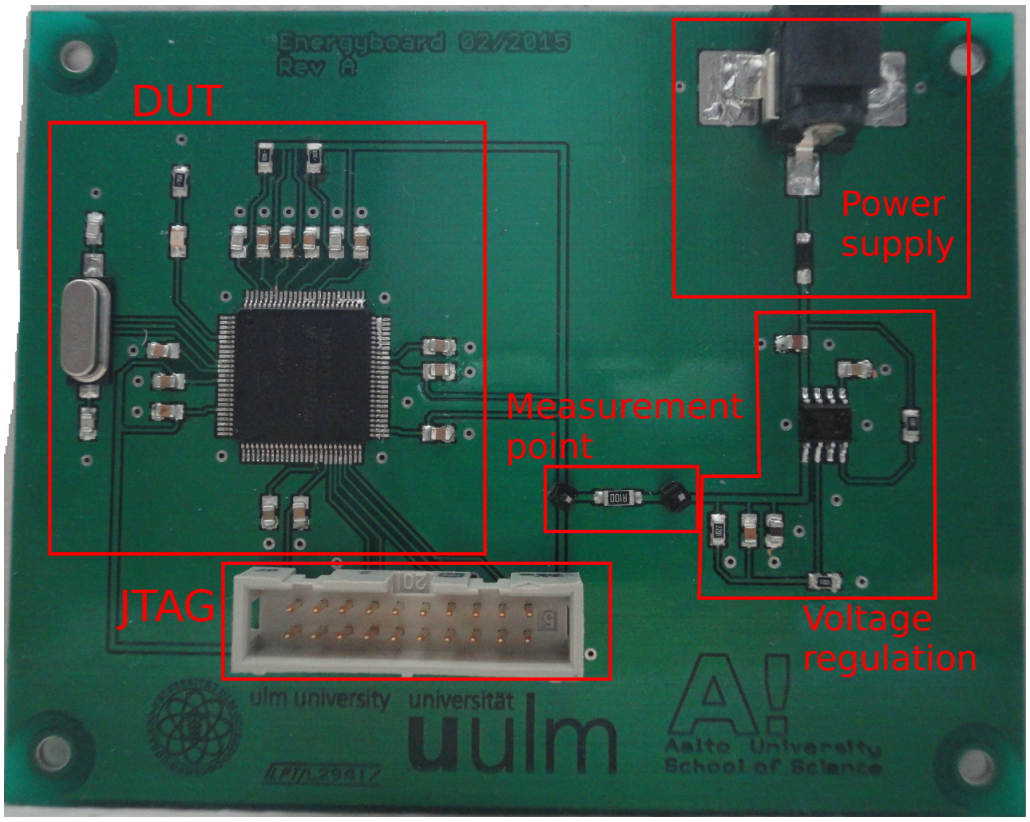
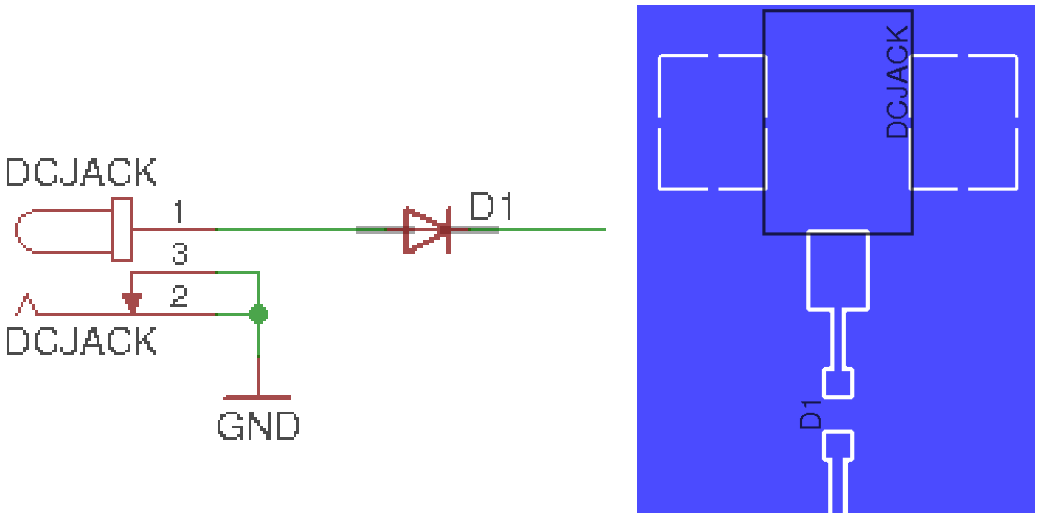


Figure 21: Overview of the Energyboard's functional blocks.



(a) Schematic (b) Layout

Figure 22: Power supply schema and layout.

Power supply A direct current (DC) connector to connect a suitable power source to the board. For programming and debugging a standard laboratory power supply would be enough but for making energy measurements a higher quality power supply may be needed. A Schottky-diode is placed right after the DC connector to provide

reverse polarity protection for the circuit.

