# A Flexible Design Space Exploration Platform for Wireless Sensor Networks

zur Erlangung des akademischen Grades eines

## Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
am Karlsruher Institut für Technologie (KIT)

**genehmigte**

## Dissertation

von

**Dominic Hillenbrand**

aus Heidelberg

Tag der mündlichen Prüfung: 19.01.2010

Erster Gutachter: Prof. Dr.-Ing. Jörg Henkel

Zweiter Gutachter: Prof. Dr.-Ing. Klaus D. Müller-Glaser

Dominic Hillenbrand
An der Bleiche 32
67319, Wattenheim

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen - die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Karlsruhe, den 5.2.2010

Dominic Hillenbrand

# Acknowledgement

I have had a great time at the Karlsruhe Institute of Technology. My PhD thesis is one of the memorable documents of this remarkable time. Writing it has been a special experience by itself since it required recapitulating my time as a PhD student from the beginning to the end.

All this would not have happened if my advisor Professor Jörg Henkel had not entrusted me with a challenging position in the "Graduiertenkolleg" 1194 (an elite research group for forthcoming scientists funded by the German research foundation DFG). For this opportunity and his continuous support at all times I am deeply indebted to him.

Professor Klaus D. Müller-Glaser has been so kind to be my second advisor. For this honor I am very grateful.

The German research foundation DFG I would like to thank for funding my PhD position and a 3 month research visit to the University of California, in Los Angeles.

Special thanks go to my students who chose to contribute to my research projects. They have been very committed and motivated. Leading them through the projects and seeing them grow has been an invaluable experience. I wish them all the success for their future.

I would like to thank Wolfgang Rihm for his extremely accurate work. Without him it would have been much more difficult and time consuming to create hardware prototypes.

Eva Kwee-Christoph has been the good fairy of the group. She has always been helpful and supported my work.

It has been reassuring to have our secretary Mrs. Murr-Grobe take care of the multitude of organizational tasks. She knows the ins and outs of our department extremely well. Her advice has been invaluable.

I also do remember and appreciate the help of those who are not mentioned here.

Last but least I would like to dedicate my thesis to my wonderful parents who have supported me throughout my years in academia.

# List of Publications

**Conferences**

- D. Hillenbrand, J. Henkel: *"Block cache for embedded systems"* in the Proceedings of the 2008 Asia and South Pacific Design Automation Conference (**ASP-DAC'08**), IEEE Computer Society Press, Pages 322–327, Seoul, Korea, January 2008
- T. Jie, D. Hillenbrand, M. Holger: *"Instruction Hints for Super Efficient Data Caches"* in the Proceedings of the 9th International Conference on Computational Science (**ICCS'09**), Springer-Verlag, Pages 677–685, Baton Rouge, LA, USA, May 2009

**Exhibitions**

- D. Hillenbrand, M. Mende, T. Armstrong, J. Henkel: "*Hyperion: A sensor node test bed for (high-speed) power measurements*" - University Booth at the 2007 Design, Automation & Test in Europe (**DATE'07**), Nice, France, April 2007

**Other Presentations**

- D. Hillenbrand: "*Exploiting heterogeneous (die-stacked) memories in future CMPs to reduce power consumption*" in the Fifth International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems (**ACACES'09**), Terrassa (Barcelona), Pages 155–158, Spain, July 2009
- D. Hillenbrand: "*Hyperion – A flexible Sensor Node Testbed*" - Workshop der Graduiertenkollegs - Proceedings des gemeinsamen Workshops der Graduiertenkollegs 2008, Schloss Dagstuhl, Page 99, Germany, May 2008
- D. Hillenbrand: "*Design and evaluation of a hardware architecture for future sensor nodes*" - Workshop des Graduiertenkollegs, Schloss Dagstuhl, Germany, October 2006

# Abstract

Wireless sensor networks are embedded systems which are distributed in their environment where they cooperate to achieve a common goal. A dense sensor network has the advantage of a high spatial resolution and can tolerate sensor node losses. Applications of sensor networks are manifold and can be found in agriculture, building management and national security.

It is generally assumed that sensor nodes have an independent power supply. In most cases the power is supplied by batteries. In some scenarios it is also feasible for the sensor nodes to scavenge energy from the environment e.g. solar panels. To achieve the required lifetime it is necessary that the power consumption is lower then the available energy. Therefore, the power efficiency is one of the most important design goals.

The well known and widely used Berkeley sensor nodes have been designed to have a low power consumption in sleep mode. The transition times between sleep and active mode have also been kept to a minimum. Thus even short sleep mode periods become worthwhile. The Berkeley sensor nodes are built from standard components. This has allowed many other researchers to recreate them for their own experiments.

Further reductions in power consumption are possible with custom sensor node SoCs (system-on-chip). Custom SoCs however are very costly and time consuming to build. Thus only very few SoCs are in existence or have been completed.

My dissertation presents a flexible design space exploration platform for wireless sensor networks and an extensible design flow. The conceived platform enables the fast creation and evaluation of custom sensor node hard- and software architectures without developing custom hardware. One important feature of my platform is that it allows the evaluation of the computational- and communication domain of a sensor node in respect to power consumption.

In my platform a SoC under evaluation may be connected to arbitrary real or virtual hardware components. The designer has to decide if real or virtual components must be used depending on the requirements of the experiment. Certain aspects like wireless communication are difficult to realize in simulation. Often it is also necessary to carry out measurements with real components in order to determine simulation model parameters. Simulation has the advantage that many component parameters can be varied quickly. This is especially important for a design space exploration platform. Another advantage of simulation is that rare or dangerous events (e.g. fires and explosions) can be considered.

Independent of whether a component of a sensor node is simulated or real it is necessary to assess its energy efficiency. The design space exploration platform allows this; for real compo-

nents by measurement and for virtual components by estimation. However, first it is necessary to parametrize the power models. Ultimately, measurements are also required to validate the simulated results of the entire system.

My design space exploration platform allows the average power consumption of all major components (SoC, memories, radio transceiver, and sensors) to be determined. The design space exploration platform has highly accurate measurement circuits which can be connected to laboratory equipment. The power supply has been optimized for a clean power signal and not for the highest possible power efficiency - unlike a sensor node.

Besides the average power measurements it is also possible to measure the (hardware simulated) SoC cycle accurately. This extraordinary capability allows power measurements on the microarchitectural level. This is especially important for sensor nodes since more accurate parameters for the SoC simulation model can be acquired. Even small optimizations in the microarchitecture are beneficial due to the required lifetimes in the range of years.

The power estimations are carried out after simulation. If power model parameters are changed then it is not necessary to rerun the simulation. Thus so called "what-if" style analyses are feasible within a reasonable time.

The extensible design flow allows the integration of existing simulation- and estimation tools. The Orinoco platform for example has been integrated to simulate on-chip interconnects. Furthermore, it is possible to integrate custom metrics in a systematic fashion. Special metrics help to optimize individual hardware components. A cache for example is easier to configure at design time if the miss rate is broken down into "compulsory", "conflict" and "capacity". Similar metrics exist for other hardware components.

My dissertation introduces a central application theme to illustrate various points more concretely. The central application theme is a camera tracking sensor network. It has also been used in the case studies.

The first case study presents a sensor node SoC which interfaces with real hardware components. Since it is a camera sensor node the design space exploration platform has been extended with a plugable camera module. The SoC's microarchitecture is presented and its power consumption has been measured cycle accurately. The results and the conceptual limitations of the measurement setup are discussed in detail as well as the mathematical underpinnings for the analysis.

The second case study covers the extensible design flow. First an exemplary metric for the identification of non-contiguous and frequently repeating instruction sequences is introduced and analyzed. If frequently executed instruction sequences have been found then it is desirable to replace them with hardware accelerators. Additionally, my dissertation explains how such custom metrics can be included into the design flow. First a simple back-on-the-envelope calculation is presented. Then an analysis of two mathematical kernels through the metric follows.

In conclusion my design space exploration platform allows the evaluation of arbitrary sensor node hard- and software architectures. Especially, the manifold means to assess the power consumption are important. Existing platforms do not provide hardware reconfiguration and

power measurement. Furthermore, it is possible to publish architecture specifications (in Verilog or VHDL) so that all researchers which have the design space exploration platform can follow and reproduce the experiments of their colleagues. The design space exploration platform is build from standard components like the aforementioned Berkeley sensor nodes. Since the schematics are included in my dissertation it is possible for my platform to reach a similar distribution. My work provides the foundations and concepts for the systematic exploration of sensor node SoCs. I hope that other researchers feel encouraged to pick up this interesting topic.

# Zusammenfassung

Drahtlose Sensornetzwerke sind in ihre Umgebung eingebettete Systeme. Die verteilten Sensorknoten kooperieren miteinander, um ein gemeinsames Ziel zu erreichen. Durch eine entsprechend hohe Anzahl von Sensorknoten kann eine hohe räumliche Auflösung und Verfügbarkeit erzielt werden. Sensornetzwerke eignen sich für zahlreiche und sehr unterschiedliche Anwendungen. Bekannt sind Anwendungen aus der Landwirtschaft, dem Gebäudemanagement oder auch der nationalen Sicherheit.

In den allermeisten Fällen wird eine autonome Energieversorgung angenommen. Dabei kommen meist Batterien zum Einsatz, in seltenen Fällen wird auch Energie aus der Umgebung gewonnen, z.B. durch Solarzellen. Um die anvisierte Laufzeit zu erreichen, muss der Energieverbrauch niedriger sein als die verfügbare Energie.

Eines der wichtigsten Entwurfsziele ist es daher, die Energieeffizienz zu optimieren. Die bekannten und weitverbreiteten Berkeley-Sensorknoten sind auf eine geringe Energieaufnahme im Schlafzustand optimiert. Auch der Übergang zwischen Schlaf- und Wachzustand ist sehr kurz. Dadurch rechnen sich bereits sehr kurze Schlafintervalle.

Die Berkeley-Sensorknoten sind aus Standardkomponenten zusammengestellt worden. Dadurch lassen sie sich einfach nachbauen, was auch vielfach geschehen ist. Noch größere Energieeinsparungen sind durch spezielle Sensorknoten SoCs („System on Chip“) möglich. Die Umsetzung erfordert jedoch viel Zeit und Geld. Daher gibt es nur wenige, oft unvollendete Sensorknoten SoCs.

In meiner Dissertation wird eine flexible Entwurfsraumplattform für drahtlose Sensornetzwerke vorgestellt. Ein wichtiger Aspekt der Arbeit ist auch der dazugehörige und erweiterbare „design flow“. Die konzipierte Plattform ermöglicht die schnelle Umsetzung und Bewertung eigener Sensorknoten- Hard- und Softwarearchitekturen – ohne eigene Hardware zu entwickeln.

Ein Schwerpunkt meiner Arbeit liegt auf der Beurteilung von Berechnung und Kommunikation in Hinblick auf den Energieverbrauch. Ein Sensorknoten-SoC kann in der Entwurfsraumplattform mit beliebigen realen und virtuellen Hardwarebausteinen verbunden werden. Ob reale oder virtuelle Bausteine zum Einsatz kommen, ist eine Entscheidung, die der Entwurfsingenieur in Einklang mit seinen experimentellen Anforderungen treffen muss.

Gewisse Aspekte, wie zum Beispiel die drahtlose Kommunikation, lassen sich oft nur unzureichend in einer Simulation nachbilden. Oft müssen auch Messungen an realen Bauteilen durchgeführt werden, um die simulierten Modelle zu parametrisieren. Die Simulation hat den Vorteil, dass Bauteilparameter beliebig und schnell variiert werden können. Dies ist besonders für

eine Entwurfsraumplattform von Vorteil. Ein weiterer Vorteil der Simulation ist, dass auch seltene oder gefährliche Ereignisse (z.B. Explosionen und Feuer) mit einbezogen werden können.

Unabhängig davon, ob ein Bauteil eines Sensorknoten simuliert wird oder real vorhanden ist, muss es möglich sein die Energieeffizienz zu beurteilen. Die Entwurfsraumplattform ermöglicht dies – bei realen Bausteinen durch Messung, in der Simulation durch Schätzung. Wobei die Energiemodelle erst durch Messungen parametrisiert werden müssen. Letztlich sind Messungen auch notwendig, um die simulierten Ergebnisse des Gesamtsystems zu validieren.

Meine Entwurfsraumplattform ermöglicht die durchschnittliche Leistungsmessung aller wichtigen Bausteine (SoC, Speicher, Funk, Sensorik). Sie verfügt über hochgenaue Messschaltungen, die mit Labormessgeräten verbunden werden können. Die Stromversorgung ist hinsichtlich eines sauberen Stromsignals optimiert und nicht hinsichtlich der größtmöglichen Effizienz, wie dies bei einem Sensorknoten der Fall ist.

Neben der durchschnittlichen Leistungsmessung ist auch eine zyklengenaue Leistungsmessung am (Hardware simulierten) SoC möglich. Diese außergewöhnliche Fähigkeit erlaubt es auch Leistungsmessungen auf der Mikroarchitekturebene durchzuführen. Dies ist insbesondere für Sensorknoten von Vorteil, da sich dadurch genauere Parameter für die Simulationsmodelle des SoC gewinnen lassen. Selbst kleine Optimierungen der Mikroarchitektur machen sich bezahlt, da oft Laufzeiten von mehreren Jahren für Sensorknoten gefordert werden.

Die Leistungsschätzungen werden nach der Simulation durchgeführt. Wenn die Leistungsmodellparameter geändert werden, kann die Schätzung erneut durchgeführt werden, ohne dass die Simulation wiederholt werden muss. Dadurch lassen sich sogenannte „what-if“ – Szenarien in vertretbarer Zeit durchspielen.

Der erweiterbare „design flow“ der Entwurfraumplattform erlaubt es vorhandene Simulations- und Analyse-Software zu integrieren. So wurde z.B. die Orinoco - Plattform zur Simulation von Chip-Verbindungsnetzwerken (Busse) verwendet. Desweiteren lassen sich Berechnungen eigener Metriken systematisch integrieren. Spezielle Metriken helfen dabei einzelne Hardwarekomponenten gezielt zu optimieren.

Ein Cache zum Beispiel lässt sich viel leichter konfigurieren, wenn die „miss“ – Rate in „compulsory“, „conflict“ und „capacity“ aufgeschlüsselt ist. Ähnliche Metriken existieren auch für andere Hardwarekomponenten.

In meiner Dissertation wird ein durchgehendes Anwendungsbeispiel eingeführt, um die besprochenen Sachverhalte anschaulich darzustellen. Es handelt sich dabei um ein Kamera-Sensornetzwerk zur Verfolgung beweglicher Objekte. Das Anwendungsbeispiel wird unter anderem auch in den Fallstudien herangezogen.

In der ersten Fallstudie wird ein Sensorknoten SoC mit realen Bauteilen vorgestellt. Da es sich um einen Kamerasensorknoten handelt, wird die Entwurfsraumplattform um ein steckbares Kameramodul erweitert. Im Folgenden wird die Mikroarchitektur des SoC eingeführt. Das SoC wird auf einem FPGA simuliert, dessen elektrische Leistung zyklengenau gemessen wurde. Die Ergebnisse werden im Anschluss an die Messung erläutert. Dabei werden auch die grundsätzlichen Einschränkungen der Messmethodik diskutiert.

Der Messaufbau und die mathematischen Gleichungen zur Auswertung sind in der Dissertation ausführlich beschrieben.

Die zweite Fallstudie befasst sich mit dem erweiterbaren „design flow“. Zuerst wird eine beispielhafte Metrik zur Identifikation von nicht zusammenhängenden, oft wiederholten Instruktionsfolgen vorgestellt und analysiert. Mit Hilfe der Metrik lassen sich Instruktions-Sequenzen identifizieren, die sich wohlmöglich energiesparend in Hardware-Beschleuniger umwandeln lassen. Desweiteren wird erläutert, wie eine eigene Metrik sich in den „design flow“ einfügt. Zur Veranschaulichung wird ein einfaches Rechenbeispiel vorgestellt, dem die Analyse zweier mathematischer Kernel durch die Metrik folgt.

Zusammenfassend lässt sich festhalten, dass die Entwurfsraumplattform die Beurteilung beliebiger Sensorknoten Hard- und Softwarearchitekturen ermöglicht. Dabei ist insbesondere die Beurteilung des Leistungsverbrauchs in vielfältiger Hinsicht möglich. Bisherige Plattformen bieten nicht die Möglichkeit zur Hardware-Rekonfiguration und Leistungsmessung.

Desweiteren lassen sich die Architektur-Spezifikationen (VHDL oder Verilog) elektronisch verbreiten, so dass jeder Forscher, der über die Entwurfsraumplattform verfügt, die Experimente seiner Kollegen nachvollziehen und fortführen kann. Die Entwurfsraumplattform selbst besteht aus Standard-Komponenten wie die eingangs angesprochenen Berkeley Sensorknoten. Da die Schaltpläne in der Dissertation enthalten sind, steht einer ähnlichen Verbreitung der Entwurfsraumplattform nichts im Weg.

Mit dieser Arbeit wurden folglich die Grundlagen und Konzepte für die systematische Erforschung von Sensorknoten SoCs gelegt. Ich hoffe, dass dadurch andere Forscher sich ermutigt fühlen, diese interessante Thematik aufzugreifen.

# List of Abbreviations

| | |
|---|---|
| 2-FSK | Binary Frequency Shift Keying |
| ASIP | Application-Specific Instruction-set Processor |
| ASK | Amplitude Shift Keying |
| BER | Bit Error Rate |
| CDMA | Code Division Multiple Access |
| CPLD | Complex Programmable Logic Device |
| CRC | Cyclic Redundancy Check |
| CSMA/CA | Carrier Sense Multiple Access with Collision Detection |
| D/A | Digital-Analog |
| DSP | Digital Signal Processor |
| DSSS | Direct Sequence Spread Spectrum |
| FDD | Frequency-division duplexing |
| FDMA | Frequency Division Multiple Access |
| FEC | Forward Error Correction |
| FFT | Fast Fourier Transform |
| FPGA | Field Programmable Gate Array |
| GFSK | Gaussian shaped Frequency Shift Keying |
| MAC | Media Access Layer |
| MCU | Microcontroller Unit |
| MEMS | Micro-electromechanical systems |
| MSK | Minimum Shift Keying |
| OOK | On-Off Keying |
| PCB | Printed Circuit Board |
| PE | Processing Element (e.g. processor, DSP) |
| PER | Packet Error Rate |
| PROM | Programmable Read Only Memory |
| QPSK | Quadrature Phase Shift Keying |
| RF | Radio Frequency |
| RSSI | Received Signal Strength Indicator |
| SNR | Signal-to-Noise Ratio |
| SoC | System-on-Chip |
| TDD | Time-division duplexing |
| UHF | Ultra High frequency |
| USD | US-Dollars |
| UWB | Ultra Wide Band |
| WSN | Wireless Sensor Networks |

# Contents

# List of Tables

# List of Figures

# 1 Introduction

## 1.1 Introduction and Background

Wireless sensor networks (WSN) are networked computers which are embedded in the environment where they sense and measure (distributed) phenomena. The sensing in a sensor node can range from temperature measurement to demanding tasks such as image recognition. Applications [5] range widely from national security (battlefield monitoring, border control [57], disaster management [22, 21]) to civilian uses in agriculture [142], building- and environmental-monitoring ([149, 146, 90, 97]- see Figure 1.1).

The task of a sensor network is typically achieved by *cooperation* among the sensor nodes. The distributed nature of the network enables tracking applications, increased measurement resolution and propagation of results. The topology of the network may change (frequently) due to the addition or loss of sensor nodes. Since WSNs may be in remote locations, widespread, unattended and large in size it is a necessity that they *self-organize* and *adapt* to changing conditions. *Available energy* is one of the most important (changing) resources. It determines the quality of service and availability of a WSN. Therefore it is necessary to design the hard- and software of a sensor node in such a way that the application requirements are met with the smallest possible power consumption.

A back on the envelope calculation:

If a (basic) sensor node has to run for one year on a single AAA-battery-cell with a capacity of $750mAh$ it may not draw more than $\approx 100\mu W$ on average:

$$I_{avg} = \frac{Q}{t} = \frac{750mAh}{8760h} = 85.62\mu A \tag{1.1}$$

$$P_{avg} = U \cdot I_{avg} = 1.2V \cdot 85.62\mu A = 102.74\mu W \approx 100\mu W \tag{1.2}$$

Table 1.1 shows the power consumption of a basic- and advanced sensor nodes. The two classes of sensor nodes differ in radio bandwidth, processing power and sensing capabilities. It can be seen that the power consumption varies significantly. The available power sources - however - do not.

Therefore computation, communication and sensing must be power efficient especially in advanced sensor nodes.