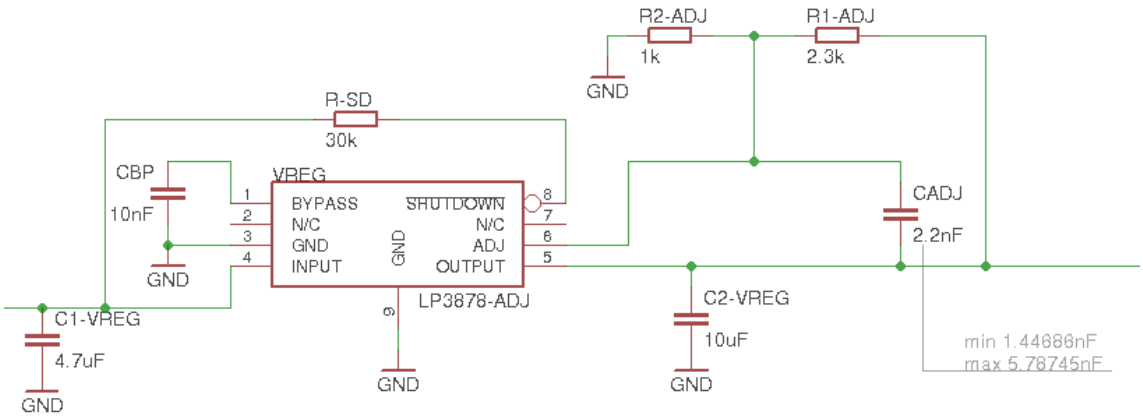


Figure 23: Voltage regulator schematic.

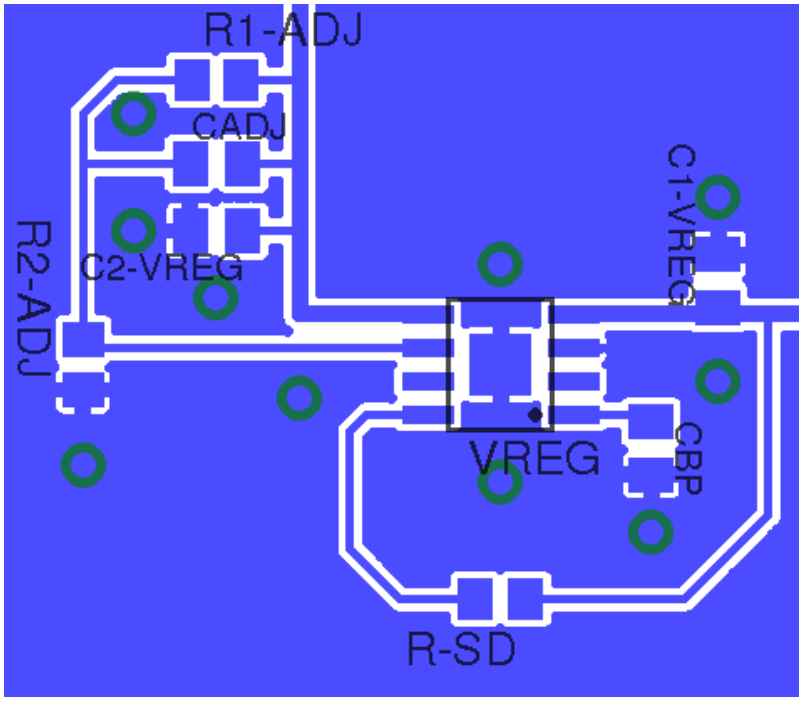


Figure 24: Voltage regulator layout.

The **voltage regulator** integrated circuit (IC) used is the LP3878-ADJ from Texas Instruments. The design of the voltage regulation circuit on the Energyboard follows closely the circuit presented in the application note of the LP3878-ADJ datasheet [40]. The schematic and layout are shown in figures 23 and 24, respectively. The voltage regulator is adjusted to the desired output voltage V_{OUT} according to the

following equation:

$$V_{OUT} = V_{ADJ} \times (1 + (\frac{R_1}{R_2})) \tag{11}$$

The adjust pin voltage can be assumed to be $V_{ADJ} = 1V$. The adjust resistors for the Energyboard are set to $R_1 = 2.3k\Omega$ and $R_2 = 1k\Omega$ which means that the output voltage results in $V_{OUT} = 3.2V$. The adjust resistors are labelled as R1-ADJ and R2-ADJ in the schematic and layout files. This is well within the supply voltage range of [3.0V, 3.6V] required by the microcontroller. The LP3878-ADJ can output a current up to $800\,mA$ which should be more than enough as the microcontroller total supply current $I_{CCTOTAL}$ is estimated to be $125\,mA$ at maximum. [41]

NAME	VALUE	FUNCTION
C1-VREG	$4.7\,\mu F$	Input
C2-VREG	$10\,\mu F$	Output
CBP	$10nF$	Noise bypass
CADJ	$2.2nF$	Feedforwardi

Table 1: Voltage regulator external capacitors.

The voltage regulator also needs external capacitors for stability. The capacitors, along with their values and functions within the voltage regulator circuit, are listed in table 1. They are ceramic X5R or X7R capacitor as recommended in the datasheet [40].

The bypass capacitor reduces significantly the output noise and is required for loop stability. The feedforward capacitor increases the phase margin which improves the transient response and settling time according to changes in load or input voltage. It also improves the stability of the IC.

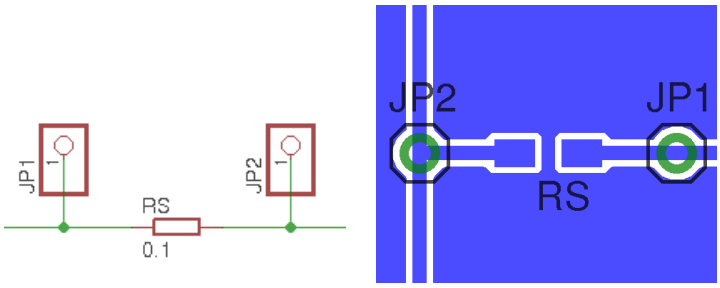


Figure 25: Measurement point schema and layout.

The **measurement point** consists of a good quality shunt resistor with two test pins on either side of it. A shunt resistor is a resistor with a small resistance for minimising the influence on the circuit. For measuring the varying voltage during the microcontroller operation, a differential probe would be optimal in this situation

as the ground is relative. Different kinds of probes are described in section 4.1. An amplifier could be connected to the measurement pins for amplifying the oscillating voltage. The operation amplifier described in section 4.2 is intended for exactly this purpose.