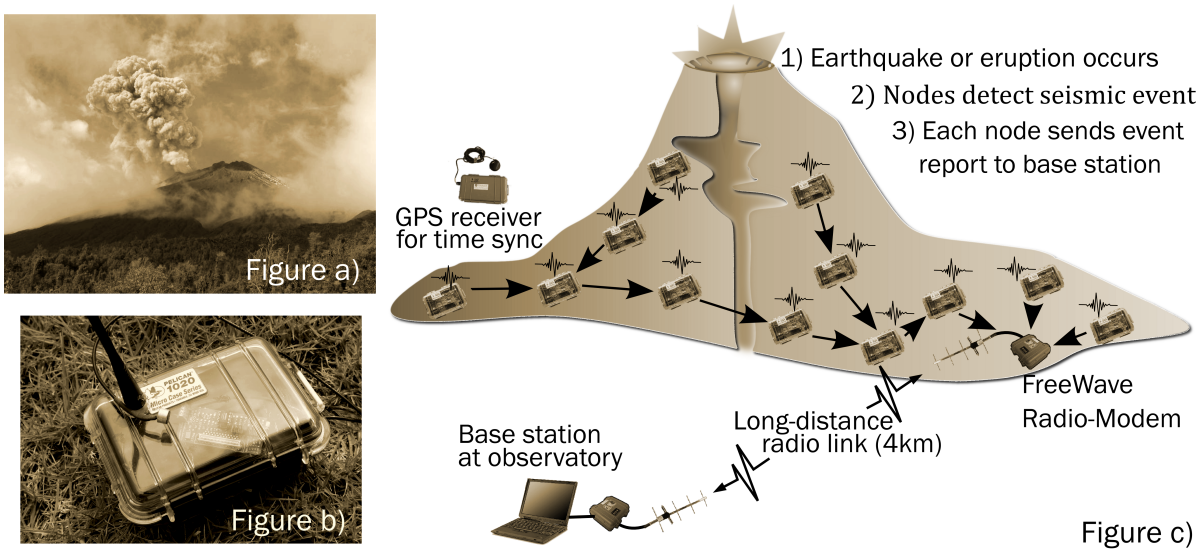



Figure 1.1: Deployment of a sensor network at the volcano Tungurahua (Figure a) in central Ecuador [147]. The sensor nodes (Figure b) measure seismic and infrasonic (low frequency sound) - signals. Figure (c), shows the architecture of the sensor network: one node supplies a GPS timebase for synchronizing the measurements. The other sensor nodes use a multi-hop network protocol to forward acquired data to the (mountain)-gateway. The gateway collects and transmits the node's data over a long distance wireless link down to the base station. The researchers there can store and analyze the incoming data nearly in real time. Traditional measurement equipment required the researchers to climb up to the mountains and read out memory cards from a few, expensive measurement stations. The low cost of the sensor nodes allows the researchers to measure at more locations than had been possible before. The low power consumption of the sensor nodes requires weekly battery changes. Pictures are based on[145].

| Device | Active Power Consumption | Standby Power Consumption |
|---|---|---|
| Basic sensor node | | |
| 500kbit Radio (CC1100) | 40 *to* 90$mW$ (−6 to 10$dbM$) | 1.2$\mu W$ |
| 1 MHz Processor (MSP430) | 440$\mu W$ | 1.5$\mu W$ |
| Temperature sensor (LM75) | 924$\mu W$ | 13$\mu W$ |
| Advanced sensor node | | |
| 2 MBit Radio (STLC4560) | 716$mW$ | 36$\mu W$ |
| 270 MHz Processor (ARM LPC3141) | 122$mW$ | 3.8$mW$ |
| VGA Digital Image Sensor (MT9V111) | 80$mW$ | 14$\mu W$ |

Table 1.1: Typical power consumption of different radio transceivers, processors and sensors. The radio transceivers consume the most power (when active) independent of the sensor node class (basic / advanced). The power consumption varies between the modes: active and standby. The given estimates can be influenced if configured differently - for example - by changing the bandwidth (radio), frequency (processor) and image capture rate (image sensor).

The hardware devices shown in the table have a *plethora of settings* that influence power consumption:

The radios can be configured for different bandwidths, modulation modes and error correction algorithms. The digital image sensor can vary the number of pictures captured per second, the resolution and color depth. The processor can be run with or without memory management unit and external memory. Additionally, the devices often support more than two power modes which vary in transition time towards active mode.

Thus the *design space* of the run-time configurable hardware settings alone is already vast.

## 1.2 Design Space Exploration

Table 1.2 shows exemplary *(software driven) simulators* which are well known in the WSN-community. They are useful for design space exploration of protocols with *existing* hard- and software. Currently, they have built-in models of basic sensor nodes.

Eventually, a software driven simulator needs to be validated in order to assess the soundness

| **Simulator Name** | Main uses | Simulation model | Scalability | Research Group / Company |
|---|---|---|---|---|
| NS-2[2] | routing, protocols | discrete event | (large) networks | USC/ISI, UCB, Xerox PARC, LBL |
| Omnet++[139] | routing, protocols | discrete event | (large) networks | Simulcraft Inc. |
| Avrora[134] | bit-level accurate sensor node simulation | discrete event / cycle accurate | single node small networks | UCLA |
| PowerTOSSIM [123, 88] | TinyOS-API simulator, rough power estimates | host native code execution and performance counter | single node small networks | Harvard (PowerTOSSIM), UCB (TOSSIM) |

Table 1.2: Software simulators that target or support aspects of sensor network simulation: The network simulators (NS-2/Omnet++) allow the design space of protocols to be explored. They abstract away many properties of the underlying hard- and software implementation and can therefore provide a faster simulation speed. This helps to shorten the modeling time. Certain properties of radio channel communication are supported to increase the realism. Avrora models only existing hardware but could be extended to simulate hypothetical devices too. (Power)-TOSSIM is restricted to application level simulation. It has simple support for power consumption estimation.

of the results. Especially, the communication performance over a wireless medium is time consuming and hard to model accurately. Radio waves propagate in a three-dimensional space and can be absorbed, reflected, experience refraction, diffraction and scattering. Additionally, realistic terrain models are required to match the deployment environment.

Currently, the community lacks advanced tools, methods and data to simulate an entire sensor network that captures all these aspects in sufficient detail. Even if more accurate tools, realistic terrain- and sensor-node models were available they would - most likely - require (respective) simulation times due to the scale and complexity of the simulated world. Current simulation run-times pose already a serious constrain to the exploration of (larger) networks.

## 1.3 Motivation

Experimentation with real sensor nodes is inherently time consuming and difficult to master. Commercially available sensor nodes[1] lower the entrance barrier and enable first experiments. Due to the *application specific nature* and *power constrains* of sensor nodes it is inevitable to eventually deviate from standard designs and *explore a wider design space*. This *design space exploration* may initially be based on software simulation.

However, to determine the *ground truth* it is necessary to extend the design space exploration in a way that allows interaction with the real environment ("*hardware-in-the-loop*").

Off-the shelve sensor nodes are fixed and cannot be easily changed. Extending existing or creating new hardware for experiments is a *time consuming* venture which requires sufficient resources and special skills.

This thesis introduces a flexible and reconfigurable *design space exploration platform* for sensor node hard- and software which enables "*hardware-in-the-loop*"-based exploration with power-consumption feedback of individual components even down to the microarchitectural level.

## 1.4 Outline

An overview of the thesis outline can be seen in Figure 1.2.

**Chapter 2** introduces state-of-the-art related work. Different (commercial) sensor node platforms are introduced and their design objectives are discussed. The distinguishing features of their hard- and software-architectures are highlighted and put into relation with each other and the design space exploration platform of this thesis. A special focus is on power-measurement and -estimation of wireless sensor networks. Furthermore, the challenges of modeling radio wave propagation are emphasized, so that the reader becomes conscious about the limitations of communication simulation in wireless sensor networks.

**Chapter 3** presents the system vision of a flexible design space exploration platform for wireless sensor networks which is the aim of this thesis. A distinction is made between sensor nodes for simple (sensing) tasks and advanced sensor nodes which are much more capable and thus provide a richer design space. Then the exemplary and central application theme is introduced: a camera tracking sensor network application. At the end of the chapter the methodology followed within this thesis is laid out.

**Chapter** 4 provides an overview of the design space of wireless sensor networks. The domains computation, communication, sensing and power supply are established and then covered separately. The focus - in this thesis - is on exploring the computation- and communication-

[1] Most commercially available sensor nodes are not meant for deployment but for research and testing purposes like every other evaluation board. Those companies usually advertise their services in hard- and software-design for (application specific) sensor network applications.

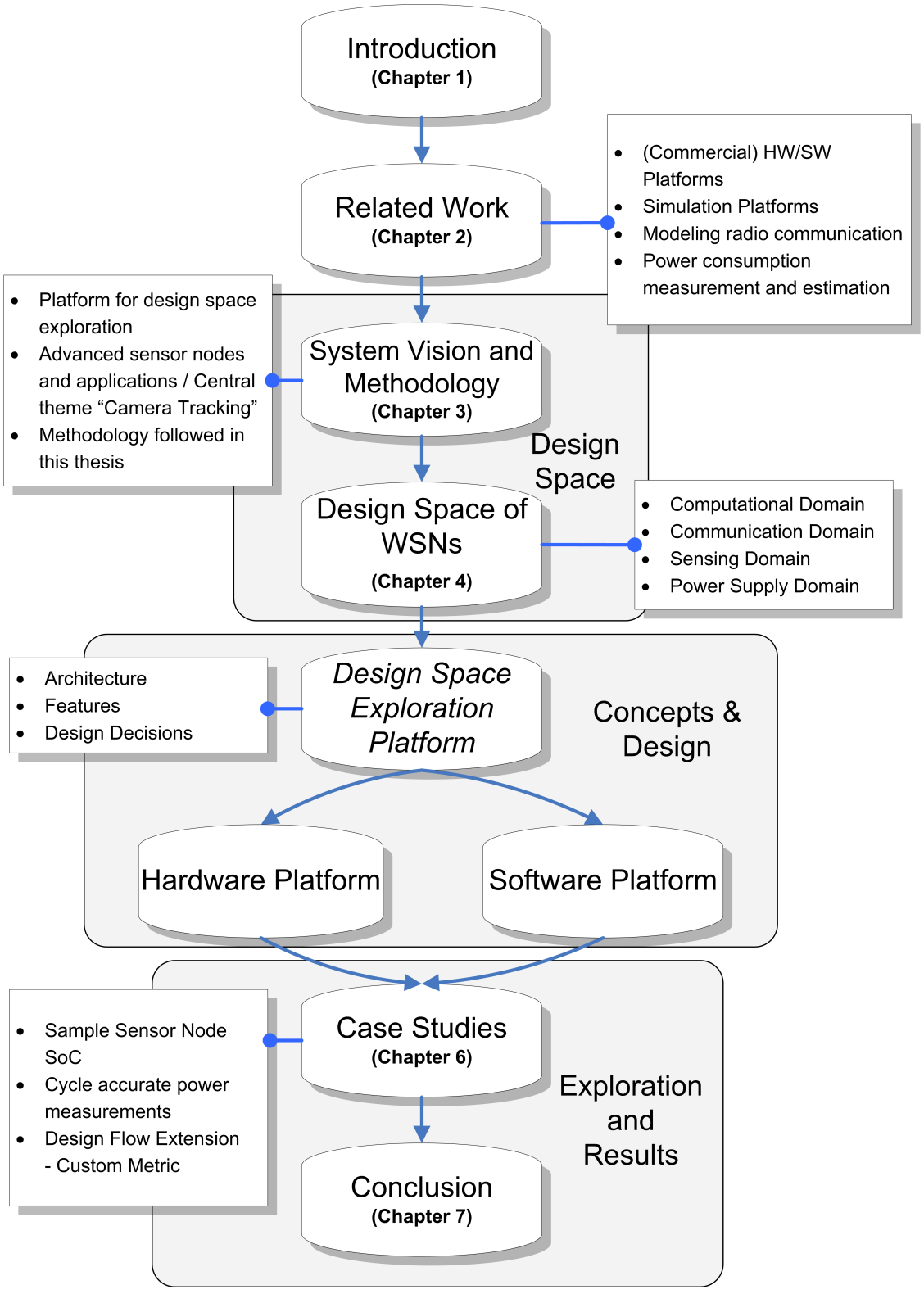



Figure 1.2: Outline of this thesis

domain. The computation domain encompasses processing elements and memory. The communication domain is covered more in-depth - in this chapter - since wireless communication is the distinguishing feature of wireless sensor networks and probably least familiar to the target audience of this thesis.

**Chapter 5** lays out the *design flow* and localizes the *design space exploration platform* within it. Both the design flow and the platform have been conceived in this thesis. An important part of the design flow is the requirement specification. The chapter outlines the mapping of physical- and non-physical features on node- and network-level. The design space exploration platform consists of two parts: a hardware- and the software driven exploration platform. Their architectures, underlying concepts, algorithms, formal underpinnings and components are presented in detail. Special attention has been paid to the power-measurement and -estimation capabilities.

In **Chapter 6** selected case studies demonstrate how the platform can be utilized to explore the design space of wireless sensor networks. The central application theme of the system vision chapter is picked up where appropriate.

**Chapter** 7 concludes the thesis with a summary and gives an outlook into future challenges.

The following chapter presents the related work and lays the foundation for the later chapters.

# 2 Related Work

## 2.1 Introduction

This chapter introduces state-of-the-art related work. It starts with the introduction of current commercial and academic sensor node platforms. It covers their soft- and hardware-architectures which is followed by a discussion of simulation environments and their capabilities for design space exploration with a strong focus on power estimation. Afterwards challenges encountered in modeling and simulating radio communication and their consequences are discussed. The chapter closes with a presentation of related work in the field of (cycle)-accurate power measurement for (sensor node) chips.

In some cases the discussion of related work contains information which I received by personal email correspondence with the authors. Where necessary I have pointed this out in the relevant text.

## 2.2 Background: (Commercial) Sensor Node Platforms

Quite a number of commercial and academic sensor node platforms have been developed. One of the most influential sensor node platforms has been developed by David Culler's group at the University of California, Berkeley. The design rationals of the Mica and Telos sensor node series have been published in [61, 106]. A major component of the Berkeley platform is the application specific operating system: TinyOS [69] and its special programming language nesC [51] as well as the complementing simulation environment TOSSIM [88].

The technical specifications of the hardware- and software have been [3] open sourced and various universities and businesses have created clones of the hard- and software[1]. The commercialization has enabled researchers to carry out experiments on a standardized platform without the design- and production effort associated with developing a hard- and software platform.

Another well-known (commercial) sensor node platform is the SunSPOT sensor node from Sun Microsystems. The SunSPOT devices have significantly more memory and processing power. They need more resources compared to the Berkeley nodes in order to run an embedded Java virtual machine called SquawkVM. SquawkVM allows the application designers

[1] Crossbow Inc. commercialized the designs first[1]. A Telos sensor node costs around 100 USD (2009) .

to use a familiar and managed[2] programming language in exchange for higher product costs[3] and power consumption. Therefore, the application domain of the SunSPOT sensor node is rooted in prototyping of small scale networks.

### 2.2.1 Design Objectives

David Culler - from UC Berkeley - has broken new grounds with his platform. His design decisions are based on the assumption of *large scale, deeply embedded* sensor networks. In deeply embedded sensor networks common assumptions are that sensor nodes are small in size and have very limited power, computation- and communication resources. These limitations also help to keep the cost of an individual sensor node low, so that large scale deployments become imaginable.

Deeply embedded sensor node platforms have been built with off-the-shelve hardware components and still perform reasonably [106, 72][4].

A drawback of Culler's platform are its severe hardware constrains which limit the application design space to tasks satisfiable with low-bandwidth communication, small memories and little computational power. Obviously, deeply embedded sensor networks are only one application scenario.

Sun Microsystems has created a research platform centered around their Java technology to explore ideas around the "Internet-of-Things". Sun's hardware platform is more capable compared to Culler's but they are both fixed in terms of their hardware components. The radio transceiver, the processor(s) , memories and wiring cannot be changed unless new hardware is manufactured. The design space exploration is thus constrained to the software programmable domain.

*The design space exploration platform devised in this thesis supports both: the exploration of the soft- and hardware domain.*

Both, Sun's and Culler's hardware platform are extensible. The extensibility is limited to the integration of sensors. Crucial components like memories and the radio transceiver are integrated on the main board. The platform of this thesis goes further: it has extension ports for (high-speed) components. Thus high-bandwidth memories, -sensors (e.g. image sensors) and -radio transceivers can be interfaced. Additionally, the platform has extensive support[5] for extremely accurate power measurement which is not found in the aforementioned standard sensor nodes.

[2] Java has automatic memory management, array bounds- and pointer checks as a safeguard against program faults. Therefore it can be safely ran on shared hardware without memory protection (unit).

[3] A SunSPOT-kit with two sensor nodes and a base stations costs 750 USD (2009).

[4] A few specialized sensor node SoCs have been published but little information is available about their performance and usage in real deployments [43, 44, 108, 62]. Due to their tiny size they were referred to as smart dust [143, 74].

[5] The components on the board are powered by separate linear regulators, the shunts are in the milliohm range and of highest precision. Furthermore, the board (layout) has been designed to support cycle accurate power measurements of the logic chip.

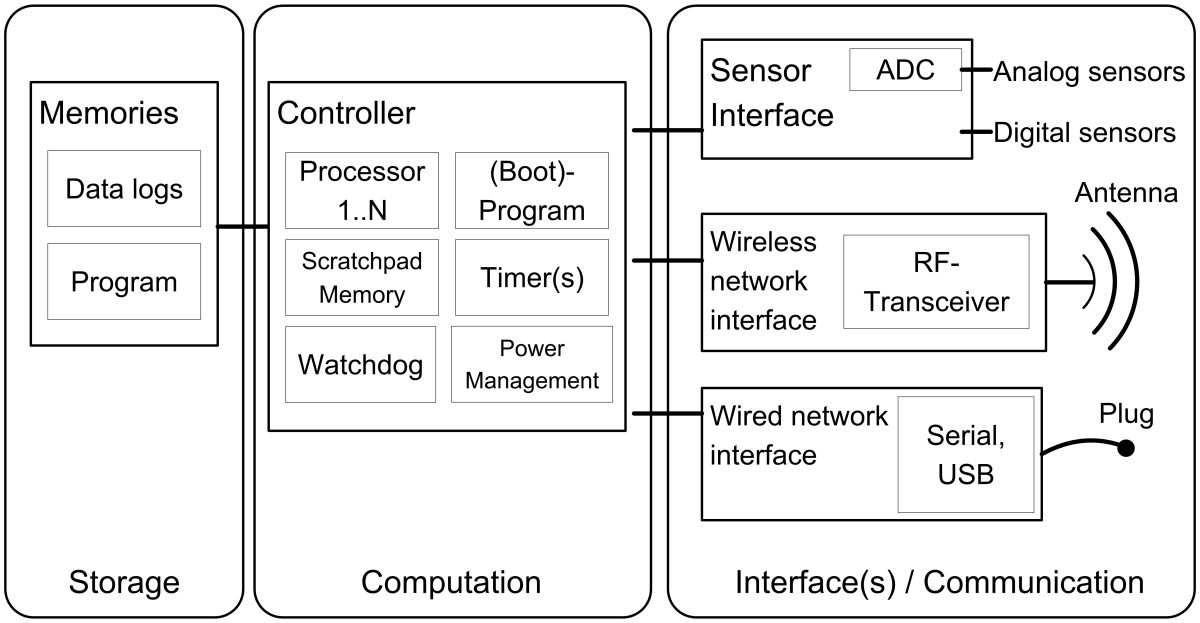



Figure 2.1: Abstract view on a basic sensor node hardware platform (Figure inspired by [32])

Power consumption feedback is - however - *crucial* to guide the design space exploration of a WSN design.

*My platform is designed to foster the design space exploration of advanced sensor nodes.* Advanced sensor nodes encompass application specific processors [141], hardware accelerators [62, 108], custom peripheral interfaces and memory hierarchy optimizations [64, 133, 63]. Latter were carried out in pursuit of this thesis.

Culler built trivial "hardware accelerators" using the peripherals of the microcontroller due to the lack of custom silicon or re-programmable logic. Despite the limitations he achieved already significant reductions in power consumption [61].

*I would like to emphasize that my sensor node platform has been conceived as a flexible design space exploration platform for WSN hard- and software. It is not meant for deployment* [38].

## 2.3 Hardware Architectures of (commercial) Sensor Node Platforms

### 2.3.1 Abstract View

Figure 2.1 shows the different design domains of a basic sensor node hardware platform. The controller in the middle connects to the memory- and interface- domain. The storage domain

is necessary to keep data and program code (persistently). For computation one or more processors are required. Simple platforms usually have one processor core. Telos for example has a single core 16 bit-MCU.

More advanced platforms like the SunSPOT (a 32 bit- and an 8 bit MCU) have more than one core. The SunSPOT has a 32bit core which is responsible for demanding processing tasks. The 8bit core runs the power management algorithms. Besides the cores the SoCs (system-on-chip) usually have non-volatile flash memory to hold the (boot) program code and low latency scratchpad memory. Typical SoC peripherals are general purpose timers, watchdog timers and a power management unit. Latter can enable and disable on-chip functional units and adjust the clock.

The design space exploration platform of this thesis allows *the computational domain to be user-defined*. This enables the design space exploration of novel hardware architectures for sensor network applications.

### 2.3.2 Concrete Architectures and their Design Rationales

#### Mica

Figure 2.2b) shows the concrete hardware architecture of a Mica node from UC Berkeley[6]. The three different domains computation, storage and interface have been highlighted according to Figure 2.1. Mica [61]has an 8bit microcontroller as its core processing element [10, 11]. The co-processor's main purpose is to realize over-the-air re-programming which the bigger processor cannot do by itself.

A special chip (DS2401) provides a unique hardware identifier which is convenient to use but not strictly necessary. Generally, it is also possible to reserve space in the flash memory to store an unique identifier.

All available input- and output pins (digital and analog) are available at the connector where the sensor- or programming board plugs in.

The transceiver is on-board and rather primitive. It supports only one modulation scheme. The bit-serial interface of the radio transceiver requires the processor to stuff every bit over the wires which limits the achievable data rate to 20 kbps. To relieve the processor from these low-level tasks and improve performance the authors utilized the microcontroller peripherals (timer and buffers) to accelerate the radio data transmissions and synchronization.

The raw interface of the radio transceiver puts pressure on the microcontroller. In return the raw interface offers more freedom in controlling the communication. The bandwidth, encoding, packet size, timing and media access control (MAC) can be tailored to the application requirements. This may lead to a reduced power consumption.

[6] Table 8.24 gives a detailed overview of the components and their performance.

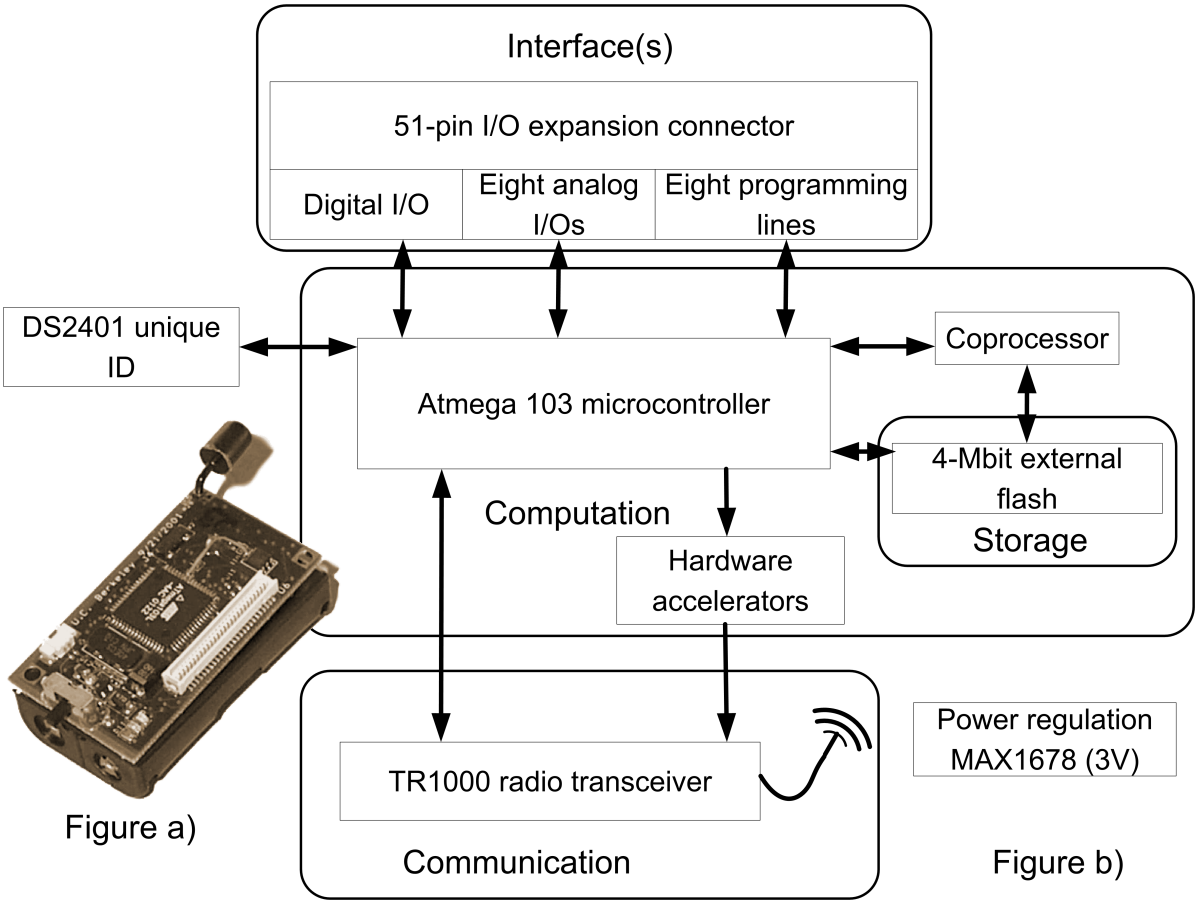



Figure 2.2: Mica-Architecture [61]- designed by Jason Hill and David Culler at the University of California, Berkeley. The three different domains computation, storage and interfaces are highlighted in **b)** (see also Figure 2.1). Figure **a)** shows the hardware device itself. The battery case is on the bottom. The processor is in the middle and the extension port on the right.

*The raw radio interface offers a wider design space. Standardized communication promises interoperability but fixes many aspects of radio communication.*

For Mica the authors had to be inventive to use the radio transceiver efficiently. The hardware driven design space exploration platform of this thesis uses programmable logic. This allows the designer to define custom processors, accelerators and radio-interfaces freely. Custom logic could for example provide a high-level packet interface with buffering and automatic re-transmission to relive the application processor.

The design space exploration platform is not limited to a single radio transceiver. Like sensors and memory the radio transceivers are extensions.

### Telos

Telos [106] is a hardware platform designed by Joseph Polastre and Robert Szewczyk from the University of California, Berkeley. It can be considered as one of the successors of Mica. Telos is a more pragmatic design and has been optimized for deployment. The power supply has been matched to the batteries and connector reliability has been improved as well. The raw radio interface of Mica offers more freedom than the new radio in Telos. The new radio transceiver - however - uses less power and its operation is more reliable. It has built-in hardware accelerators for symbol detection and bit serialization but cannot achieve the accuracy in synchronization which had been possible with Mica's raw radio interface. Another important change concerns the processor. Telos uses a more powerful 16 bit microcontroller which uses even less power.

*Although Telos has seen many improvements it is less suited as a design space exploration platform than Mica. An advantage over Mica is that Telos can be reliably used in deployments.*

Obviously, it was not the authors goal to design a flexible platform to assists design space exploration. They intended to build a sensor node platform usable for real large scale experiments. As such the Telos can be considered a milestone.

One of the assumptions underlying the design of Telos is *low duty cycling*[7]. Low duty cycling means that a sensor node sleeps for most of the time and is woken up infrequently to carry out sensing, signal processing (computation) and communication tasks[72].

The average power can be formulated as:

$$P_{avg} = t_{sleeping} \cdot P_{sleeping} + t_{active} \cdot P_{active} + t_{waking} \cdot P_{waking} \tag{2.1}$$

Where the total average power is determined by the power consumed while sleeping, waking or being active. If waking power is negligible the average power can expressed as a function of the duty cycle:

[7] This assumption has also had its impact on the design of the TinyOS-operating system which is often used with Mica and Telos nodes. It will be covered in a following section.

$$P_{avg} = \alpha \cdot P_{active} + (1 - \alpha) \cdot P_{sleeping} \tag{2.2}$$

Where $\alpha$ represents the duty cycling factor. If we assume that $P_{active}$ is $50mW$, $P_{sleeping}$ $10\mu W$ and $P_{avg}$ $100\mu W$ (see Equation 1.2 on page 1) and solve for $\alpha$:

$$\alpha = \frac{P_{avg} - P_{sleeping}}{P_{active} - P_{sleeping}} \overset{P_{active} \gg P_{sleeping}}{=} \approx \frac{P_{avg} - P_{sleeping}}{P_{active}}$$

$$\alpha \approx \frac{100\mu W - 10\mu W}{50mW} \approx 0.18\%$$

If a sensor node was to run for one year on single AAA-battery cell with a capacity of $750mAh$ as in the introductory example on page 1 then the node may only be active for 15.77 hours throughout the whole year. The Telos node holds two batteries.

The overall design of the Telos has been optimized for low duty cycling according to the authors [106]: it takes the processor $6\mu s$ and the radio up to $580\mu s$ to wake up. For comparison: Mica's processor and radio take $180\mu s$ and $20\mu s$ respectively. Surprisingly, the predecessor platform has a lower total wake up time. In Telos the wake up time is dominated by the new radio transceiver.

### SunSPOT

The SunSPOT platform has been conceived as a Sun Labs research project. The previously introduced platforms Telos and Mica had originated from PhD-projects which have been commercialized. Sun's interest in selling the nodes is (probably) rooted in the idea of spreading the platform within the research community and see how their Java technology is leveraged in novel ways as an embedded systems platform.

The motivation for realizing the platform in the first place has been the dissatisfaction with prior platforms. Sun labs aimed to have a platform with improved software tools and more capable hardware to research security issues and new protocols in sensor networks (Sun Labs Flyer: "*Turning Vision into Reality*").

### SunSPOT - Power Supply, Monitoring and Control (Figure 2.3)

The hardware platform itself has been designed by Robert Alkir who kindly answered my questions by email. A high-level overview which I derived from the schematics is shown in Figure 2.3. The upper right shows the three domains: interfaces, computation and storage. Everything else in the figure is related to *power supply, power consumption monitoring and power-control*:

In Mica and Telos only very little circuitry is related to power supply or even management and none to monitoring.

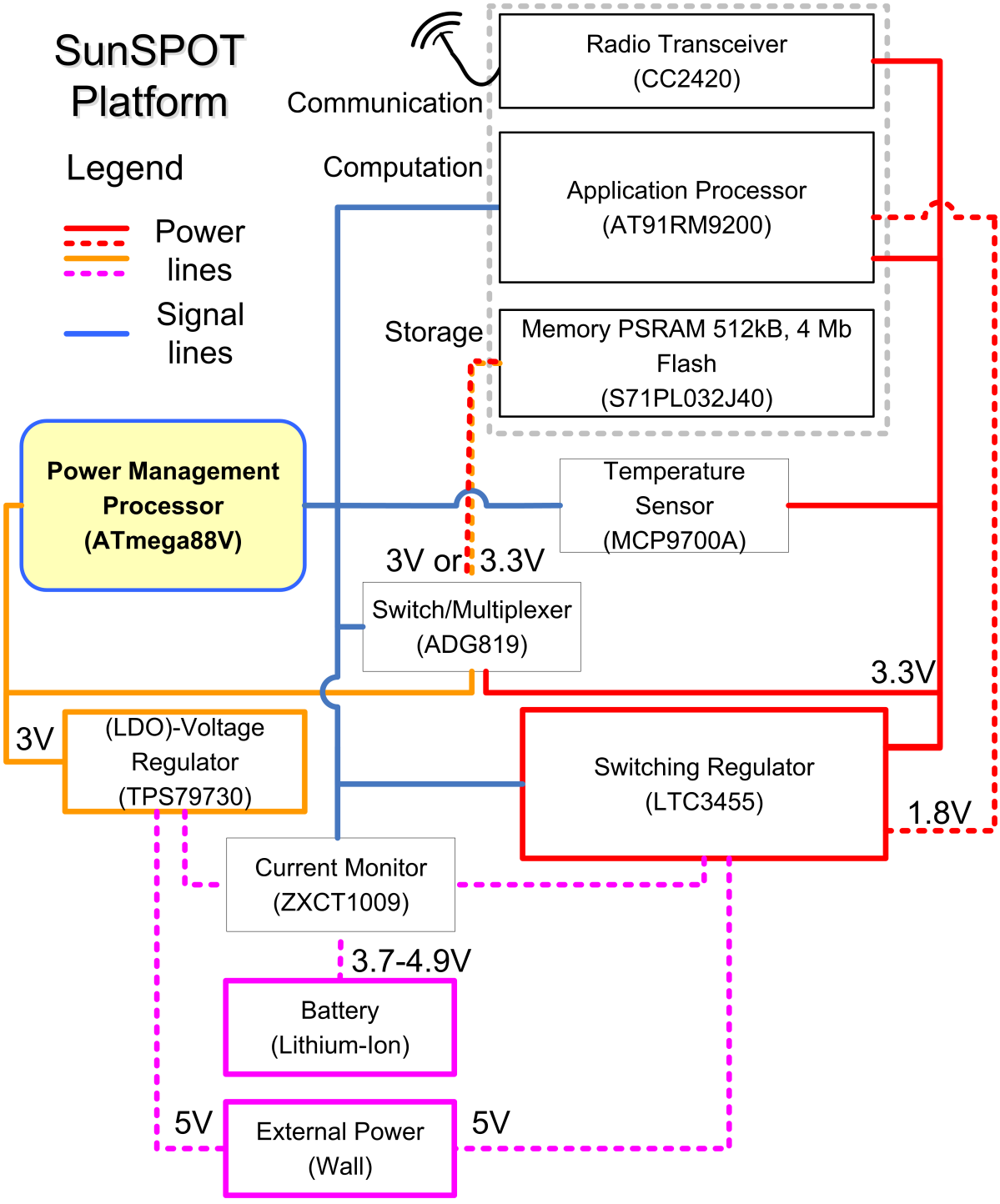



Figure 2.3: SunSPOT platform

The design space exploration platform developed in this thesis has power measurement features which go beyond those of SunSPOT. Every major hardware component has its own regulated power supply. The power supply has been designed to cause very little power signal noise and is therefore not necessarily efficient. The power consumption of individual components (logic, memory, radio, sensor e.g.) can be measured separately to obtain a detailed power break-down. External power measurements are supported by on-board circuitry which interfaces with professional laboratory equipment in order to maximize accuracy.

Back to Figure 2.3: the SunSPOT has two voltage regulators which provide three different voltages[8]. The regulators are fed by either a *lithium-ion* battery or an external source. The LDO-voltage regulator provides the standby current for the power management processor and the pseudo SRAM memory. If the node is active then the memory, application processor and radio transceiver are powered from an efficient *switching* regulator. This regulator also manages the charging of the battery.

At the heart of the power management is a tiny microcontroller (power management processor) which measures and monitors the voltages, the battery current and its temperature[9]. Additionally, it controls the power source switch of the aforementioned pseudo SRAM and the *switching* regulator.

### SunSPOT - Power Consumption and Efficiency

In *active mode* the SunSPOT can reach up to ca. $420mW$ and in *deep sleep mode* it consumes around $120 - 130\mu W$ [130, 129] which is far more than the Telos which needs $15\mu W$[106]. The reason is that the SunSPOT keeps the contents of the pseudo SRAM (512kb) buffered - otherwise it would probably be on par with the Telos. If the memory was not buffered then only a few analog components[10] and the power management processor would be active. Latter is only every 1/256 of a second active according to Robert (power: ca. $0.2\mu W$ in sleep and $27\mu W$ active @ 32 kHz - Atmel data-sheet).

*Thus the (sleep) power* efficiency *of the SunSPOT is comparable to the Telos and Mica but its hardware is much more capable.*

One major drawback of power cycling the application processor, radio and (flash) memory when transitioning to and from deep sleep mode is the prohibitively high wake-up time [11]. Therefore it does not pay off to transition frequently between active and deep-sleep mode:

*Thus the concept of low duty cycling underlying the Telos is not realized in the SunSPOT.*

[8] Mica's architecture (Figure 2.2) has only a single (inefficient) voltage regulator and Telos has none at all,

[9] The temperature sensor is necessary to ensure the battery is never operated in an unstable condition.

[10] Apart from the buffered PSRAM the analog components use most of the standby power. The analog consumers are: LDO voltage regulator, current monitor diodes, power switch, oscillator, decoupling capacitors, pull-up/down resistors

[11] The switching regulator's control circuits take their time to reach a stable output voltage. A short wake-up time is also beneficial if time synchronization has to be implemented.

### SunSPOT - Exploration of Latency- and Power Improvements

To reduce the latency it would be beneficial to power the radio separately and run protocol maintenance tasks such as routing on a smaller and less power hungry processor. Low data rate sampling of sensors could also be handled by a smaller processor.

Small processors can also run directly from batteries. Hence the big application processor could remain switched off for longer.

The responsibilities of the power management processor could be extended to avoid introducing new processors into the system.

The hardware driven platform Hyperion of this thesis supports the exploration of heterogeneous multi-core architectures without re-spinning hardware chips due to its programmable hardware chip.

It was mentioned before that the permanent buffering of the pseudo SRAM consumes most energy in the deep sleep mode. The power consumption of the SunSPOT could be further minimized by using upcoming *non-volatile* memory technologies such as MRAM (Magneto-resistive Random Access Memory) for example. The access latency and bandwidth characteristics are on par with pseudo SRAM but the memory contents are kept even if powered down.

The Theia memory extension for the Hyperion platform is equipped with M- and FRAM to facilitate the exploration of novel memory types. Their wide spread use in sensor nodes is currently limited by their small capacity and high costs.

### SunSPOT - Concluding Remark

The SunSPOT hardware platform is mostly fixed. An exception is its sensor board connector which allows new sensors to be plugged in. Like Telos it uses a narrow band radio transceiver. The memory and processing resources are abundant when compared to Telos or Mica. Due to its heterogeneous multi-processor design, the large memory, the powerful application processor and the power measurement and control capabilities - the SunSPOT sensor node - has a bigger design space. The Java based software platform of SunSPOT supports quick prototyping of software when existing Java libraries and tools are used. As a platform for deployment it may perform sub optimal due to the overhead imposed by the Java virtual machine[12]. The SunSPOT hardware platform can also be used without Java if necessary.

### Hogthrob - A Design Space Exploration Platform

The Hogthrob project aimed to explore the design space of WSN soft- and hardware. The application was livestock monitoring in farming [140]. The requirements [85] outlined that the

[12] The (interpreting) Java virtual machine for SunSPOT simulates a hypothetical machine and requires more CPU cycles and memory than natively compiled code.

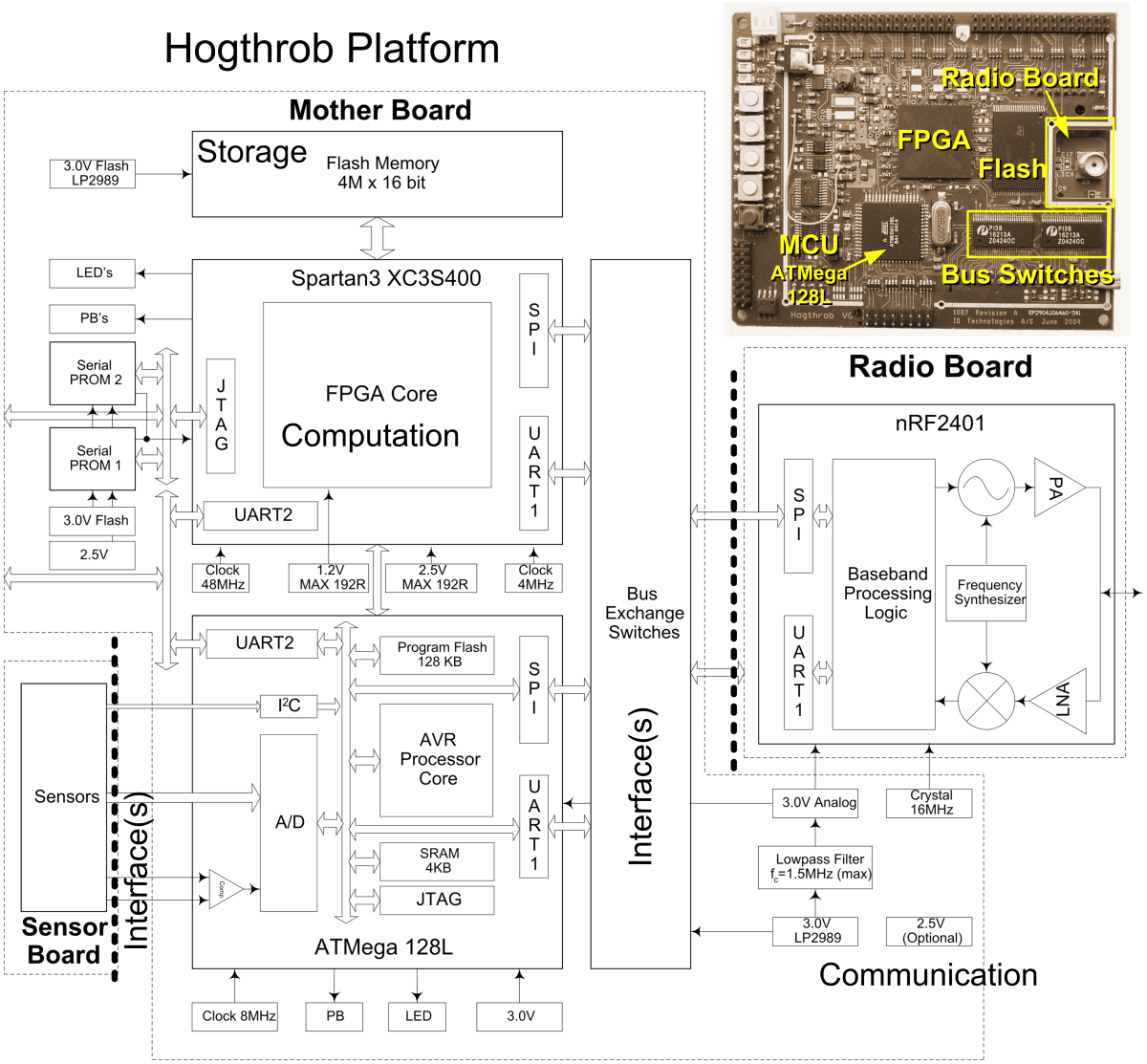



Figure 2.4: Hogthrob hardware platform - The computational part (middle) consists of an ATMega microcontroller and an FPGA. A bus switch controls whether the FPGA or the microcontroller has access to the external radio board. Only the microcontroller has access to the external sensor board. The flash memory on the top is connected to the FPGA. The picture on the top right shows a photograph of the Hogthrob board with a radio module plugged in. The architectural figure and the photo are based on [84].

battery power must last for 6 months, that the hardware must have a small form factor (ear-tag) and be robust and cost as little as possible. The research project was conducted by the following parties:

- Department of Informatics and Mathematical Modeling (IMM) - Technical University of Denmark
- Department of Computer Science - University of Copenhagen (DIKU)
- The Royal Veterinary and Agricultural University of Denmark (KVL)
- National Committee for Pig Production (NCPP)
- IO Technologies

The project started in May 2004 and ended in 2007. IO Technologies developed a hardware programmable exploration platform (see Figure2.4). Like Hyperion the radio and sensor modules are plugable.