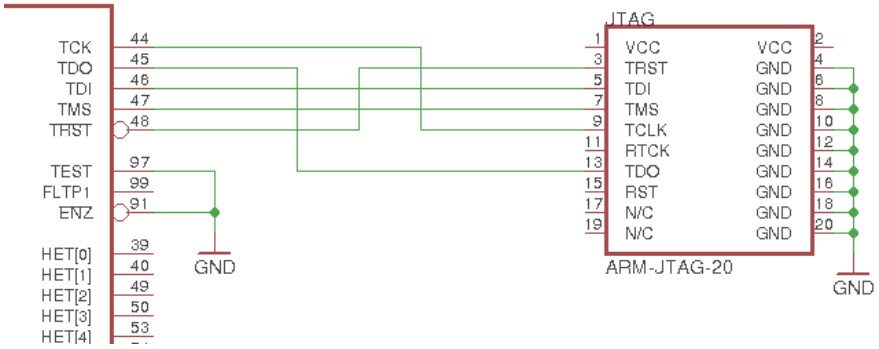


Figure 26: JTAG schematic.

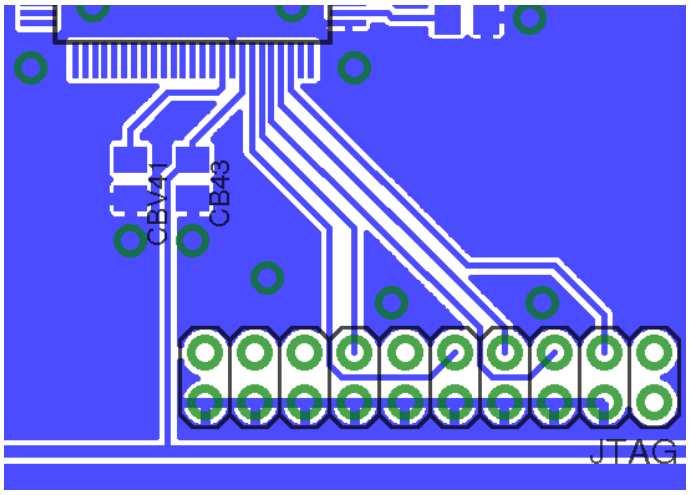


Figure 27: JTAG layout.

The **JTAG connector** shown in figures 26 and 27, is meant for programming and debugging the microcontroller. To program a.k.a. to flash the microcontroller using a computer needs a JTAG programmer in-between. The JTAG programmer is connected to the computer usually through USB and to the microcontroller through a JTAG cable connected to the connector on the PCB. The JTAG programmer is controlled with special software like OpenOCD [34].

The circuitry associated with the DUT consists mainly of the power lines, essential reset and power supply signals to make the chip functional. Most of the logic pins like general purpose, CAN and timer pins are unused and left floating on the Energyboard with the exception of the ADIN pins. The power pins (VCCIOR) are used for supplying the chip with current as shown in figure 28. The VCCP

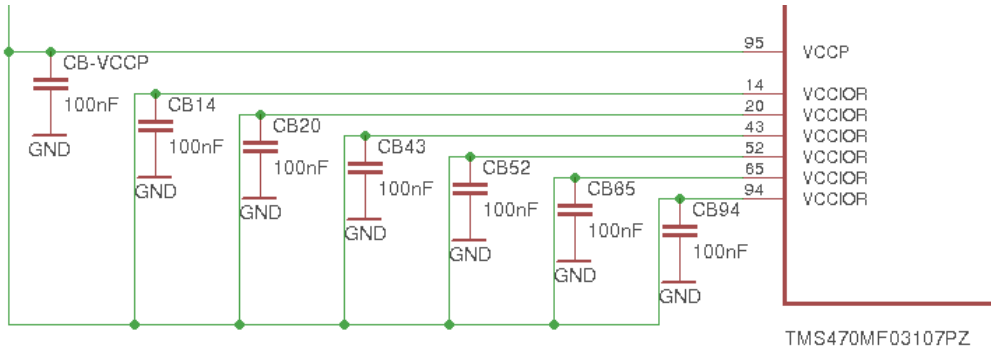


Figure 28: Decoupling capacitors for stabilising the supply voltage.

pin supplies current to the Flash memory of the microcontroller. The part of the layout related to the microcontroller can be seen clearly in the Energyboard layout in appendix B.

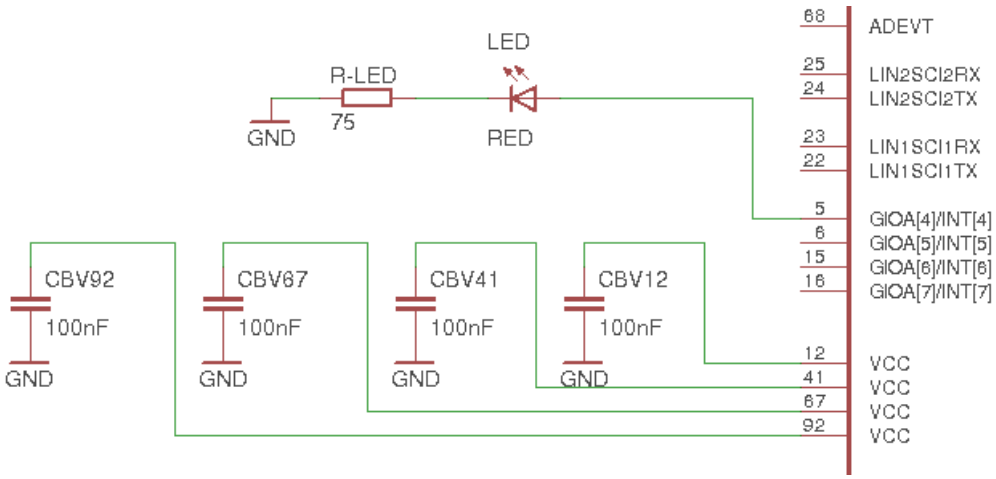


Figure 29: LED and decoupling capacitors for stablising the internal supply voltage.

There are four VCC pins at $1.55\,V$ voltage level which are pulled to the ground as shown in figure 29. The $1.55\,V$ is the supply voltage for Cortex-M3 microprocessor but the required voltage adjustment is done in the microcontroller internal power supply so that there is no need to provide it externally. Pulling them to the ground through capacitors serves for stabilising the voltage and filtering out current peaks. Also shown in figure 29 is the LED circuitry. The LED is intended for quickly producing output for debugging purposes.

7.3 Manufacture and assembly

The board was manufactured using non-plated copper technology with a solder mask. The board is not RoHS compliant as it uses tin/lead solder – a soft solder – to finish the surface and to protect the exposed copper pads until the board is

assembled. In context of this thesis, assembling means soldering IC-components onto the board. The manufacturing process of a two-sided PCB consists roughly of the following procedures: [5, 9]

- Copper plating
- Patterning and etching
- Drilling
- Solder masking and plating

The process starts with covering both sides of the PCB substrate entirely with a layer of copper which creates a blank PCB with no tracks (also called traces). The blank PCB is coated with photosensitive coating and exposed to ultraviolet light masked with the wanted track pattern. The mask or film is created with a photoplotter. The exposure to UV-light removes the etching-resistant coating from the areas exposed by the mask. The UV-processed PCB is then etched for example by submerging it into a tank filled with suitable etching solvent which removes the unwanted copper from the PCB. This procedure is called photoengraving but there are other methods too. The PCB is then washed and dried.

After this, the drillings for vias and such are made. A via creates an electrical connection between different layers of a PCB. The drillings are done by an automated drilling machine which is controlled with numerically controlled drill (NCD) files. After that, the vias are copper plated to connect the two layers.

In order to finish the PCB, a soldering mask is applied to protect the copper from oxidation and help during the assembly stage when the components are soldered onto the board for preventing the formation of solder bridges. Solder plating on the other hand refers to plating exposed copper areas, such as pads and landing areas, with solder. This protects the copper and keeps the area solderable when assembling the PCB.

The components of the Energyboard were soldered by hand using techniques described for example in [5]. This is a slow procedure compared to flow soldering that is used in industrial production where the pads are coated with solder. The surface-mount devices (SMD) are pressed onto the pads and the PCB or solder is heated to make the solder "flowing". When the solder cools down again the joints of the SMD-components and pads are linked together. The hand soldering technique is pretty similar but instead of heating the solder on all pads at the same time, the soldering iron is used for heating the pads one at a time. Usually the preapplied solder on the pads is not enough and some extra solder has to be applied at the same time.

7.4 Testing

After the two Energyboards were assembled and soldered, the boards were tested using a Fluke 175 True RMS multimeter. Voltage measurements were performed on various exposed microcontroller pins. It is a crude test but already reveals a problem on both Energyboards as the voltage on the microcontroller RESET pin is pulled low. It should be pulled high when in normal operation mode.

PIN	Board A	Board B
GIOA[4]/INT[4]	$0.445\,V$	$0.495\,V$
VCC 12	$1.556\,V$	$1.591\,V$
VCC 41	$1.556\,V$	$1.592\,V$
VCC 67	$1.555\,V$	$1.592\,V$
VCC 92	$1.556\,V$	$1.591\,V$
VCCIOR 14	$3.192\,V$	$3.204\,V$
VCCIOR 20	$3.192\,V$	$3.204\,V$
VCCIOR 43	$3.192\,V$	$3.204\,V$
VCCIOR 52	$3.192\,V$	$3.204\,V$
VCCIOR 65	$3.192\,V$	$3.204\,V$
VCCIOR 94	$3.192\,V$	$3.200\,V$
VCCP	$3.192\,V$	$3.205\,V$
VCCAD	$3.192\,V$	$3.205\,V$
RESET	$0.237\,V$	$0.234\,V$
PORRST	$1.078\,V$	$1.083\,V$
ENZ	$0.0\,V$	$0.0\,V$
TEST	$0.0\,V$	$0.0\,V$
VREG-OUTPUT	$3.193\,V$	$3.206\,V$

Table 2: Pin voltage measurements.

Table 2 shows the voltage measurement on a selection of pins. As there were two Energyboards available for testing one is called 'Board A' and the other one 'Board B'. All, except the VREG-OUTPUT pin, are pins of the TMS470M microcontroller. The VREG-OUTPUT is the output pin of the voltage regulator. All other voltages, except the RESET pin, show values that are within expected ranges. The $0.2\,V$ value shows that the RESET pin is pulled low by the microcontroller. This would indicate that the microcontroller is in reset state and cannot start properly.