There is no additional memory on board except for the on-chip SRAM inside the microcontroller and the FPGA. A possible explanation is that the objective was to explore basic sensor nodes for the aforementioned cost sensitive animal monitoring application.

The hardware driven design space exploration platform *Hyperion* which has been conceived in this thesis can also target advanced sensor nodes. A major difference between the Hyperion- and Hogthrob-platform is that Hyperion has extensive on-board circuitry for accurate power measurements. The power measurement supports all major components: hardware logic, radio transceivers, memories and sensors. *For Hyperion the power measurement capabilities have been a major reason to engineer a custom board in the first place.*

The Hogthrob hardware platform could have been realized with a standard FPGA board and (custom) daughter boards for sensing and communication. A description of the hardware can be found in the Hogthrob manual [84] or in in this publication [140].

Although a lot of engineering effort had been invested into the platform it was not used for design space exploration or deployment. This has been confirmed by Martin Leopold and Marcus Chang via email. Nonetheless, the project contributed an interesting performance estimation methodology which is discussed on page 32.

## 2.4 Software Architectures of (commercial) Sensor Node Platforms

Before the appearance of sensor nodes microcontroller based embedded systems were programmed in hand-written assembler code or C-dialects to cope with the severe resource constrains and specialties of embedded systems[13]. The sensor network community has generated

[13] In industry this is still common practice.

a series of alternative approaches. Partly, because many of the researchers have a computer science background and enjoy building new systems from scratch but also due to the special requirements found in sensor networks. Unlike many other embedded systems it is not uncommon for sensor nodes to communicate over radio, require online software updates, have an *energy dependent lifetime* and be programmed by researchers from various disciplines to carry out scientific experiments.

In the following section three different software architectures of sensor network operating systems are introduced and compared. They have been chosen because each of them has a distinct point in the *design space*:

Their run time models provide different possibilities for simulation and therefore *power estimation techniques* (see section 2.7 on page 42). The different software architectures and their underlying run time models may be worthy targets for hardware acceleration. They are responsible for the integration of HW/SW-components their run-time adaptation, reconfiguration, the task mapping and scheduling.

## 2.4.1 Overview

In the following section the architectural features of the software platforms: *SOS* (from UC-Los Angeles), *TinyOS* (from UC-Berkeley), *Contiki* (from the Swedish Institute of Computer Science) and *the Squawk Virtual Machine* (from Sun Research) are introduced:

- TinyOS is often cited in WSN publications due to its novel design and optimization towards deeply embedded sensor nodes.
- The main focus of the SOS authors was to prove that even basic sensor nodes can afford run time reconfiguration and management of software modules.
- Contiki has taken a pragmatic approach and has a wide set of modules which cover many use cases. Its flexibility combined with an Internet protocol stack have lead to industry adoption where Contiki is used as a general purpose embedded operating system.
- The Squawk virtual machine enables Java applications to run on the SunSPOT sensor node (see page 15). This lowers the entry barrier for those who prefer Java (excellent tools, huge community) but entails non-negligible overheads in latency, memory-space and processing power.

### Software Architectures

A TinyOS [87, 69] system is assembled from a set of (fine-grained) reusable software components[14] at design time. The assembly is compiled according to a wiring specification.

*TinyOS can be considered as an application specific operating system.*

[14] One intrinsic component is the event-scheduler.

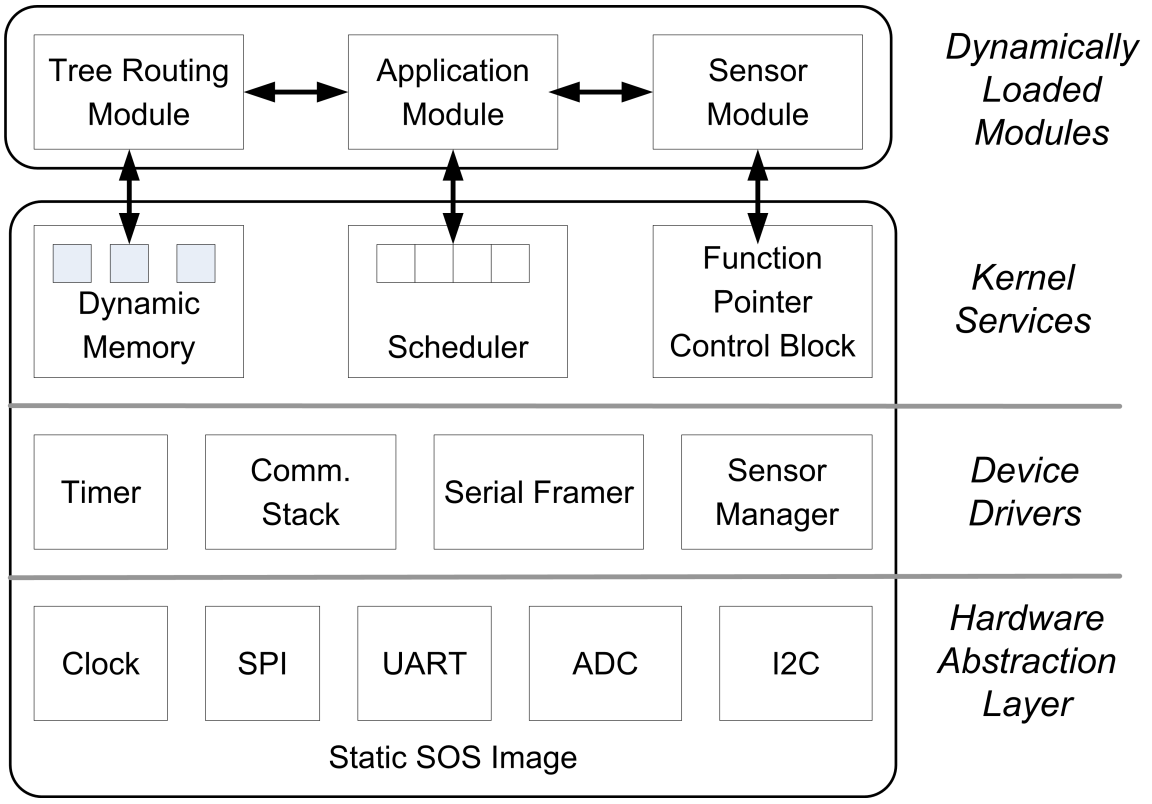



Figure 2.5: The figure shows the layered architecture of the SOS operating system from the University of California, Los Angeles. The system can be divided into two main parts: 1) the set of dynamically loaded modules and 2) the static operating system image. Latter provides kernel- and device driver-services and encapsulates the hardware access. The modules enter the kernel through the function pointer control block. Their execution is controlled by the event scheduler. The modules may implement libraries and application functionality. A distinctive feature of SOS is the extensive run time support for module loading, updating and management. The dynamic memory management - for example - has a notion of ownership and limited support for garbage collection to facilitate dynamic memory usage among modules. The figure is based on [29].

During the assembly process all referenced software modules are combined into a single source file. This source file is subject to whole program analysis and optimization by the compiler. Small functions are inlined which saves code space and increases performance[15]. Unused functions are completely eliminated. Thus TinyOS yields the smallest possible system code size. A drawback is that it does not support code sharing among run time loaded modules.

Memory allocation is generally static in TinyOS. This prevents memory fragmentation and allocation failures. Static memory allocation also benefits compiler code generation and checks. TinyOS probably has the smallest working memory footprint due to the whole system optimization and customization.

TinyOS [54] has a 3-layer hardware abstraction layer which caters for performance and portability: the first layer (HPL = Hardware Presentation Layer) deals with the low level access to memory- or port mapped hardware interfaces and is used by the device drivers in the second layer.

The second layer (HAL = Hardware Adaptation Layer) encapsulates the devices, arbitrates the access, manages the configuration, maintains device state machines and is *power-aware*.

An example: on platforms with low transition costs between sleep- and active-mode the scheduler puts the system aggressively into sleep mode. If the transition costs are higher the scheduler only transitions the system to sleep mode if no outstanding tasks are queued up for the near future.

The third layer (HIL = Hardware Interface layer) implements hardware independent interfaces (not tailored to a specific device) for cross-platform application components. HIL functionality may partially be implemented in software depending on the capabilities of the underlying hardware. Due to the higher abstraction it is likely that not all features of the underlying device are accessible.

An SOS system [53, 29] (see Figure 2.5) consists of a kernel image plus a set of one or more dynamically loaded modules. The static system image facilitates code sharing and hosts kernel- and device driver-services. It also encapsulates the hardware access. The dynamically loaded modules run on top of the system image. They enter the kernel through a jump table (function pointer control block).

*The modularization concept of SOS allows incremental software updates. This saves energy if updates are conducted frequently* [53].

Contiki [40] has a fixed core with a program loader and (optional) libraries which are combined into a static system image. The static system image can be extended with dynamically loadable modules [39] - like in SOS. The kernel does *not* provide a hardware abstraction layer [40]. The application and device drivers must therefore interface directly with the hardware. *Consequently, there is no (system-defined) power management support in the Contiki core.*

[15] Very small functions may suffer from a high overhead in code size due to the obligatory function pro- and epilogue. The TinyOS programming model suggests many small functions.

The Squawk virtual machine [129, 130]implements an interpreting Java virtual machine and does not require an underlying operating system. The system consists of a bytecode interpreter, program loader, (bytecode) verifier, a transformer (bytecode optimizer), garbage collector and class libraries. The authors of Squawk (for the hardware platform please see pages 15 ff) have adapted the virtual machine to the special requirements of sensor nodes:

Squawk has an optimized program format called *suites.* Suites use a denser bytecode format than the one used in the standard virtual machine. They are also *executable-in-place* which leads to a reduced memory footprint.

The thread scheduler in Squawk attempts to save power by switching the system into shallow- or deep-sleep mode whenever possible. Both modes vary in power consumption and wake up time. An application may override the schedulers sleep policy.

A rather unique feature of Squawk is the ability to support multiple *isolated* applications on a single node. An application state can be frozen and transferred over the network for debugging purposes or change of location.

*The benefits of running in a managed environment (virtual machine) comes at a high cost for Squawk:*

The virtual machine requires a powerful and expensive hardware platform to get up and running (please see the table on page 167 for a comparison). Another serious drawback of the Squawk virtual machine is the interrupt latency which is lowest on an idle system and can reach up to 13 milliseconds if a garbage collection occurs (100kb heap). Surprisingly, the Squawk-VM does not implement real time scheduling and garbage collection. The Java community has standards and implementations for both.

Squawk can run on a PC platform since it is written in the portable Java language. On a PC a few hardware sensors are available in simulation but no power consumption simulation.

TinyOS has its own simulator (Power)-TOSSIM [123, 88]. (Power)-TOSSIM substitutes hardware specific components for simulated components and offers power estimation support (see page 46). There is no specific (power) simulator for SOS but the Avrora full system simulator [134] is capable of running SOS and TinyOS systems. Avrora has a built-in power model for several common sensor node hardware components [81] (see page 43).

### 2.4.2 Execution and Communication Models

All four run time platforms (TinyOS, SOS, Contiki, Squawk [53, 29, 129, 69, 87, 54]) have their own execution- and communication models:

Squawk and Contiki (latter with an optional addition) support the traditional thread programming model. To keep the implementation simple it is not possible to preempt the Squawk system code. Interrupts in Squawk are polled from the main event loop.

*Contiki runs always with interrupts enabled and provides preemptive thread switching which makes it suitable for real time applications.*

In TinyOS interrupts may issue asynchronous function calls. In the TinyOS terminology they are referred to as tasks. Tasks in TinyOS run to completion and can only be temporarily preempted by interrupts. The division in short running, time critical interrupt handlers and longer running, non-critical tasks is the way TinyOS tries to ensure a responsive system behavior.

*Conceptually - however - there is no hard guarantee for real-time responsiveness in TinyOS.*

In SOS the kernel dispatches events posted by interrupt handlers or modules to the respective module event handlers. The scheduler dequeues events from either the default or the prioritized FIFO-queue.

*Thus SOS can distinguish between critical and non-critical events but has no preemptive task scheduling*[16] *and is therefore not real time capable.*

In SOS the inter-module communication can either be asynchronous or synchronous. Synchronous communication is realized by function calls. Contiki supports asynchronous message passing and synchronous communication between processes by scheduling them synchronously.

In TinyOS inter-module communication is realized by function calls which may post tasks to a system wide FIFO queue. This is usually done for long(er) running operations. Task completion is signaled over asynchronous callbacks. TinyOS does not support asynchronous message passing like Squawk. In Squawk synchronous inter-module communication is realized by function calls like in TinyOS.

### 2.4.3 Componentization

TinyOS uses the NesC programming language [51] which knows two constructs: *modules* and *configurations*:

Modules contain components which consist of code, data and and interface specifications. A TinyOS component provides interfaces for its users and specifies those it needs. Interfaces in TinyOS are bi-directional.

A TinyOS configuration encompasses a design time component wiring specification. The composition of components is aligned with the TinyOS concurrency model. Race conditions in component configurations are reported by the compiler - granted that the individual component specifications are correct.

Both, SOS and Contiki support run time loadable (binary) modules [39, 29] but they do not have a formal component model like TinyOS. The dynamic loading and unloading of (binary) modules - in SOS and Contiki - may cause various fault conditions. Therefore the SOS run

[16] SOS uses *cooperative* multi-tasking.

time has link-time type checks, supervises message delivery and tracks dynamically allocated memory resources to prevent the manifestation of errors.

Squawk is based on Java which supports source code level interface specifications like NesC. Additionally, Squawk supports binary application modules (called suites). They have a binary format and are run time loadable. Suites in Squawk, can be digitally signed for increased trustworthiness.

Apart from the obvious engineering advantages of component based systems they also facilitate novel ways of power simulation. In TinyOS - for example - the hardware specific components can be substituted with mock-ups which simulate the hardware functionality on a personal computer ( [88, 123] see page 46).

Generally, it can be noted that the well defined boundaries of components lend themselves to a *per-component power break-down* in simulation (power estimation) and prototypes (measurement).

## 2.5 Model based Power-Analysis and Estimation

The design space exploration of a sensor node platform (hardware and software) requires a careful power consumption analysis. It is crucial that the designer understands the power costs incurred in all components. A power break-down into *hardware components* can be obtained by estimation (see section 2.7 or 2.8) -measurement.

The power consumption exhibited by an *application* may be hard to follow even when a break-down by hardware component is available. Quanto is a concept from UC Berkeley and will be introduced in the following section. In Quanto software may use labels to track power consumption on a high level.

The Hogthrob project from Denmark uses per component power consumption- and application specific-utilization data to predict the performance of an application on another platform. Its analytic model will be introduced after Quanto.

### 2.5.1 Quanto

In [48] Rodrigo Fonseca (UC Berkeley) and his colleges from Stanford describe an energy profiling methodology called Quanto which works on the node- and network level. A major motivation - according to the authors - was to discover "energy-leaks" occurring *during deployment*. In a prior deployment [136] (logging environmental data in redwood trees) the authors had noticed that 15% of the nodes failed within a week; while the remaining ones kept running for months. The exact cause could not be determined in the aftermath.

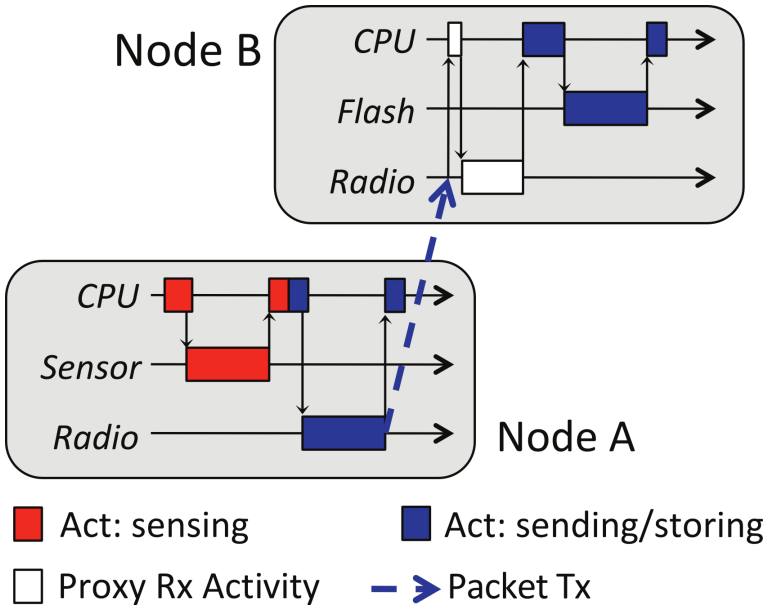



Figure 2.6: Activity labels - The software designer can associate *high-level* tasks with activity labels such as **sensing** and **sending/storing**. The activity-log can later on be combined with the power-log to map consumed time and power to specific activities. This activity power break-down is necessary to identify complex system behavior more easily. In this example the activity **sensing** on Node A is followed by activity **sending/storing** which extends to Node B. The so called *proxy-activity* on Node B is attributed to Node A's **sending/storing** activity after the activity label has been retrieved from the received packet.

### Conceptual Energy Break-Down

Quanto collects time stamped *power measurements* along with their corresponding *activity* type. This data is later on used to *estimate* the power consumption of each power state for every hardware component. The assignment of activities is necessary to disambiguate the total power consumption on the application level.

The so called *activity labels* must be attached to data structures. Examples are job queues and network packets. Latter lets Quanto extend the power tracking to the network-level. The receiving nodes attribute the power consumption caused by an incoming packet to the activity specified in the attached label. Even the energy which had been necessary to receive the incoming packet is considered. Activities whose attribution must be deferred are so called proxy-activities (see Figure 2.6).

In Quanto the application designer models the assignment of *activity labels* as needed. The underlying operating system[17] is responsible for the correct *propagation* of activity labels. The authors identified four cases where propagation is necessary:

- activity across devices (CPU runs code which controls the radio)
- activity across nodes (packets are labeled with activity- and node identifier)
- bind proxy activities (deferred attribution of energy/time - see Figure 2.6)
- following activities over (time) / interleaved execution flow within the operating system

The activity labeling, forwarding and logging of Quanto is intrusive. It requires changes to the underlying operating system and to (existing) applications. The logging requires CPU time, memory (for additional code- and data) and therefore consumes additional energy during deployment[18]. For the tested (trivial) application the Quanto power profiler was responsible for 71% of the CPU *active* time. It remains to be seen how Quanto performs in non-trivial applications since all benchmarks in the publication had been *micro-benchmarks*.

In this thesis a similar methodology has been developed: the full system (application and system-software) may be broken-down into *categories* (similar to activities). Categories may be routing and signal processing for example. Unlike Quanto categories are tracked along the *call chain*. Upon function entry a new category *may* become active which would lead to a restoration of the prior category (state) upon return. Quanto lacks the *flexibility* of *hierarchical* tracking[19]. It *overrides* an active activity unless the state is preserved within a data structure. Thus state recovery - in Quanto - must be done explicitly by annotating data structures like radio packets with activity labels.

[17] In this case the operating system is TinyOS.

[18] Radio power must be added if the log is transmitted wirelessly.

[19] Hierarchy allows a much greater degree of freedom in modeling. Quanto could not take this approach because it runs on low-end sensor nodes (hardware constrains) and must not use too much of the sparse system resources. This is especially true if Quanto has to profile the network during a (potentially) long running deployment. Quanto's activity overriding policy may be a good fit for the specific run-time model of TinyOS (on page 24) which is the operating system the authors had extended.

*Data structure annotations are intrusive but they also let Quanto track activities across sensor nodes.*

In this thesis a single activity cannot be tracked across nodes but *categories* may be chosen in a way which allows the identification of external stimuli. Unlike Quanto my scheme allows each node to have different categories since the mapping of categories to application- and system functions is not part of the application / system source code but separated out in a machine readable specification file.

*Thus compared to Quanto my scheme does not require source code changes.*

Obviously, Quanto's and my methodology have different design objectives and constrains.

Quanto is an online profiling methodology for basic sensor nodes which allows the designer to analyze the power consumption behavior of the network after deployment. My work is focused on the *design space exploration* of (advanced) sensor nodes.

Both platforms can assess the power consumption of individual components. The Hyperion hardware platform can measure the individual power consumption of each component. Quanto requires only that the total power consumption is measured. After a sufficient run time Quanto estimates the individual power consumption.

### Power Estimation in Quanto

As mentioned before: Quanto relies only on *total system power measurement* to estimate the power consumption of each component. The power measurement setup of Quanto is based on iCount [42] and is covered separately in section 2.7.

The underling assumption in Quanto is that each hardware component has a fixed number of *observable power states* with a *constant power draw*.

Even seemingly small and simple components like an ultra-low power radio transceiver or a microcontroller already exhibit a large number of *power modes*: the microcontroller in [48] for example, has 16 power modes (e.g. CPU 6 modes, D/A-converter 3 modes) which may be combined to even more *power states*.

Quanto tracks the energy $\triangle E$ and time spent $\triangle t$ in the power states $i$=0..n. For every (stable) interval Quanto *generates* a linear equation of the form:

$$\triangle E = \triangle t \sum_{i=0}^{n} \alpha_i p_i \tag{2.3}$$

The binary variable $\alpha_i$ encodes the $i$-th power state (on / off) and $p_i$ is the corresponding power draw. The assumption is that after a sufficiently long run time Quanto has enough equations to determine the *unknown* power draw of each hardware component[20].

[20] The attribution to the previously covered *activities* is a separate step and unrelated to the power estimation itself.

In their publication [48] the authors formulate a *multivariate* regression equation and solve it using the *ordinary least squares* (OLS) technique. In [48] the regression problem is neither over- nor under-determined[21]. The regression equation can be written as:

$$\vec{Y} = f(\underline{X}, \vec{\beta})$$

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_j x_{ij} + \ldots + \beta_p x_{ip} + \varepsilon_i, \quad \begin{cases} i = 0..n & \text{\#equations} \\ j = 0..p & \text{\#variables} \\ n = p & : \text{in Quanto} \end{cases}$$

where, the $x$ are the independent variables (*power modes*), $\beta$ the unknown parameters (*power draw*), $Y_i$ the dependent variables (*total power measured*) and $\varepsilon$ the error-term. For a multivariate formulation we need the following vectors and matrices:

$$\vec{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_i \\ \vdots \\ Y_n \end{pmatrix}, \underline{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & & \ddots & \vdots \\ x_{i1} & x_{i2} & \ldots & x_{ij} & \cdots & x_{ip} \\ \vdots & \vdots & \ddots & & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nj} & \cdots & x_{np} \end{pmatrix}, \vec{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_j \\ \vdots \\ \beta_p \end{pmatrix}, \vec{\varepsilon} = \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_i \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

($\underline{X}$ is also called the design matrix. )

Now, the regression equation can be written as:

$$\vec{Y} = \underline{X}\vec{\beta} + \vec{\varepsilon}$$

Using a least-square estimator $\vec{b}$ for $\vec{\beta}$ the equation becomes:

$$\vec{Y} = \underline{X}\vec{b} + \vec{r} = \hat{\vec{Y}} + \vec{r} \tag{2.4}$$

where $\vec{r}$ is the residual $\vec{r} = \vec{Y} - \hat{\vec{Y}}$. If the residuals sum of squares is minimized using the *method of least squares* then the solution is

$$\vec{b} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_j \\ \vdots \\ b_p \end{pmatrix} = (\underline{X}^T \underline{X})^{-1} \underline{X}^T \vec{Y}$$

[21] The number of unknown variables and equations is always the same because the authors average all measurements for each power state. The assumption is of course that sufficient measurements have been carried out.

In their publication the authors suggest that quantization effects are bigger for short time intervals $\triangle t$ and little energy consumption $\triangle E$. This variance of the error[22] in the observations (power measurements) can be accounted for in the *weighted least squares* approach. The authors suggest a weight (approximation) of $w_j = \sqrt{\overline{E}_j \overline{t}_j}$. Where $\overline{E}_j$ and $\overline{t}_j$ are the average energy and time consumed in a power state $j$.

The weight matrix is:

$$\underline{W} = \begin{pmatrix} w_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & \ddots & \vdots \\ 0 & 0 & \dots & w_j & \cdots & 0 \\ \vdots & \vdots & \ddots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & x_p \end{pmatrix}$$

Which gives us a weighted least squares solution:

$$\overrightarrow{b} = (\underline{X}^T \underline{W X})^{-1} \underline{X}^T \underline{W} \overrightarrow{Y}$$

The authors did not discuss whether the weighting increased the accuracy. The results given in the publication did not use weights. Nevertheless, the results which had been obtained for a trivial application (lighting three different LEDs!) are a quite accurate estimation of the real power consumption.

The methodology provides a break-down into total component power consumption and also per activity. Activities may span multiple hardware components over a prolonged time.

Quanto is a promising approach for run time power profiling of basic sensor nodes.

*The assumption that (all) hardware-state changes are observable and exhibit a constant power draw is difficult to maintain for advanced sensor nodes.* Non-trivial, power efficient processors dynamically adapt to temperature constrains [89] and their workloads. David Albonesi - for example - presented a publication [7] which looks into automatic (fine-grained) tuning of microprocessor resources such as queues and caches. Commercial processors like [82] already gate more than 50% of their latches dynamically.

Another crucial pre-condition - pointed out by the authors - is that the component power draw must result into linearly independent equations. This requires that two hardware components may not always switch simultaneously during profiling.

[22] The errors must still be uncorrelated.

| Name | Type | Example |
|---|---|---|
| $\overrightarrow{MV_e}$ | Energy (consumption) *system* vector | $\overrightarrow{MV_e}=\begin{bmatrix} e_{\text{active}} \\ e_{\text{idle}} \\ e_{\text{adc}} \\ e_{\text{tx}} \end{bmatrix}$ |
| $\overrightarrow{MV_t}$ | Execution-time *system* vector | $\overrightarrow{MV_t}=\begin{bmatrix} t_{\text{active}} \\ t_{\text{idle}} \\ t_{\text{adc}} \\ t_{\text{tx}} \end{bmatrix}$ |
| $\overrightarrow{AV}$ | *Application* vector (utilization of system primitives) | $\overrightarrow{AV}=\begin{bmatrix} \text{active} \\ \text{idle} \\ \text{adc} \\ \text{tx} \end{bmatrix}$ |
| $\overrightarrow{C}$ | CPU time needed by peripherals | $\overrightarrow{C}=\begin{bmatrix} C_{\text{adc}} \\ C_{\text{tx}} \end{bmatrix}$ |

Table 2.1: System and architecture characterization vectors of the vector power estimation method. $\overrightarrow{MV}$ stands for "mote"-vector - a different term for sensor node commonly used in the community.

### 2.5.2 Vector based Power Estimation Methodology

In his PhD thesis [85] and in [86] Martin Leopold (the research had been done under aegis of the Hogthrob project, see page 18) applied the concepts of a performance analysis technique [116] to a basic sensor node model (a microcontroller with A/D-converter plus narrow-band radio transceiver). The key idea is to characterize the system and the application independently. This allows an application vector to be mapped to different architectures in order to obtain performance estimates without running experiments or simulations.

Martin and his co-authors introduce the *system- and application-characterization* vectors which are shown in Table 2.1.

The energy and execution-time of a sensor node can then be expressed as:

$$\begin{aligned} \text{Energy} &= \overrightarrow{MV_e} \cdot \overrightarrow{AV} \\ \text{ExecutionTime} &= \overrightarrow{MV_t} \cdot \overrightarrow{AV} \end{aligned}$$

Table 2.2: Energy and execution time can be derived if the system- and application vectors are given

For $\overrightarrow{MV_e}$ and $\overrightarrow{MV_t}$ micro-benchmarks must be devised that measure the energy and time needed

for certain primitive operations (e.g. sending, receiving, sampling, processing). For a simple node these operations can be CPU *active*, CPU *idle*, transmit *(tx)* or A/D-conversions (*adc*). Micro-benchmarks must be carried out to obtain these system vector components: $\overrightarrow{MV_e}[e_{\text{active}}]$ … $\overrightarrow{MV_e}[e_{tx}]$ and $\overrightarrow{MV_t}[t_{active}]$ … $\overrightarrow{MV_t}[t_{tx}]$. Additionally - the CPU time ($\overrightarrow{C}[X]$) required by CPU dependent operations (*adc/tx* in this example) - must be measured so that it can be factored out later on. Here, $\overrightarrow{C}[\text{adc}]$ and $\overrightarrow{C}[\text{tx}]$ represent the CPU-time required to issue a single A/D-conversion or packet transmission.

Martin introduces the following vectors and matrices to determine the application vector $\overrightarrow{AV}$ (see the table on the next page) : The application trace vector $\overrightarrow{T}$ is generated from an application sampled over a $\Delta t$ and records the time spent in certain *states* (see the example vector of $\overrightarrow{T}$). A multiplication with the *architecture matrix* **AM** results in a preliminary $\overrightarrow{TT}$ (total) time-vector where the CPU time of the peripherals is not yet factored out.

In the example it is assumed that an A/D-conversion and a transmit do not happen simultaneously[23]. Furthermore it is assumed that the time taken in states with A/D-conversion and a transmit have an equal time share. Thus some of the matrix elements in **AM** are $\frac{1}{2}$ - in the example - to account for such a (specific) architecture.

The **CPU** matrix exists - as mentioned before - to subtract the CPU usage of other components (e.g. software driver CPU overhead to control A/D-conversion or issue a transmission) from the (CPU) *active* vector element since it is *not independent* in this example. Thus the product of $(\mathbf{AM} \times T)$ (=preliminary $\overrightarrow{TT}$ vector) with the **CPU** matrix yields the final $\overrightarrow{TT}$ vector where the *(total) active* element only represents the time spent for processing, and not for controlling peripherals. This distinction is necessary to map the application vector to architectures where no or a different CPU time is necessary to drive those peripherals.

The $\overrightarrow{AV}$ (application vector) is then derived by dividing the $\overrightarrow{TT}$ elements entrywise with the execution-time system vector $\overrightarrow{MV_t}$ elements to obtain the frequency of the system primitives in the application (see Equation 2.5).

$$\begin{aligned} \overrightarrow{TT} &= \mathbf{CPU} \times (\mathbf{AM} \times \overrightarrow{T}) \\ \overrightarrow{AV} &= \overrightarrow{TT} \diamond \overrightarrow{MV_t}^{-1} \quad (a) \\ &= \left(\mathbf{CPU} \times (\mathbf{AM} \times \overrightarrow{T})\right) \diamond \overrightarrow{MV_t}^{-1} \end{aligned} \tag{2.5}$$

$a$) Hadamard multiplication = entrywise product

The energy can be derived from the equation $\overrightarrow{MV_e} \cdot \overrightarrow{AV}$ introduced on the facing page. The power consumption would subsequently be

[23] Due to a shared bus for example.

| Name | Type | Example |
|---|---|---|
| $\overrightarrow{T}$ | Trace Vector | $\overrightarrow{T} = \begin{bmatrix} \text{active} \\ \text{idle} \\ \text{adc} \\ \text{tx} \\ \text{adc \& tx} \\ \text{active \& adc} \\ \text{active \& tx} \\ \text{active \& adc \& tx} \end{bmatrix}$ |
| **AM** | Architecture Matrix | $\mathbf{AM} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{1}{2} & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & \frac{1}{2} & 0 & 1 & \frac{1}{2} \end{bmatrix}$ |
| **CPU** | CPU Matrix | $\mathbf{CPU} = \begin{bmatrix} 1 & 0 & -\left(\frac{1}{\overrightarrow{MV_t}[t_{adc}]} \cdot \overrightarrow{C}[adc]\right) & -\left(\frac{1}{\overrightarrow{MV_t}[t_{tx}]} \cdot \overrightarrow{C}[\text{tx}]\right) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ a) |
| $\overrightarrow{TT}$ | Total Time Vector | $\overrightarrow{TT} = \begin{bmatrix} \text{total active} \\ \text{total idle} \\ \text{total adc} \\ \text{total tx} \end{bmatrix}$ |

a) $-\left(\frac{1}{\overrightarrow{MV_t}[X]} \cdot \overrightarrow{C}[\text{X}]\right)$ The peripherals time is divided by $\overrightarrow{MV_t}[X]$ (time for a single operation X) which yields the frequency of the operation. Then $\overrightarrow{C}[X]$ - the CPU-time necessary for operation X - is multiplied. The result is the amount of CPU-time needed by a certain system component, e.g. A/D-converter or radio. Marcus Chang a co-author kindly clarified this.

Table 2.3: Description of the vectors and matrices introduced by the vector performance estimation method

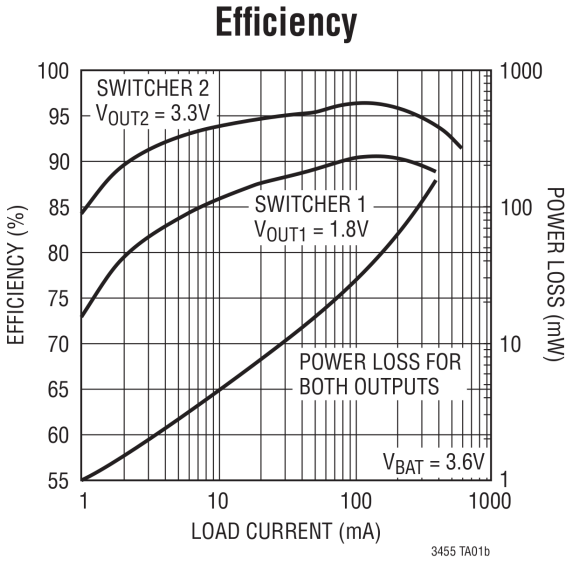



Figure 2.7: Energy efficiency of SunSPOT's switching regulator. An example of a non-linear hardware component. Source: Linear Technology LTC3455/LTC3455-1 [31]

$$P = \frac{\text{Energy}}{\text{Time}} = \frac{\overrightarrow{MV_e} \cdot \overrightarrow{AV}}{\overrightarrow{MV_t} \cdot \overrightarrow{AV}} \qquad (2.6)$$

where $\overrightarrow{MV_e}$ and $\overrightarrow{MV_t}$ can be vectors from an architecture of interest. Equation 2.6 has been derived in this thesis.

The difficulty in the vector methodology is to find appropriate micro-benchmarks for system primitives and traces which sufficiently represent the chosen workload(s). A more accurate model needs also to consider the transition time and power costs when components change their state (e.g. various power states in a microcontroller). The crucial point - however - is that the model is linear and the real world often is not:

The SunSPOT introduced on page 15 - for example - uses a switching regulator for power supply. Figure 2.7 shows the power loss of the two regulated outputs for a given load (logarithmic scale).

It is obvious that the relation between load and power consumption (loss) is highly non-linear in this case. Another example - given by the author(s) - is that the CPU performance is also linearly mapped which may be problematic in some cases. The methodology certainly has its place but it remains to be seen how and where it can be applied successfully to non-trivial sensor nodes.

Furthermore, a break-down of a state-event trace is required to compute the utilization of in-

dividual system components which may be difficult or impossible to achieve as well[24].

In this thesis the power consumption break-down is either obtained by measurement with the hardware driven design space exploration platform or by estimation in the software driven platform.

[24] The example presented earlier assumed that either a radio transmission or an A/D-conversion can be carried out because of a shared bus. Furthermore, an assumption - made in the architecture matrix - is that both events are equally likely which may not be the case.

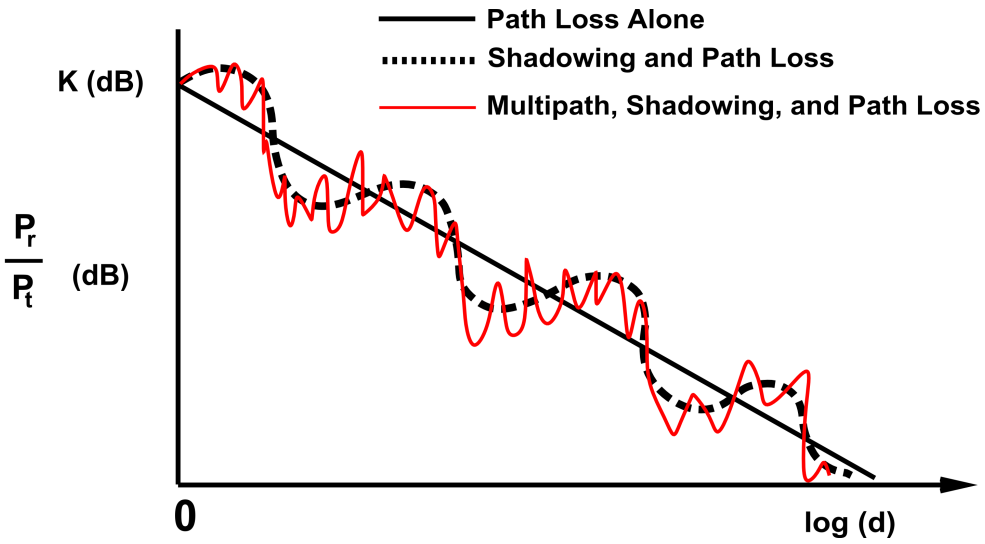



Figure 2.8: The figure shows the path loss as a function of the log-distance ($d$). The path loss is the relation between received- ($P_r$)and transmitted ($P_t$) - power. $K$ is the path loss at a reference distance. It can be seen that the path loss depends on the distance and shadowing effects. The constructive and destructive multi-path components are superimposed on the path loss. The figure is based on [52].

## 2.6 Challenges in Modeling Radio-Communication

In wireless sensor networks radio communication is prevalent. Unlike wired communication systems the wireless medium is much less predictable. The imperfections of the link-layer has various implications on the performance of the upper layers [154]. Simulation results based on the idealized disc shape model are not considered to be particularly helpful [34, 13, 155] nowadays. The following section sketches the challenges and implications surrounding radio communication. The section after the following one discusses influential publications from the wireless sensor network community and their impact on this thesis.

### 2.6.1 Background

Radio communication is subject to noise and interference. The signal power varies over time and with location. Distortions are rooted in environment dynamics and imperfections within the radio:

Radio output power may be influenced by variations in production and supply voltage [154]. Another source of variability are antennas. They transmit and receive electromagnetic waves generated by a radio. The antennas interact with the radio waves which propagate three dimensionally through space. The radiation patterns depend on the characteristics of the antenna. Antennas are directional, vary in efficiency, bandwidth, polarization and (effective) aperture[25].

[25] Power captured from a plane wave.

In the environment the signal power is attenuated by absorption (shadowing, reflection, scattering, diffraction) and susceptible to interference (constructive and destructive signal superposition - see Figure 2.8). Latter is referred to as *small scale* propagation effects because the involved distances are in the order of wave lengths. The other effects (e.g. shadowing) are called *large scale* effects because they involve comparatively large distances.

Due to the complexity of radio wave propagation different modeling techniques are used by themselves or in combination [52, 111, 80]: analytic models (free-space path loss model), ray-tracing models (Ten-Ray model [8], general ray tracing) and empirical models (log-normal shadowing path loss model , Okumura model [131], Hata model [55]).

The log-normal shadowing path loss model (Equation 2.7) - for example - has been successfully used to approximate the *wireless channel* in a simple sensor node experiment [155, 93] - (a radio model is also required). The experiment involved 21 Mica2 sensor nodes in a string topology with one meter spacing. A TDMA protocol was used to avoid collisions since the model does *not* support interference.

$$L_{\text{Path}} = P_{tx}[\text{dBm}] - P_{rx}[\text{dBm}] = K + 10\gamma \log_{10}\frac{d}{d_0} + X_\sigma \begin{cases} K & \text{path loss at reference distance} \\ \gamma & \text{path loss exponent (rate of decay)} \\ d & \text{length of path} \\ d_0 & \text{reference distance} \\ X_\sigma & \text{random variable, zero mean, Gaussian} \end{cases} \tag{2.7}$$

Furthermore, the empirical model derived in [155] is only valid for the frequency of the Mica2 transceiver and the surroundings where the experiment was conducted. *This may sound quite limiting but prior to the Castalia simulator [104, 13] much simpler models were common place in the wireless sensor network community.*

If the environment contains blocking and scattering objects whose dielectric properties are unknown then statistical models like the log-normal model may be suitable [52]. If elevation- and material-data exist then general ray tracing is able to give insights into *application specific* scenarios.

Commercial and academic software tools for network planning (see Figure 2.9 and 2.10) use 3D-ray tracing along with antenna models and terrain data to accurately estimate the signal propagation (for example NIR-Plan for Cellular from Hexagon System Engineering). Such tools would be interesting if they could be extended to include the specialties of sensor networks.

Recent advances in chip multi-core architectures may enable acceptable simulation speeds in the future. Real time ray tracing in computer vision is already nearing[26].

*For the time being wireless sensor network researchers must be aware of the limitations of the underlying radio- and channel models and their (possible) implications on the higher layers.*

[26] NVIDIA® OptiX™ ray tracing engine / SIGGRAPH 2009

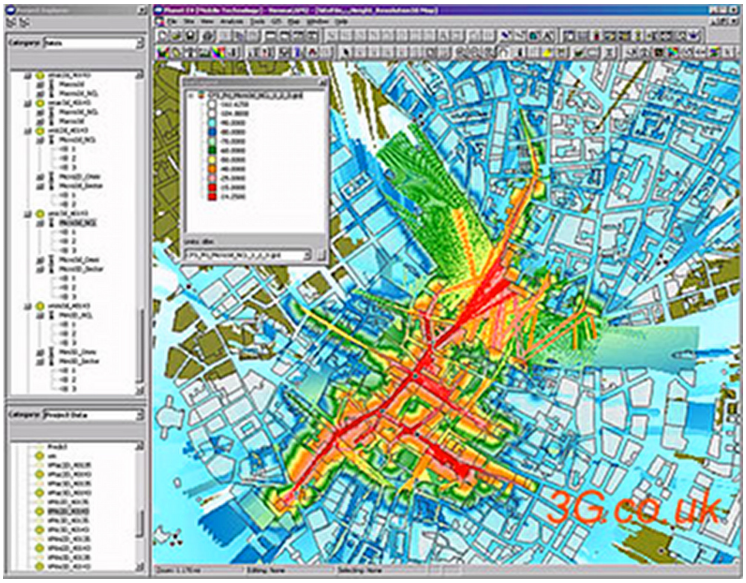



Figure 2.9: A screen shot of a commercial (wireless) network planning software. Source: 3g.co.uk

Alternatively, real deployments can be carried out. Colleagues of mine [21, 22], and I have conducted experiments in the field with the support from students and found that handling 40-60 sensor nodes is still manageable. However, it is advisable to plan the field tests carefully and integrate rich introspection capabilities into the sensor node soft- and hardware.