There was a suspicion that the supply voltage is not stable enough and causes some problems, like instability, inside the microcontroller. However, accurate measurements with an oscilloscope did not confirm this suspicion. The measurement setup consisted of an USB oscilloscope, the PicoScope 3204, which was connected to a laptop running the PicoScope Oscilloscope Software 6.10.16 [37]. The sample rate was set to $50\,MS/s$ and the sample interval to $20\,ns$. The measurements were conducted with a passive 1X attenuator probe The time window in figures 30 and 31 is

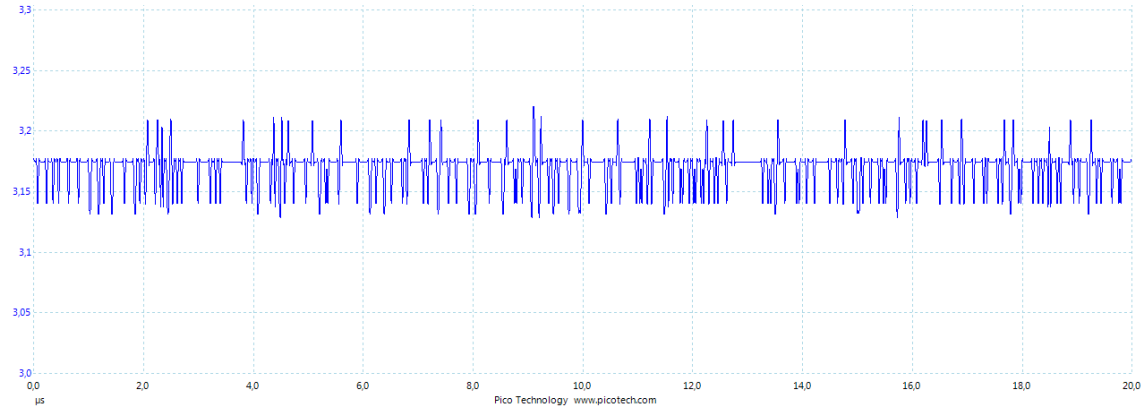


Figure 30: Regulated supply voltage of Board A.

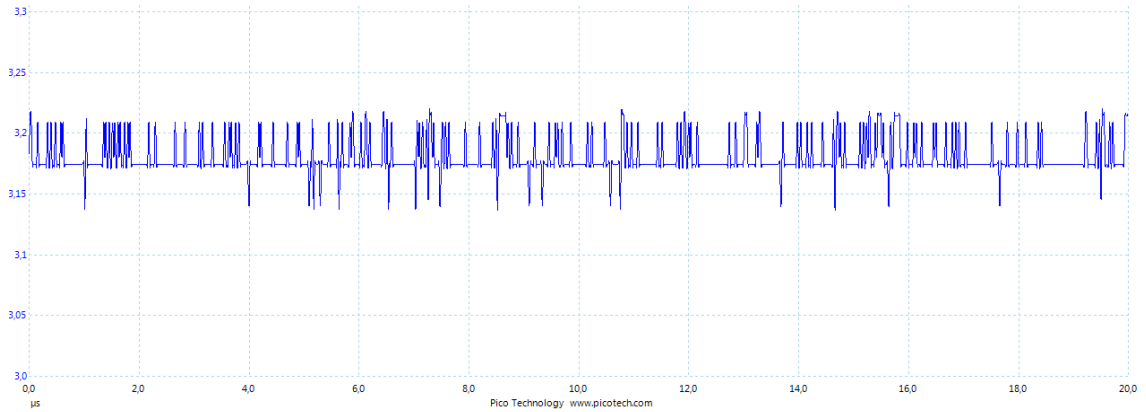


Figure 31: Regulated supply voltage of Board B.

$20\,\mu s$.

The oscilloscope traces shown in the figures 30 and 31 present the measurement results. They X-axis show a time-span of $20\,\mu s$. The regulated supply voltage of Board A varies between $[3.168\,V, 3.170\,V]$ while the average is at $3.169\,V$. For Board B, the voltage varies between $[3.179\,V, 3.182\,V]$ with an average of $3.180\,V$. The variation is minimal so it is unlikely that voltage stability would be the root cause of the bug.

Another suspicion was, that the $100\,pF$ capacitors in connection with the quartz crystal are too big. Much smaller capacitors, below $22\,pF$, should be tested. It turned out, that the crystal on Board A is not oscillating properly which could definitely be the cause of the problem. The crystal on Board B worked properly though, which means that the ultimate problem lies somewhere else.

In the end, the exact cause for the bug still remains undiscovered when writing this text. More testing and debugging is needed to discover the root cause of the problem.

8 Future work and summary

8.1 Future work

No energy measurements could be conducted as there is a bug in the Energyboard design. The bug could not be identified during the project time-frame. However, once the Energyboard is debugged, microbenchmarks can be executed and measurements conducted. Some energy measurement system setups, like oscilloscopes and probes, are described in section 4. The next step would be to create energy models and then benchmark the models using application benchmarks.

Looking further into the future, making energy optimisation of software practical for application developers would require automating the process. A practical solution would be to implement automated energy optimisation into a compiler. There is an interest in implementing energy optimisations in the WCET aware C-compiler WCC presented briefly in section 2.6. Its architecture is designed to be extensible for adding other optimisation goals.

A interesting continuation of this work would be to make a comparison between measurement platforms based on the four circuits described in this thesis. The automated measurement circuit is already being looked into at Ulm University. The other two circuits would still need to be constructed. Also, a comparison between traditional probes and wireless methods, as used in power attacks, could be a possibility.

8.2 Summary

This thesis explored the topic of designing and constructing a measurement platform for instruction-level energy models. The description of the prototype can serve as a guide for future projects of similar nature as the challenges are likely to remain the same. A prototype platform was built and is being developed further.

This thesis investigated four measurement circuits. The circuits are designed to minimise noise by leaving out hardware unnecessary for running microbenchmarks and conducting energy measurements. The classic shunt resistor circuit was chosen as the basis for the Energyboard prototype. Another measurement platform, based on the automated measurement circuit, is being developed further at Ulm University. A survey concerning low-energy processors was also conducted to find suitable measurement targets. A low-energy Cortex-M3 based microcontroller targeted at automotive embedded systems was selected for this thesis project.