### 2.6.2 Findings and Modeling Expertise in the WSN Community

In 2002 a large scale study with 156 sensor node was conducted at the Center for Embedded Networked Sensing (CENS) in California[27]. The authors [35] discovered various irregularities in wireless communication. Figure 2.11 - for example - shows the contours of packet reception probability as seen by the center node of a large sensor node grid. Clearly, the probability distribution is directional. Furthermore, the authors noticed *packet loss* over short distances and *asymmetric links*. They came to the conclusion that environmental changes and small deviations in the communication system may have a strong impact on a large scale, low power sensor network. Especially, the prevalence of asymmetric links was emphasized in the publication. The advice for protocol engineers was therefore to cater for asymmetric links and to be aware that circular propagation models are insufficient.

Based on the initial findings a follow-up CENS publication [6] presented a tool and quantitative results of a 55 sensor node network which had been deployed in a park landscape, on

[27] CENS is located at the University of California, Los Angeles (UCLA). Besides UCLA, CENS has the following founding members: UC Merced, UC Riverside, University of Southern California and the California Institute of Technology (Caltech). *"CENS is a major research enterprise focused on developing wireless sensing systems..."* `http://research.cens.ucla.edu/about/`

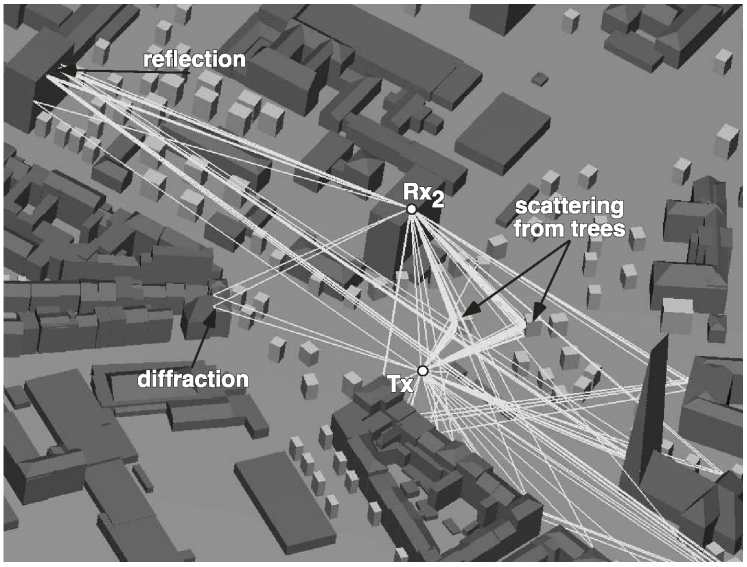



Figure 2.10: The 80 strongest propagation paths between a mobile transmitter and a roof-top positioned base station are shown. The 3D-raytracing model has been developed by the IHE at the KIT [49, 50].

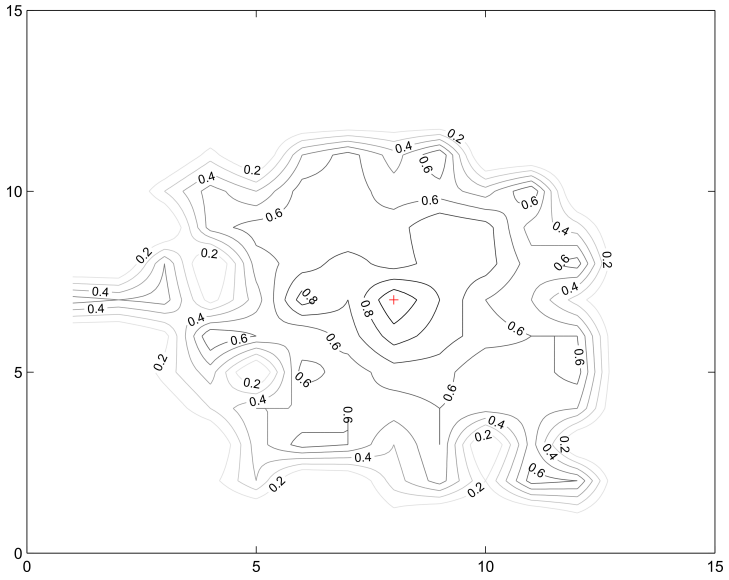



Figure 2.11: This figure shows the contours of packet reception probability experienced by a base node centered in a 13x12 sensor node grid with 2 feet spacing (≈ 61cm) [35]

campus and indoors. They found no correlation between packet reception and distance within 50% of the transmission range. Additionally, no correlation in temporal variation in regard to distance or transmission power which partially applied to asymmetric links too. Asymmetric links accounted for 5 to 30 percent of the total number of links. According to the authors results, asymmetry is likely to be caused by variations in the radio hardware and the power supply.

Inspired by these publications the authors of [154] proposed an empirical radio model for simulation which factors in: 1) non-isotropic radio ranges, 2) continuous signal power changes with incremental changes in direction and 3) heterogeneity in transmission power. By simulation the authors found that routing protocols are *significantly* influenced by radio irregularities whereas MAC-layer protocols seemed to suffer less. Localization protocols were assumed to suffer as well but had not been studied.

In the same year David Kotz from Dartmouth College strongly criticized six assumptions often made in simulation which lead to *significant differences* between simulation and experimental results [34, 79]: (1) flat world, (2) circular transmission range, (3) symmetry, (4) perfect reception within range and (5) signal strength as a function of distance. David's publication contains simulation results and measurements to back his claims.

Based on the findings of [35, 6, 153, 154, 34] Marco Zuniga[28] from the University of Southern California applied information theory to characterize the "*transitional region*" in radio communication of low power sensor networks [155, 93]. *In the transitional region highly unreliable connections are encountered.* The *connected* and *disconnected* regions - for comparison - have perfect or no connectivity at all.

Zuniga stresses that *two* models are required to capture the behavior of radio communication: *a channel model and a radio model.* For the channel model Zuniga uses the log-normal shadowing model (see Equation 2.7 on page 38) where some of the parameters must be determined empirically by curve fitting[29]. The radio reception model is based on a Bernoulli random variable where the probability depends on the modulation scheme (e.g. FSK = frequency shift keying for the Mica2). The equation for the packet reception rate is:

$$p = (1 - \frac{1}{2}\exp^{-\frac{\alpha}{2}})^{8f} \quad \begin{cases} \alpha & \text{see equation below} \\ f & \text{frame size (packet,payload,crc)} \end{cases}$$

$$SNR(\gamma) = \alpha\frac{R}{B_N} \quad \begin{cases} \gamma & \text{derived from the RSSI} \\ R & \text{data rate} \\ B_N & \text{noise bandwidth} \end{cases}$$

The RSSI (receiver signal strength indicator) is (usually) provided for each received packet by the radio transceiver. The remaining parameters depend on the radio and can be derived from a data sheet or configured at run time.

[28] Both Marco and David mention the cellular telecommunication community in respect to their more advanced radio models and rich literature.

[29] $K$ - path loss at reference distance and $X_\sigma$ Gaussian random variable with zero mean (shadowing effects)

The authors combined the channel and radio model to formulate a transitional region coefficient Γ which describes ratio of the radius of the transitional- and connected-region. Thus Γ can be considered a link quality indicator for different environments. The path loss exponent $K$ and the (shadowing) standard deviation $\sigma$ have a strong influence on Γ (see Equation 2.7 on page 38 for the meaning $K$ and $\sigma$) whereas the frame size and encoding have a lesser impact.

One of the most important findings - according to the authors - is that even an ideal radio receiver in a real environment has a *transitional* region where random values of zeroes and ones are possible.

*Thus the perfect reception within-range-model does not apply - even in this (idealized) case.*

Zuniga's influential paper inspired Athanassios Boulis (National ICT Australia (NICTA) to integrate the proposed models into Castalia. Castalia is an OmNet simulator extension [139] which raises the state of the art of wireless sensor network simulation [13].

Although the tools of the wireless sensor network community have started to improve it remains a necessity to evaluate the ground truth. Modeling wireless radio communication is challenging and many parameters and variables are involved for environment-, radio- and antenna modeling for example.

In this thesis a design space exploration platform has been devised. For accurate measurement results a hardware-in-the-loop approach has been chosen. The hardware driven design space exploration platform *Hyperion* performs real time hardware simulation of the sensor node system-on-chip (SoC) and interfaces with *real* and *plugable* radio transceivers. Thus the inexplicable challenges of communication simulation have been avoided at the cost of a more involved experimental setup. Additionally, the software driven design space exploration platform *Sensim* provides a bit accurate simulation of the radio interface and (plugable) models for radio communication simulation.

## 2.7 Assessment of Power Consumption by Simulation

In order to explore the design space of a sensor node platform it is crucial to assess the power consumption accurately. This can be done by simulation and measurement. Power measurement requires the hardware or at least parts of it to be built. The time needed to build the hardware is lost for design space exploration. Therefore simulation is often applied to obtain estimates. Ideally simulation is fast and accurate so that many experiments with varying parameters can be conducted at a small cost and with little effort. Unfortunately, simulation is often less perfect than one hopes. Therefore credible simulators must be validated which usually involves adjusting parameters to measured values.

Section 2.8 deals with the details of power measurement. The following section introduces two sensor network simulators which have support for power consumption estimation: PowerTOSSIM and Avrora. They may not be the best simulators but they are good examples for two different approaches.

### 2.7.1 Full System Simulation

Avrora (UCLA/Cornell project [134, 135]) is a scalable full system simulator for sensor networks (see Figure 2.12). It supports the Mica2 platform (see page 12) and can run applications based on TinyOS and SOS (see page 20). AEON (University of Tübingen) an extension keeps track of the power consumed by each sensor node. It can power down nodes if an energy threshold has been passed but lacks detailed battery models like [102, 110].

Avrora simulates the *full hardware* bit accurately and is therefore agnostic to the software it runs. Furthermore, the software does not need to be modified to work in simulation. For debugging this feature is invaluable and it also improves the accuracy of power estimations. Although Avrora simulates a full system it still maintains a high simulation speed which is important for design space exploration. For simple applications Avrora has been scaled up to 10.000 nodes on a big multi-processor machine [134].

Avrora achieves a high speed for several reasons: first, it detects sensor nodes which transitions into sleep mode and resumes their simulation only when an event is queued up for them. Second, it runs each node's simulation in parallel and third it optimizes their synchronization.

The assumption in Avrora is that synchronization between sensor nodes is only necessary because they communicate. Obviously, a sensor node that has run too fast into the future cannot sensibly handle radio packets it should have processed in its past.

The Avrora authors identified two challenges: the *send-receive* and *sample* challenge. The send-receive challenge states that a node should not run past the point in time where it could have received a radio packet. The sample challenge is related and states that the receiver signal strength indicator (RSSI) can only be determined when all transmissions in an interval are known.

Formally, the conditions can be expressed by:

$$wait(n,t_2),until(m,t_1) \mid t_2 > t_1$$

A node $n$ may not cross time $t_2$ until node $m$ has reached time $t_1$. For the send-receive challenge, the communication latency for one byte can be estimated as (Mica2 sensor node):

$$\begin{aligned} L &= \left\lceil \frac{\text{\#CPU cycles per second}}{\text{radio data rate in bytes per second}} \right\rceil \cdot 1\ [\text{byte}] \\ &= \frac{7372800\ [\text{CPU cycles/second}]}{2400\ \ [\text{bytes/second}]]} \cdot 1\ [\text{byte}] = 3072\ \ [\text{CPU cycles}] \end{aligned}$$

The sampling latency of the RSSI value has been given by the authors as (Mica2 sensor node):

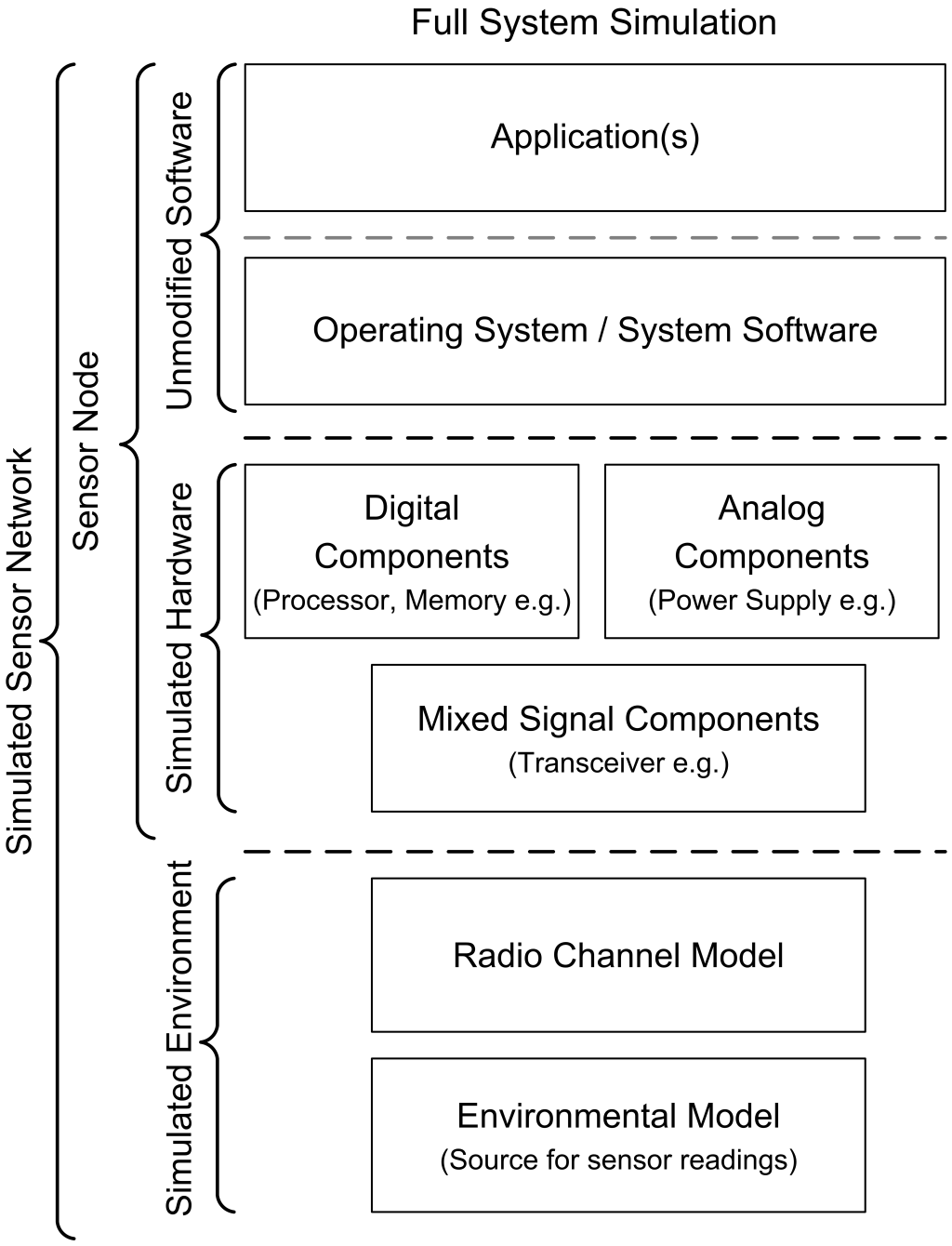



Figure 2.12: Avrora is a full system simulator. In a full system simulator the sensor node software runs unmodified on top of the simulated hardware. Digital components like the memory, processor and digital interface of mixed signal components (radio transceiver) are modeled cycle accurately. The analog components and environmental models may be modeled as a set of differential equations. For sensor readings or the radio channel it is common to sample representative statistical distributions. The radio model may be used to synchronize the nodes simulation time. For the Mica2 sensor nodes the communication delay has been calculated as 3072 cycles [134]. Often it is possible to progress each node's simulation independently for a simulation interval. On a (chip) multi-processor Avrora is able to exploit such parallelism to some extend.

$$S = 13 \text{ [ADC-cycles]} \cdot 64 \left[ \frac{\text{CPU cycles}}{\text{ADC-cycle}} \right] = 832 \text{ [CPU cycles]}$$

The following conditions must hold:

$$\textit{wait(m,t+}\mathbf{L}\textit{)until(n,t)} \mid m \neq n \text{ and } \textit{wait(n,t+}\mathbf{S}\textit{)until(m,t)} \mid m \neq n$$

Avrora synchronizes all sensor nodes every **L** cycles. Sensor nodes which want to sample the RSSI-value wait till all involved nodes have crossed the sampling point and then proceed to the synchronization point[30].

*This "trick" allows Avrora to run a node's simulation threads for* **L** *cycles instead of* **S** *cycles which reduces the synchronization overhead and thus improves performance.*

The synchronization mechanism would have to be changed if sensor nodes would influence the environment in such a way that other nodes would be affected.

Avrora's speed greatly depends on the amount of information which is collected during a simulation run.

AEON the power simulation extension is an example of a tool that hooks into Avrora. It calculates the consumed energy by tracking the number of cycles spent in every hardware component's state. The model is rather simple. The energy of a component $E_C$ is given by:

$$E_C = V_S \sum_{m=1}^{N} i_m \cdot c_m \cdot t_{cycle} \tag{2.8}$$

where $V_S$ stands for the constant supply voltage, $i_m$ stands for the constant current and $c_m$ stands for the number of cycles spent in state $m$ out of $N$. $t_{cycle}$ stands for the duration of a CPU cycle. The authors devised a set of microbenchmarks to measure the currents $i_m$ of each state. This is usually done by accessing hardware components in infinite loops and measuring the average current draw with an ampere meter.

Like in Quanto (starting on page 26) the assumption is that all power states are known as well as their *constant* current draw. Quanto will be challenged if the underlying hardware uses advanced low power techniques [7, 127, 82] which is common once the realm of tiny microcontrollers is left aside. Avrora can track power states in simulation but is likely to suffer from state explosion in complex sensor node system-on-chips (SoCs) which may involve multiple (heterogeneous) cores, caches and memories.

State explosion also complicates measuring the current draws: first because of the amount of states which must be measured and second because it may be difficult to generate all the required microbenchmarks. Varying current draw from analog components necessitates more advanced power models too.

[30] For the curious: the implementation can be found in the freely available source code "IntervalSynchronizer.java".

The Hyperion hardware platform which has been developed in this thesis uses real hardware (e.g. radio transceiver) for power measurements. The accompanying full-system simulator (Sensim) similar to Avrora provides power estimations based on recorded simulation traces. Unlike Avrora, Sensim computes the power consumption at simulation time. Therefore Sensim can run traces against differently parametrized hardware-, radio- and environmental models without simulation re-runs. A few utility software components of Avrora have been integrated in Sensim to shorten the development time.

### 2.7.2 Application Specific Simulation

TOSSIM from UC Berkeley is a simulator for TinyOS-applications only [88]. It is one of the most scalable sensor network simulator designs. TOSSIM is not a standalone simulator like Avrora (Section 2.7.1) but a compile-time target platform. Instead of generating code for the Mica2- or Telos-hardware platform, the code is compiled to the architecture of the developer's machine. This could be Linux on x86 for example. Obviously, a multi-gigahertz PC is capable of running TinyOS applications quite a bit faster than an 8 MHz microcontroller.

The TOSSIM platform offers the same set of hardware components to the applications as the real sensor node platforms do. However, the TOSSIM hardware components are not drivers for real hardware but simulation models which happen to provide the same software interface. The TinyOS component model which has been introduced on page 25 supports this approach well. Figure 2.13 shows a detailed break-down of a TinyOS application which targets a real sensor node platform and a simulation platform.

Special attention has been given to the components which model radio communication. For simple test purposes a fault free model is available. A more realistic model samples a Gaussian distribution which may be parametrized by distance. This model has been shown to be adequate for a string of sensor nodes placed on the ground.

The connectivity between the nodes can be specified by a directed graph. The edges going to and out of a (sensor) node are parametrized with probabilities for successful sending- and receiving.

The timing of the radio communication have been "accurately" modeled. The effects of missed start symbols or acknowledgments can be assessed for example.

BTW: An in-depth discussion of challenges in modeling radio communication can be found in section 2.6 on page 37.

The simulation speed of TOSSIM comes at a cost in accuracy. The way interrupts are modeled is different compared to the real platform. They do not preempt (long running) tasks. Furthermore, the execution of code is instantaneous in TOSSIM. Thus the execution time of a short interrupt and a long running task is the same in simulation. As previously explained, the device drivers of the target platform are substituted with simulation components. Thus their (subtle) code is not part of the simulation.