References

[1] EAGLE. URL http://www.cadsoftusa.com/. PCB Design Software by CadSoft.

[2] ARM Holdings plc. Cortex-m3 processor, . URL http://www.arm.com/products/processors/cortex-m/cortex-m3.php. Referenced 2014-12-19.

[3] ARM Holdings plc. Processors, . URL http://www.arm.com/products/processors/. Referenced 2014-12-17.

[4] *ARM7TDMI-S: Technical Reference Manual*. ARM Limited, r4p3 edition, 2001.

[5] Janet (Jan) Louise Axelson. *Making Printed Circuit Boards*. TAB Books, first edition, 1993. ISBN 0-8306-3951-9.

[6] Mostafa Bazzaz, Mohammad Salehi, and Alireza Ejlali. An accurate instruction-level energy estimation model and tool for embedded systems. *IEEE Transactions on Instrumentation and Measurement*, 62(7):1927–1934, July 2013. ISSN 0018-9456. doi: 10.1109/TIM.2013.2248288. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6478810.

[7] A. Borovyi, V. Kochan, Z. Dombrovskyy, V. Turchenko, and A. Sachenko. Device for measuring instant current values of CPU's energy consumption. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2009. IDAACS 2009. IEEE International Workshop on*, pages 126–130, September 2009. doi: 10.1109/IDAACS.2009.5343010.

[8] L. Coetzee and J. Eksteen. The internet of things - promise for the future? an introduction. In *IST-Africa Conference Proceedings, 2011*, pages 1–9, May 2011.

[9] Eurocircuits. Making a pcb - pcb manufacture step by step. URL http://www.eurocircuits.com/pcb-prototype-and-small-batch-services/making-a-pcb-pcb-manufacture-step-by-step. Referenced 2015-04-02.

[10] Eurostat. Information society statistics - households and individuals. Internet, April 2014. URL http://ec.europa.eu/eurostat/statistics-explained/index.php/Information_society_statistics_-_households_and_individuals. Referenced 2015-04-21.

[11] H. Falk, P. Lokuciejewski, and H. Theiling. Design of a wcet-aware c compiler. In *Embedded Systems for Real Time Multimedia, Proceedings of the 2006 IEEE/ACM/IFIP Workshop on*, pages 121–126, October 2006. doi: 10.1109/ESTMED.2006.321284.

[12] Heiko Falk and Paul Lokuciejewski. A compiler framework for the reduction of worst-case execution times. *Real-Time Systems*, 46(2):251–300, July 2010. ISSN 0922-6443. doi: 10.1007/s11241-010-9101-x.

[13] Jason E. Fritts, Frederick W. Steiling, Joseph A. Tucek, and Wayne Wolf. Mediabench II video: Expediting the next generation of video systems research. *Microprocessors and Microsystems*, 33(4):301 – 318, 2009. ISSN 0141-9331. doi: http://dx.doi.org/10.1016/j.micpro.2009.02.010. URL http://www.sciencedirect.com/science/article/pii/S014193310900026X. Media and Stream Processing.

[14] Frédéric Gaillard and Andreas Eieland. Microprocessor (MPU) or microcontroller (MCU)? what factors should you consider when selecting the right processing device for your next design. Technical report, Atmel Corporation, 2013. URL http://www.atmel.com/images/mcu_vs_mpu_article.pdf.

[15] Charles Hamilton. *A Guide to Printed Circuit Board Design*. Butterworths, 1984. ISBN 0-408-01398-2.

[16] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies ltd., and Toshiba Corporation. *Advanced Configuration and Power Interface Specification Revision 5.0 Errata A*, 2013.

[17] Alan Holt and Chi-Yu Huang. *Embedded Operating Systems - A Practical Approach*. Undergraduate Topics in Computer Science. Springer, 2014. ISBN 978-1-4471-6603-0. doi: 10.1007/978-1-4471-6603-0.

[18] INFSO D.4 Networked enterprise and RFID INFSO G.2 Micro and nanosystems in Co-operation with RFID working group of EPOSS. Internet of things in 2020, roadmap for the future, September 2008.

[19] International Telecommunications Union. ITU internet reports 2005: The internet of things, 2005.

[20] Margarida F. Jacome and Anand Ramachandran. Power aware embedded computing. In Richard Zurawski, editor, *Embedded Systems: Handbook*. CRC Press, 2006. ISBN 0-8493-2824-1.

[21] Randy Johnson and Stewart Christie. JTAG 101: IEEE 1149.x and software debug. white paper, Intel, January 2009.

[22] Nikolaos Kavvadias, Periklis Neofotistos, Spiridon Nikolaidis, C A Kosmatopoulos, and Theodore Laopoulos. Measurements analysis of the software-related power consumption in microprocessors. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, 53(4):1106–1112, 2004. doi: 10.1109/TIM.2004.830784.

[23] George Kyriazis. Heterogeneous system architecture: A technical review. Technical report, AMD, 2012. URL http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-system-architecture-hsa/.

[24] Theodore Laopoulos, , Periklis Neofotistos, C A Kosmatopoulos, and Spiridon Nikolaidis. Measurement of current variations for the estimation of software-related power consumption. *Instrumentation and Measurement, IEEE Transactions on*, 52(4):1206–1212, 2003. doi: 10.1109/TIM.2003.816837.

[25] Luciano Lavagno and Richard Zurawski. Embedded systems: Toward networking of embedded systems. In Richard Zurawski, editor, *Embedded Systems: Handbook*. CRC Press, 2006. ISBN 0-8493-2824-1.

[26] Dejan Lazich. Embedded security, December 2014. Lecture at Ulm University on 2014-12-11.

[27] Chunho Lee, M. Potkonjak, and W.H. Mangione-Smith. Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. In *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*, pages 330–335, December 1997. doi: 10.1109/MICRO.1997.645830.