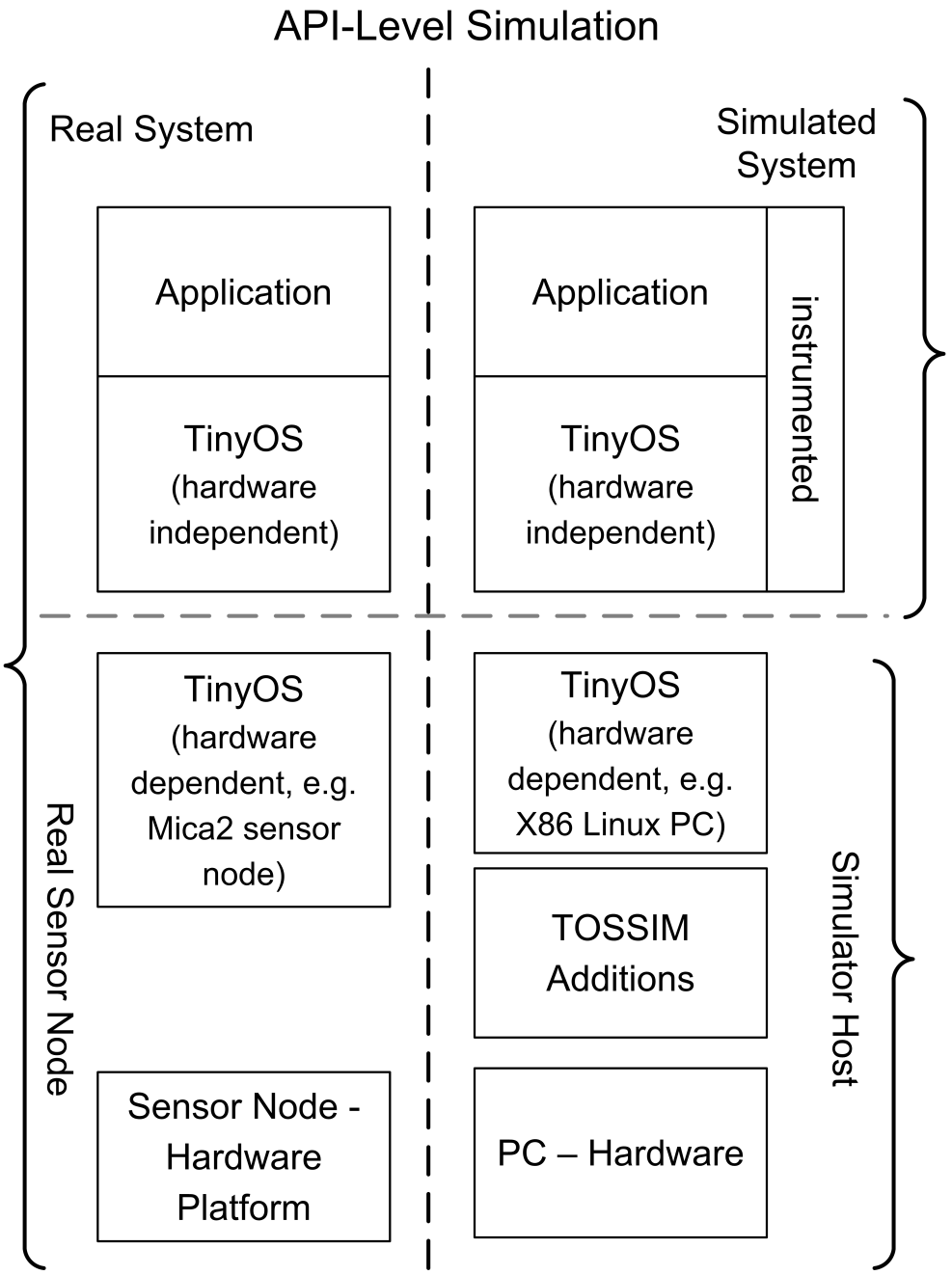



Figure 2.13: The figure shows the difference between a real TinyOS system and a simulated TinyOS system using TOSSIM. The real system consists of the application, the hardware independent and dependent TinyOS code which has been compiled for the target sensor node hardware platform. A TOSSIM simulated application is compiled to the simulation host hardware platform e.g. Linux on x86 and thus runs at full speed. The hardware dependent software parts - typically the device drivers - are substituted with high-level simulator components. Unlike in full system simulation the binary code differs between real- and simulated platform which may be a source of inaccuracies. For the simulation it is necessary that TOSSIM adds its discrete event simulation loop and reserves space for the state of each simulated sensor node. (Power)-TOSSIM an extension for TOSSIM instruments the application- and TinyOS-code in order to estimate the number of cycles in a simulation run.

The group around Matt Welsh at Harvard University tried to leverage the speed of TOSSIM and still obtain enough data for an accurate power estimation. The fruit of their work is called PowerTOSSIM [123]. PowerTOSSIM comes as an extension for TOSSIM.

PowerTOSSIM uses a power model comparable to Avrora/AEON (page 43) and Quanto (page 26). Therefore the challenges imposed by low power optimized hardware which have been discussed on page 45 apply too.

A short recapitulation of the power model used by PowerTOSSIM: The time spent in a certain power state is recorded and the consumed energy can be estimated by $E = V_S I \cdot T$ ($V_S$ supply voltage, $I$ current, $T$ time interval) - see also equation 2.8 on page 45). Like in AEON the constant current draw of each state is determined by running microbenchmarks on the real hardware.

*A PowerTOSSIM specific challenge is the absence of cycle accurate execution times for the target platform (e.g. Mica2).*

Since the CPU consumes quite a bit of power it cannot be left out of the power estimation. Matt and his students devised a slightly daring but interesting method to obtain CPU-cycle counts without compromising TOSSIM's high performance too much:

They instrument the application and the TinyOS code (see Figure 2.13) with basic block counters (basic block = sequence of machine instructions without jumps in between). *After* a simulation run the recorded counter values are back annotated to the C-source code and mapped to the basic blocks of the target's (e.g. Mica2) machine code. The number of cycles are the product of the basic block counter and the number of cycles needed to execute the basic block summed up over the entire application.

There is one problem though: *The program flow is not (time)-identical with the real platform because the code execution delays are not obeyed.* The mapping from and to basic blocks between different processor architectures over the source code is quite a challenge in itself and has still room for improvement as the authors concede in their paper.

A feature of the Mica2 processor is that the number of cycles needed by a basic block may be data dependent. Thus PowerTOSSIM cannot always determine how many cycles are really consumed. *Another limitation of the PowerTOSSIM methodology is the absence of platform specific code (TinyOS libraries and compiler additions) in the simulation:*

For example: low-level radio device driver code which heavily uses port and memory mapped I/O is not part of the simulation and may contribute significantly to power consumption depending on the application. I noticed that the compiler for the TMoteSky (a Telos clone) platform generates a purely software based multiplication subroutine which is not accounted for by PowerTOSSIM. Thus with *every* multiplication the cycle counter deviation becomes greater.

In the PowerTOSSIM paper the presented power measurements and estimations match quite well. The authors of the AEON paper - however - estimated the power consumption of the (active) processor in the application *CntToLedsAndRfm* and reported a **difference** of 5700% compared to PowerTOSSIM. The total deviation in power consumption is only 15% (the CPU

is idle 99.8% of the time). Conceptually, AEON should yield more accurate results and thus it is likely that PowerTOSSIM needs to be refined. Unfortunately, neither the authors of AEON nor PowerTOSSIM published a paper to shed light on the underlying problem.

Speed-wise PowerTOSSIM and Avrora/AEON should be roughly on par since TOSSIM alone (without the instrumentation and power-state logging) runs 50% faster [134]. Unlike PowerTOSSIM, Avrora/AEON can leverage (chip) multi-processors. Therefore the speed comparison depends on the size of the sensor network network (= number of nodes) and the number of processor cores available for simulation.

## 2.8 Assessment of Power Consumption by Measurement

Power measurements provide a *ground truth* and the obtained results are necessary to adjust simulation parameters for design space exploration. The scope and measurement methods vary with the requirements.

ICount from UC Berkeley / MIT exploits existing features in power supply hardware to measure average power consumption with little extra effort. Thus it is suited for online power profiling at deployment time.

On the opposite end of the scale are laboratory setups which can measure the power consumption of a (custom) microarchitecture within clock cycle accuracy. Such involved measurements are necessary to determine the power costs of microarchitectural transactions in non-trivial sensor node system-on-chips (SoCs). Cycle accurate power measurement will be introduced in the section after ICount.

### 2.8.1 Real-time Energy Profiling

The authors of [42] (UC Berkeley and MIT) proposed a very simple yet elegant method of power measurement:

The key idea is that a certain class of switching regulators[31] generate pulse signals which correspond to an almost fixed amount of energy over a wide range of current draws. Ideally, only a single wire needs to be introduced between a switching regulator and an external counter of the microcontroller (see Figure 2.14). The energy delivered within a cycle is approximately:

$$E = \frac{1}{2} L i^2 \text{ [J], where i=peak inductor current}$$

Because inductor inductance varies in production each board needs to be calibrated.

Figure 2.15 a) and b) show the relative error and resolution experienced with ICount.

[31] A boost, current-mode, pulse-frequency modulated switching regulator.

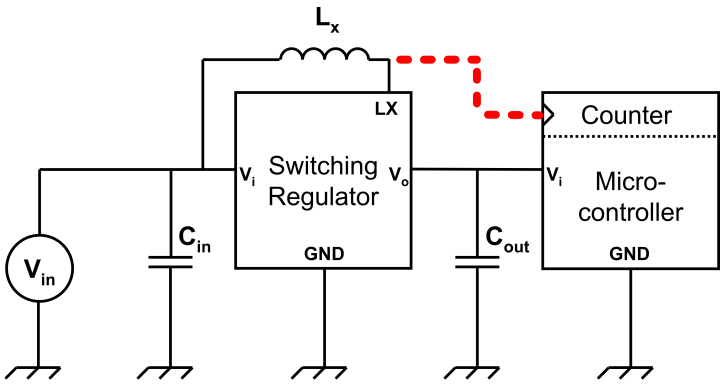



Figure 2.14: ICount taps a (pulse) signal from a pulse-frequency modulated switcher and feeds it into an external counter of the microcontroller. The counted pulses correspond to a fixed amount of energy. An increased power consumption causes more pulses to be generated within $\triangle t$.

The relative error stays with ±20% over five decades. The authors emphasize that for a typical energy draw exhibited by a basic sensor node the relative error is less than: ±15% ($5\mu A - 50mA$). The resolution (here: smallest measurable energy unit) has been determined for multiple (fixed) voltages using six different resistive loads.

$$
\begin{aligned}
P &= UI \overset{I=\frac{U}{R}}{=} \frac{U^2}{R} \\
P &= \frac{E}{t} \longrightarrow E = Pt \\
E &= \frac{U^2}{R} t, \text{ in the paper specficially: } E_{trial} = \frac{U_{out}^2}{R_{load}} T_{gate}
\end{aligned}
\tag{2.9}
$$

The authors estimated the energy $E_{trial}$ over time $T_{gate}$[32] using the equation derived on this page. The value of $E_{trial}$ has then been divided by the number of pulses within $T_{gate}$ which yields the resolution. The unit is $\frac{\mu J}{pulse}$.

In Figure b) it can be seen that resolution strongly depends on the voltage and increases sharply for high-load currents[33].

As mentioned before: ICount exploits the inner-workings of a pulse-frequency modulated switching regulator in order to track the power consumption of a sensor node with a *high time resolution*. Due to the low cost of this approach it is affordable to use this technique for real time power profiling of *deployed* sensor networks.

However, the high time resolution comes at a cost in measurement accuracy. If an accuracy better than ±15% is required then more elaborate and costly measurement circuits are necessary or

[32] The authors choose 1 second for $T_{gate}$ but did not give a justification.

[33] Prabal Dutta wrote - in his paper - that the resolution is reduced which is wrong. He told me that he intended "reduction" to refer to the plotted lines (Figure 9 in [42]).

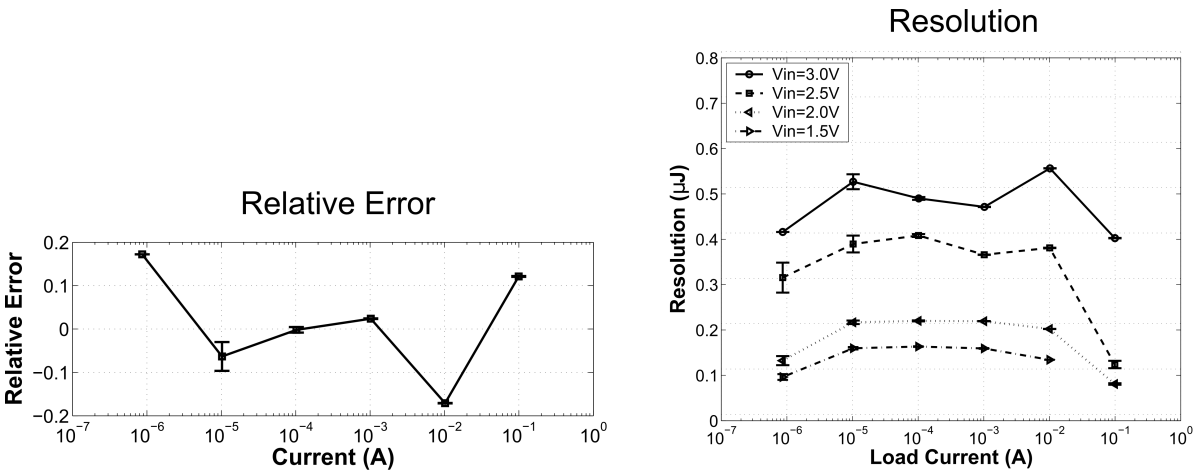



Figure 2.15: Figure a) shows that the relative error ($\frac{\text{absolute error}}{\text{ground truth}}$) experienced by ICount stays within ±20% over several orders of magnitude in current draw. Figure b) shows that the resolution varies non-linearly for different current draws and voltages. The current draw was adjusted by loading the regulator with different resistors: 33Ω, 330Ω, 3.3$k\Omega$, 33$k\Omega$, 330$k\Omega$ and 390$M\Omega$.

the time resolution needs to be reduced[34]. The error may even be higher if measurement artifacts are factored in: the counter - for example - introduces an additional frequency dependent power draw.

Generally, it remains to be seen if basic sensor nodes will benefit from voltage regulators at all. The Tmote sensor node which has been used by the authors usually runs directly from the batteries.

A voltage regulator's efficiency also varies a lot depending on the load. The regulator which has been used in the SunSPOT hardware platform (see page 35) - for example - has its highest efficiency close to the maximum power output and is *very* inefficient otherwise. The high frequency noise of the regulator must also be filtered out with decoupling capacitors which contribute to the power losses. To mitigate these high power costs the SunSPOT splits the system into "active" and "standby" components. The standby components run directly from the battery whereas the "active" components run off the regulator (see Section 2.3.2). Obviously, ICount does not integrate well with such an architecture.

Furthermore, ICount is not suitable for sensor nodes where a (sophisticated) power supply system interferes with the regulator in such a way that small current fluctuations are smoothed out. The authors concede that this may be the case for regulators with power factor correction and nodes with "big" (decoupling) capacitors. Latter may also prevent the application of (super)-capacitors in energy scavenging systems.

Despite ICounts shortcomings it must kept in mind that it only uses standard components. A

[34] E.g. averaging over a low-pass filtered voltage measured across a measurement resistor.

*customized* sensor node *power supply* should be able to operate efficiently in stand-by and active mode.

Currently, ICount enables power profiling for deployed sensor networks at a small cost but is not suitable for permanent use as is due to its high(er) power consumption in *standby*-mode.

*Prabal Dutta confirmed this in an email and we both agreed that sensor nodes would benefit from specialized regulators which currently seem to be commercially unavailable; maybe due to fundamental reasons.*

The hardware driven design space exploration platform *Hyperion* (part of this thesis) has been designed for highly accurate power measurements in a laboratory environment[35]. Like ICount it has support for power measurement with an extremely high time resolution but also offers low-pass filtered, average power measurements with a much higher accuracy. Hyperion - unlike ICount - uses *linear-voltage regulators* and has measurement circuit for every major hardware component. This enables an undistorted power break-down.

Linear regulators generate a smoother power signal than switching regulators but usually have a lower overall efficiency. Since Hyperion has been conceived as a design space exploration platform the focus was on measurement accuracy and not power supply efficiency.

### 2.8.2 Cycle Accurate Power Measurement

Naehyuck Chang from the Seoul National University has published several papers [26, 28, 27, 83, 122] about a *cycle-accurate capacitor switched power measurement* method and has demonstrated its applications to embedded systems. Cycle accurate power measurement enables the designer to investigate the power consumption on a microarchitectural level.

*Since sensor nodes must be extremely efficient in active- and especially in standby-mode it is crucial to assess even smallest amounts of power consumption within short time frames of activity.*

If the energy costs of microarchitectural transactions become measurable then the results can also be integrated into a power model within a cycle-accurate simulator. Since this thesis is about the conception of a design space exploration platform for sensor nodes it is paramount to show how such power model parameters can be acquired.

Even though it is possible to assess the power consumption with cycle accuracy it still remains a challenge to determine the precise energy cost of a microarchitectural transaction. Their power consumption can be state dependent, span multiple cycles and overlap with other transactions. This is especially true for (highly) pipelined architectures.

Naehyuck investigated the energy costs of the pipeline stages in an ARM7 processor [27, 28]. He discovered that address-, immediate-, register- and opcode-values have different shares in power consumption depending on the pipeline stage. Based on his findings he proposed a (simple) linear power model [27]. Unfortunately, he did not assess its accuracy.

[35] Hyperion has shielded plugs to interface with professional measurement equipment.

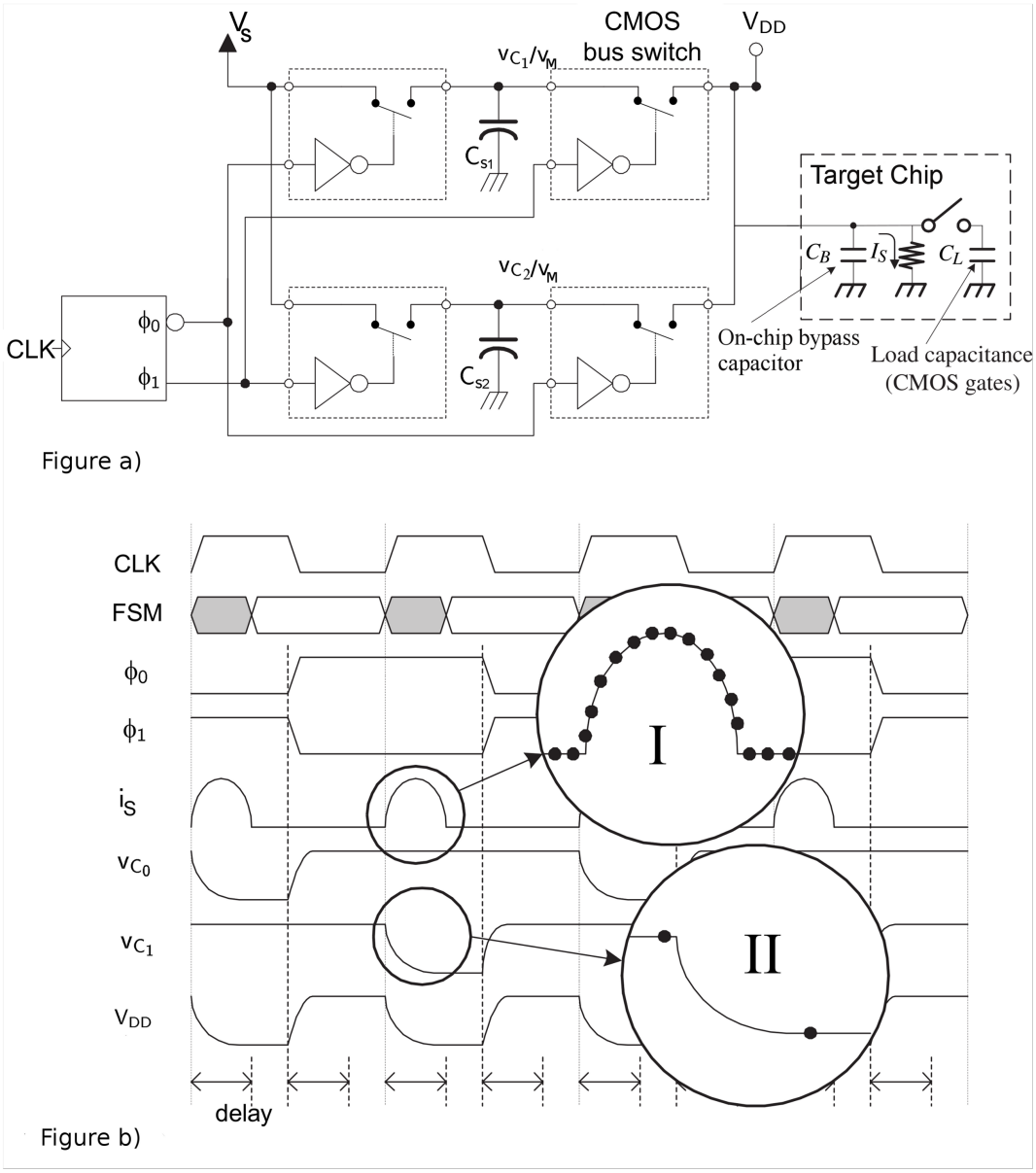



Figure 2.16: **Figure a)** shows the conceptual schematic of a switched capacitor measurement setup. Two capacitors $C_{S1}$ and $C_{S2}$ alternate between being charged by $V_S$ and supplying energy to the target chip ($V_{DD}$). The switches are shown with inverter gates because the authors implementation uses CMOS bus switch chips with active low enable lines. Conceptually, it would have been better to leave them out of the circuit diagram. The target *equivalent circuit* consists of an on-chip bypass capacitor $C_B$ and a load capacitance $C_L$. The leakage or static current is modeled as a current flow $I_S$ through a resistive load. **Figure b)** shows the timing diagram. The key idea is that the energy consumed within a cycle can be determined by measuring the voltage drop caused by the capacitor discharge. Only two sampling points (I) are needed. However, if a current measurement (II) is carried out then a high(er) sampling rate is necessary (Nyquist–Shannon sampling theorem). The figures were derived from [26] and [83] and have been extended and changed to be consistent with Figure 2.18 on page 59.