[28] Adrian McEwen and Hakim Cassimally. *Designing the internet of things*. John Wiley & Sons, 2013. ISBN 9781118430620.

[29] MediaBench Consortium. Mediabench ii benchmark. URL http://euler.slu.edu/~fritts/mediabench/. Referenced 2015-04-15.

[30] M.I. Montrose. Time and frequency domain analysis for right angle corners on printed circuit board traces. In *Electromagnetic Compatibility, 1998. 1998 IEEE International Symposium on*, volume 1, pages 551–556vol.1, August 1998. doi: 10.1109/ISEMC.1998.750154.

[31] Ravi Musunuri, Shashidhar Gandham, and Maulin D. Patel. Issues and solutions in wireless sensor networks. In Richard Zurawski, editor, *Embedded Systems: Handbook*. CRC Press, 2006. ISBN 0-8493-2824-1.

[32] S. Nikolaidis and T. Laopoulos. Instruction-level power consumption estimation embedded processors low-power applications. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, International Workshop on, 2001.*, pages 139–142, 2001. doi: 10.1109/IDAACS.2001.941998.

[33] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann, fourth edition, 2009. ISBN 978-0-12-374493.

[34] Dominic Rath. OpenOCD. URL www.openocd.org. Referenced 2015-03-09.

[35] Nicolas Roeser, Arno Luppold, and Heiko Falk. Multi-criteria optimization of hard real-time systems. In *Proceedings of the 8th Junior Researcher Workshop on Real-Time Computing (JRWRTC 2014)*, pages 49–52, 2014.

[36] Findlay Shearer. *Power Management in Mobile Devices*. Elsevier, 2008. ISBN 978-0-7506-7958-9.

[37] Pico Technology. URL www.picotech.com. Referenced 2015-05-18.

[38] Tektronix. Probe fundamentals, October 2009. URL http://circuitslab.case.edu/manuals.html.

[39] Tektronix. XYZs of oscilloscopes, 2011. URL http://info.tek.com/www-xyzs-of-oscilloscopes-primer.html.

[40] *LP3878-ADJ Datasheet*. Texas Instruments, May 2005. Revised February 2015.

[41] *TMS470MF04207/TMS470MF03107 16/32-Bit RISC Flash Microcontroller*. Texas Instruments, January 2012.

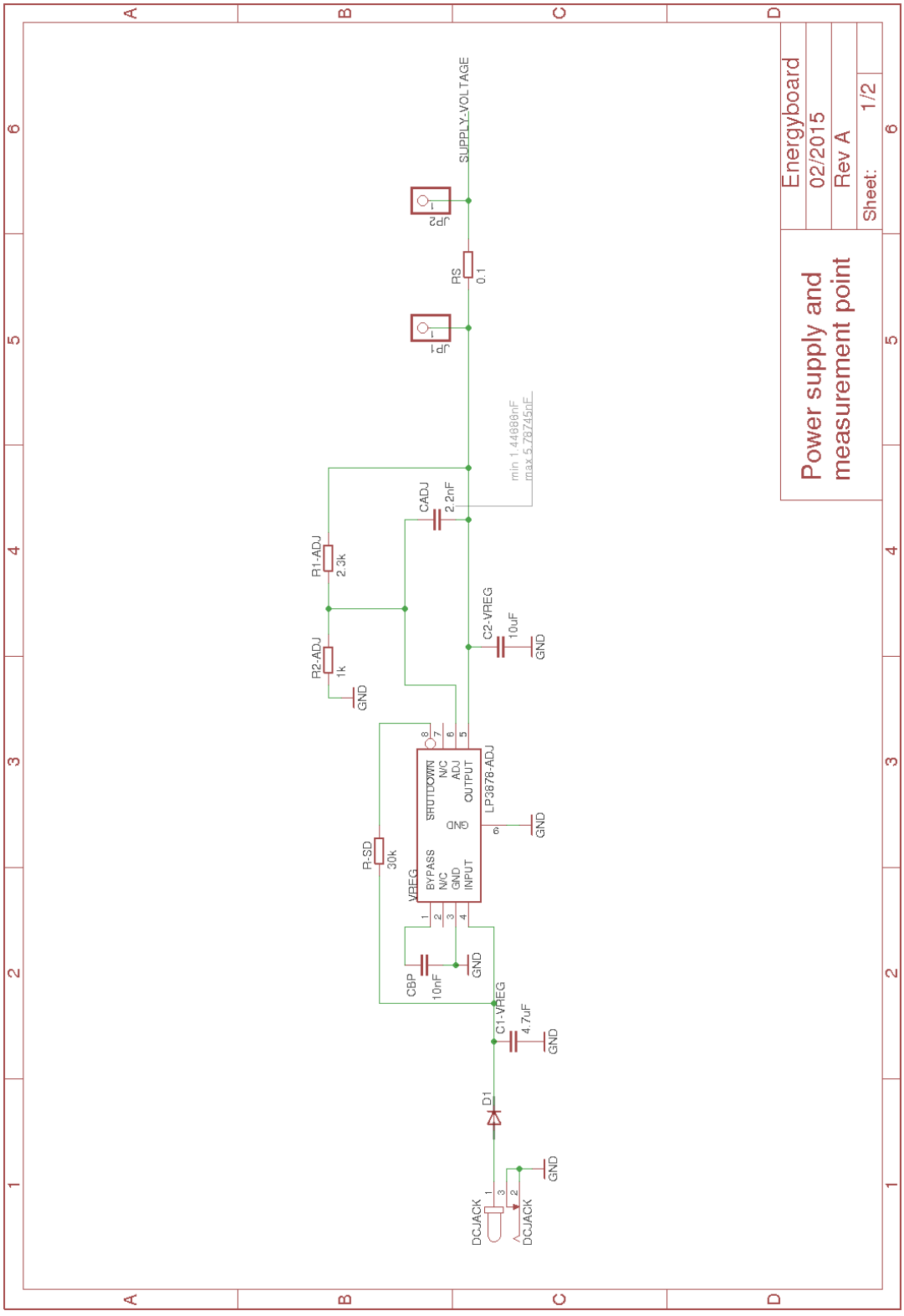
[42] Ulrich Tietze, Christoph Schenk, and Eberhard Gamm. *Halbleiter-Schaltungstechnik*. Springer Vieweg, 14. edition, 2012. ISBN 978-3-642-31025-6.

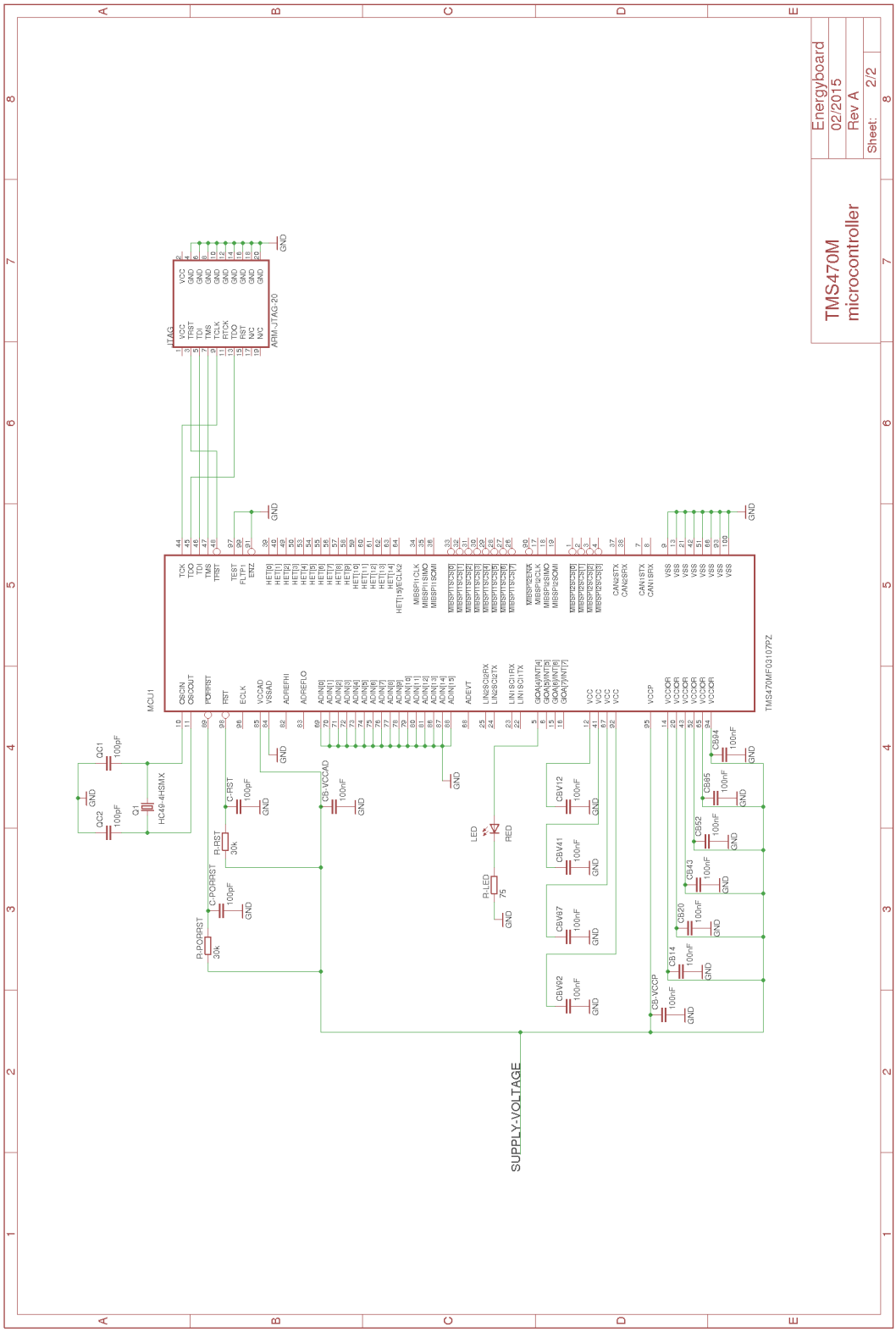
[43] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: a first step towards software power minimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):437–445, December 1994. ISSN 1063-8210. doi: 10.1109/92.335012. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=335012.

[44] Vivek Tiwari, Sharad Malik, Andrew Wolfe, and Mike Tien-chien Lee. Instruction level power analysis and optimization of software. In *VLSI Design, 1996. Proceedings., Ninth International Conference on*, pages 326–328, Bangalore, 1996. IEEE. doi: 10.1109/ICVD.1996.489624.

[45] Richard Wolfson. *Essential University Physics*, volume 1&2. Pearson Addison Wesley, first edition, 2007. ISBN 0-321-43564-8, 0-321-43565-6.

A Energyboard schematics




Energyboard
02/2015
Rev A
Sheet: 2/2
TMS470M
microcontroller
JTAG
ARM-JTAG-20
MCU1
TMS470MF03107PZ
SUPPLY-VOLTAGE
C-RST
R-RST
C-PORRST
R-PORRST
C-VCCAD
C-VCCP
LED
R-LED
RED
HC49-4HSMX
GND

B Energyboard PCB layout