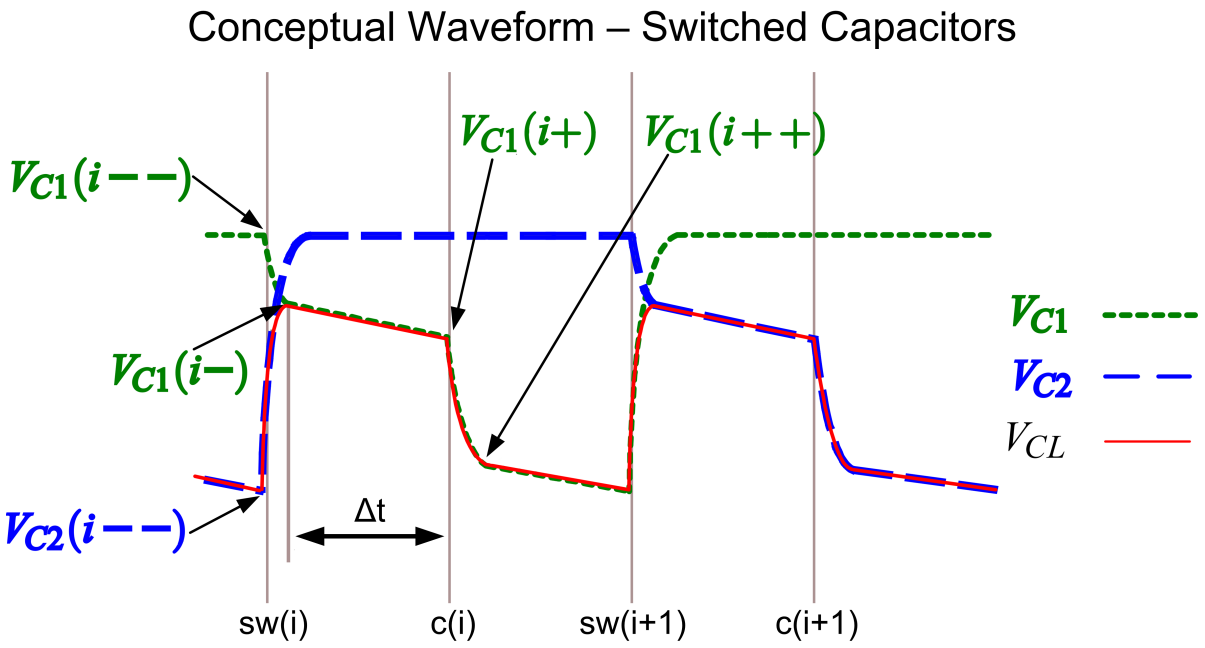



Figure 2.17: The figure shows the conceptual waveforms on which the switched capacitor model is based. The voltages across the two switching capacitors $C_{S1}$ and $C_{S2}$ and the load capacitance $C_L$ are shown. $\Delta t$ is the period for which the average static/ leakage power is calculated. sw(·) refers to the switch-events and c(·) refers to the clock-events. This enhanced figure is based on [83].

Figure 2.16 shows the conceptual circuit diagram of the switched capacitors power measurement methodology. The device under test (target chip in the diagram) is supplied with energy by two capacitors in an alternating sequence. While one capacitor is recharged from the main supply, the other discharges its energy into the target chip. It is also possible to use only one capacitor if the clock period is extended accordingly.

The voltage difference before and after the discharge corresponds to the amount of energy delivered:

$$\begin{aligned} E &= \frac{1}{2}CV^2 \\ \Delta E = E_2 - E_1 &= \frac{1}{2}CV_2^2 - \frac{1}{2}CV_1^2 \\ &= \frac{1}{2}C\left(V_2^2 - V_1^2\right) \end{aligned} \tag{2.10}$$

Equation 2.10 shows that the amount of energy depends on the capacitance and the voltage squared.

The conceptual waveforms (voltages across the switching capacitors $C_{S1}$, $C_{S2}$ and the load capacitor $C_L$) are modeled in Figure 2.17. The slightly confusing designations of the different charge states for $V_{C1}$ are: ++ (discharged), + (discharge through leakage / static current), – (connected to capacitor $C_B$) and – – (fully charged).

The first voltage drop from $V_{C1}(i--)$ to $V_{C1}(i-)$ is caused when the bypass capacitor $C_B$ is connected. The slight down slope from $V_{C1}(i-)$ to $V_{C1}(i+)$ along $\Delta t$ is due to leakage / static power. The leakage is modeled as a current $I_S$ through the resistor. The sharp drop from $V_{C1}(i+)$ to $V_{C1}(i++)$ comes with the rising clock edge when $C_L$ is connected. Please, bear in mind that the target chip is modeled as a simple equivalent circuit and $C_B$ and $C_L$ are just representatives for the total bypass- and load capacitances.

The size of the (on-chip) bypass capacitor can be approximated by applying the charge conversation principle[36]. With $Q = CV$ we can write:

$$\begin{aligned} C_{S1}V_{C1}(i--) + C_BV_{C2}(i--) &= C_{S1}V_{C1}(i-) + C_BV_{C1}(i-) \\ C_B &= C_{S1}\frac{V_{C1}(i-) - V_{C1}(i--)}{V_{C2}(i--) - V_{C1}(i-)} \end{aligned} \tag{2.11}$$

With

$$P = \frac{E}{t}$$

and Equation 2.10, we can describe the average static power consumption:

[36] Electric charge can neither be created nor destroyed. Thus the total charge after a state change remains the same.

$$P_S = \frac{1}{2}\left(C_{S1} + C_B\right)\left(V_{C1}(i-)^2 - V_{C1}(i+)^2\right) \cdot \frac{1}{\Delta t}$$

The transition from $V_{C1}(i+)$ to $V_{C1}(i++)$ is dominated by the dynamic power[37] and the equation for the energy - in the $i$-cycle - is:

$$\begin{aligned} E_D(i) =& \frac{1}{2}\left(C_{S1} + C_B\right)\left(V_{C1}(i+)^2 - V_{C1}(i++)^2\right) \\ \underset{\longrightarrow}{\text{generalized}} & \frac{1}{2}\left(C_{S1} + C_B\right)\left(V_M(i+)^2 - V_M(i++)^2\right) \end{aligned} \tag{2.12}$$

If dynamic power and static power are combined we may express the total energy over $n$-cycles as:

$$E_{TOTAL} = \sum_{i=0}^{n} E_D(i) + n\tau P_S$$

where the assumption is that the leakage power is constant and given by $\tau P_S$ (remember $E = Pt$). $\tau$ is the length of a clock period and $P_S$ the average leakage power.

To assess the intra-cycle power consumption Naehyuck's method only needs two sample two voltages points[38]. A current measurement - for comparison - requires that the full voltage envelope is sampled according to the Nyquist–Shannon sampling theorem. Hence, the measurement equipment must be able to acquire samples in the order of (multiple) GHz because logic gates switch at a constant speed in the nano second regime.

The presented measurement methodology can determine the intra-cycle power consumption. *However, it can not be applied if the target chip employs fine grained power management to minimize leakage losses e.g.* Fine grained power management causes the leakage power to *vary* which contradicts the model's assumption.

The non-ideal behavior of the involved parts influence the measurement accuracy. The (power) switches e.g. have a relatively high on-resistance of several ohms [26]. The switched capacitors $C_{S1}, C_{S2}$ are also imperfect and have resistive- and inductive losses. Unlike, resistors I am not aware of special measurement capacitors. Therefore they must be matched (if two are used) and calibrated (Josep Rius [113] about Naehyuck Chang's publication[27]).

Eunseok Song who has analyzed Chang's work and devised a more elaborate power model [125] points out that the bypass capacitance $C_B$ may *vary* across cycles and thus should not be assumed constant. Song proposes to calculate the energy transferred into $C_L$ instead of $C_B$. The

[37] Actually Naehyuck should have pointed out that he did not factor out leakage power. I assume this may cause distortions if the amount of leakage becomes significant e.g. for chips with feature sizes below $65nm$.

[38] In [125] Eunseok Song suggests three measurements per measurement interval since $C_B$ is not necessarily constant across clock cycles.

load capacitance $C_L$ is less susceptible to changes in the supply voltage experienced in the measurement setup.

The catch is that the capacitor method is based on measuring voltage changes before and after a clock cycle. Ideally a real board has decoupling capacitors around the chip which keep the supply voltage constant at all time.

The (dynamic) energy in Song's method can be calculated with the following equation:

$$E_D = \frac{1}{2} C_{L(\text{Switching})} V_{DD}^2 \overset{\cdot 2 \text{ due to CMOS}}{\longrightarrow} C_{L(\text{Switching})} V_{DD}^2 \tag{2.13}$$

$V_{DD}$ is the voltage applied to the target chip on a board without the measurement circuitry. *Chang's dynamic energy calculation - in comparison - yields the energy consumed under measurement conditions.* ($V_M$ in Equation 2.12 on page 56).

The equation 2.13 must be multiplied by 2 because in VLSI-CMOS half of the charge-up energy is dissipated as heat [75]. The other half is dissipated when charged-down. This - however - does cause a current draw from the supplies.

In CMOS the load capacitors can be in four different states: high-, low- and transitioning from high to low and vice versa (see Figure 2.18). Song's model estimates the energy which goes into the target chip (based on Equation 2.13):

$$E_D = C_{L(\text{Switching})} V_{DD}^2 = \left(C_6(n) + C_7(n)\right) V_{DD}^2 = C_{67}(n) V_{DD}^2$$

$C_7(n)$ and $C_7(n)$ (see Figure 2.18) stand for the capacitors which are charged in the $n$-th clock cycle. By applying the charge conversation principle it is possible to calculate $C_{67}$ (see [125] for a detailed derivation):

$$\begin{aligned} C_{67} =& C_{H67}(n) - C_H(n) \\ =& \left(1 - \frac{V_3(n)}{V_2(n)}\right)\left(\frac{V_1(n+1) - V_3(n)}{V_2(n+1) - V_3(n)}\right) C_M \\ & - \frac{V_3(n)\left[V_2(n+1) - V_3(n) - 3\right] + 3V_2(n)}{V_2(n)\left[V_2(n+1) - V_3(n)\right]} \cdot \frac{I_{Leak} T_M}{8} \end{aligned}$$

where $C_H$ is the equivalent static capacitance of the sum of $C_B$, $C_1$, $C_2$, ...,$C_4$ and $C_{H67} = C_H(n) + C_6(n) + C_7(n)$. $\frac{I_{Leak} T_M}{8}$ is the leakage charge experienced in the time interval for which the charge conversation principle had been applied in the derivation. $T_M$ is the duration of the clock cycle and $I_{Leak}$ is the *constant* leakage current.

Song has compared his method against Chang's by measurement: for different values of $C_M$ ($5.3 - 22nF$) his method has an overall (constant) deviation of less than−1%. Chang's method has an average deviation of −2.6% ($22nF$) which grows up to −6% ($5.3nF$) as the capacitor size is decreased. From these results it is obvious that Chang's model depends on the size of $C_M$.

Smaller values of $C_M$ cause greater supply-voltage fluctuations ($V_M$) which improves the signal-to-noise (SNR) ratio of the measurements but also influences the circuit behavior. As mentioned before, Song's method can elude this catch-22 situation: his power estimation model is much less susceptible to changes of $V_M$ (power supply voltage seen by the target chip on the measurement board). He determines the dynamic capacitance of $C_L$ (remember: less susceptible to variations in $V_M$) and calculates the energy for a hypothetical and constant power supply voltage $V_{DD}$ found in a non-measurement board.

*Both methods assume constant leakage currents which may not be true in the case of fine grained power management.* A further limitation is that the clock speed must be slowed down so that the supplying capacitors can recharged for the next cycle. Song - for example - has reduced the clock speed to $625kHz$ in his experimental setup. Thus the dynamic switching activity is stretched over time whereas other events may not (e.g. power gating causes leakage to be reduced immediately).

In this thesis a *resistor based* intra-cycle power measurement method has been devised and implemented in the hardware driven design space exploration platform *Hyperion*.

The resistor method seems easier to implement (which is not necessarily true as we will see) and puts high demands on the measurement equipment.

Song's and Chang's methods - for comparison - strive to reduce the number of captured samples to simplify the measurement process. They only need to measure a *few* voltage changes between clock cycles. In Hyperion the voltage drop across the measurement resistor is sampled by a scope with 4 GS/s. Repeated measurements have shown a measurement variability of $\pm 1.43mW$. The measurement setup is explained in-depth on page 98.

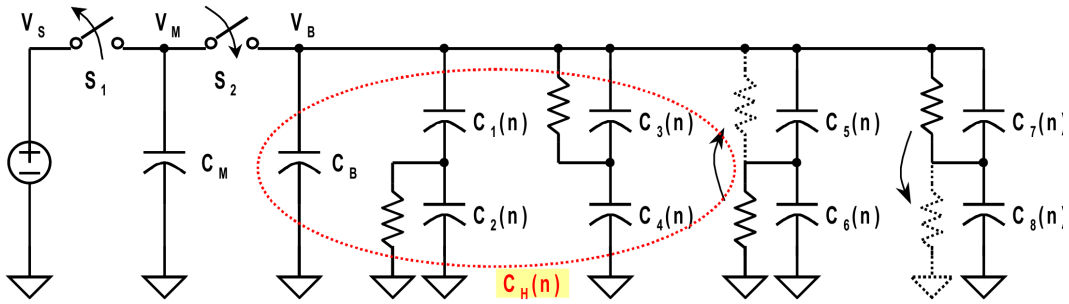
The energy can be calculated by integrating over the captured wave form:

$$E(t) = \int \frac{u(t)^2}{R_{Shunt}} dt$$

*Hyperion can measure chip power consumption even if fine-grained power-management is employed.* Since it does not need nor try to disambiguate leakage- from dynamic power which would require information about the circuit's internal state.

Another difference is that Hyperion does not restrict the clock frequency of the target logic since it does not need to charge measurement capacitors. *The flip-side - however - is that Hyperion must always sample at a high frequency to capture the waveform across the measurement resistor.* The high sampling frequency is necessary because logic-gates switch at a constant speed independently of the clock and the Nyquist–Shannon sampling theorem must be fulfilled.

Thus for Hyperion the length of a measurement period depends on the amount of memory inside the scope whereas the capacitor methods may run virtually indefinitely. Firstly, because they need less sampling points (2-3 per measurement interval). Secondly, because the target chip runs at a reduced clock speed.




| Group | Symbol | Description |
|---|---|---|
| 1 | $C_1(n)$ / $C_2(n)$ | Stay in the low state |
| 2 | $C_3(n)$ / $C_4(n)$ | Stay in the high state |
| 3 | $C_5(n)$ / $C_6(n)$ | Switch from low to high |
| 4 | $C_7(n)$ / $C_8(n)$ | Switch from high to low |

Figure a)

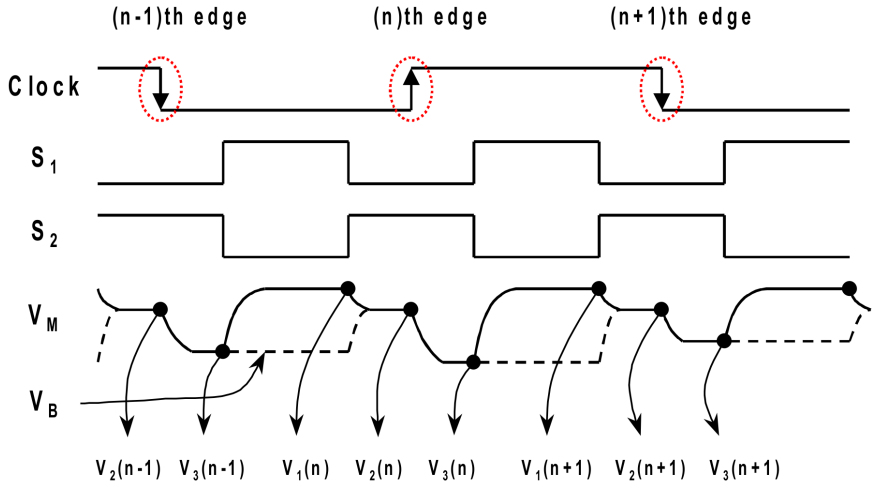



Figure b)

Figure 2.18: **Figure a)** shows Eunseok Song's [125] improved model of the target chip. The VLSI-CMOS capacitors can be in four different states: low, high, transitioning from low to high and vice versa. The capacitors $C_B$, $C_1$, $C_2$, ...,$C_4$ are static and are therefore combined into one equivalent capacitance $C_H$. Compared to Figure 2.16 ($C_{S1/S2}$) only one capacitor $C_M$ is used which simplifies the setup and may help to improve accuracy (no capacitor matching necessary e.g.). The drawback is that the clock speed must be reduced to account for the recharge time of $C_M$. **Figure b)** shows the timing diagram [125]. Song's model requires three measurement points to estimate the dynamic power.

## 2.9 Summary and Conclusions

A selection of influential hard- and software platforms have been introduced in this chapter. Interestingly, only a few hardware platforms (e.g. Mica / Telos from UC Berkeley and the SunSPOT) enjoy a widespread use and thus many (comparable) publications.

A number of *specialized* sensor node system-on-chip (SoC) designs have been published as well: WiseNET [43, 44], PicoRadio [108, 107] and Spec [62]. Their designs all had a very strong focus on the *radio* transceiver: the common perception among the creators must have been that a low-power, low data-rate transceiver is the ultimate goal. None of the designs has been subject to an rigorous *application driven design process* nor to an (automated) design-space exploration. WiseNET, PicoRadio and Spec seemed to have never made it out of the laboratory into experimental deployments. Thus their efficiency for a given application has never been determined. PicoRadio may have been the most flexible platform due to its built-in (small) FPGA fabric.

Hogthrob was conceived as a FPGA based sensor node for *design-space exploration* but the project had never made use of it[39]. Despite being an exploration platform Hogthrob did not have built-in *power measurement* which is crucial to assess the performance of a sensor node.

A vector based power estimation methodology has been conceived as part of the Hogthrob project. The key idea had been to specify a system by means of an application- and system characterization vector. By applying varied system vectors it is possible to predict an application's performance on an unknown hardware platform. This methodology while very interesting still lacks accurate models for system- and application vectors.

The Quanto methodology uses linear regression to estimate per component power consumption by tracking power states and total power consumption. Additionally, it helps the designer to understand where power has been consumed on a higher level: based on source code *extensions* and *annotations* the hardware component power consumption is projected onto network- and node-scale *(high-level) activities*. Unfortunately, Quanto does not track activities hierarchically to save system resources which limits its tracking capabilities (it runs on a sensor node after all) .

Quanto has been used together with the ICount power measurement method. In ICount a specific kind of switching regulator is wired to a hardware counter to accurately estimate the total power consumption by counting pulses. This power measurement method has a high resolution and is quite inexpensive. It is suitable for profiling selected nodes in a deployment for limited periods of time. In a real deployment ICount would suffer from regulator inefficiencies in standby mode. Under laboratory conditions it is not necessary to use (such) a regulator since professional measurement equipment is accessible.

The capacitor switched power measurement methods allow deep insights into microarchitectural power consumption which is a good basis for accurate (power) simulation models. Unlike resistor based measurements only three voltages must be sampled within a clock cycle. A clock cycle may and must be stretched to some extend to allow the capacitors to recharge. Due to the

[39] LEAP [41, 95] was another exploration platform (not FPGA based) which seems to have been abandoned too.

slower running clock it is possible to record cycle accurate measurements almost indefinitely. A drawback of the capacitor based cycle accurate measurement is that leakage currents must not change. This may cause problems with hardware designs which employ fine grained (leakage) power management.

Besides real power measurements two simulators with power estimation capabilities were introduced: Avrora/AEON and PowerTOSSIM. Both simulators target existing sensor node hardware platforms and are *not* prepared for design space exploration. Their simple power models track hardware state changes and utilization time in order to sum up the consumed energy. The assumption is that all states are observable and have a constant power consumption. Hardware devices beyond tiny microcontrollers - however - have many hidden states.

Avrora/AEON is a full system simulator. It simulates and tracks the state of all hardware components accurately. Thus it is operating system software agnostic. PowerTOSSIM on the other hand generates an application specific simulator. TOSSIM substitutes hardware components with PC-simulation components to maximize the simulation speed. Comparisons done by the AEON authors reveal major differences in the power estimations between both simulators. My personal analyses reinforce the assumption that PowerTOSSIM is flawed.

Both simulators model radio communication but the results should not be relied on. The challenges of modeling radio communication in large scale, low-power sensor networks have yet to be improved. To familiarize the reader with the challenges encountered in radio modeling a brief excursion into the subject was given. The awareness that proper wireless communication models are a necessity has risen within the wireless sensor network community. Until trustworthy models are accepted experiments are required to back new concepts.

On the software platform side this chapter has presented the novel TinyOS operating system, the generic Contiki, the software reconfigurable SOS and the embedded Java virtual machine Squawk. At the time of writing SOS had been discontinued, whereas Contiki and TinyOS enjoyed a quite active community. Surprisingly, none of the operating systems has extensive system support for power management. Contrary, some of the (simple) mechanisms are likely to interfere with the application requirements and must therefore be deactivated or changed before a real deployment. Contiki avoids such hassles by not offering any built-in power management strategies at all. Since power management is very application specific the integration between system policies and application requirements seems very delicate.

The related work has now been introduced, summarized and a connection has been made to this thesis. The discussion of related literature does not end though. It extends over the following chapters and cross-references to the related work are made. Before announcing the following chapter I would like to thank the authors of the related work who have answered my questions by personal correspondence. They have motivated me and helped to enhance the quality of this chapter.

### 2.9.1 Goals of this Thesis

Current sensor node development platforms emphasize the exploration of software for wireless sensor networks and neglect the hardware side. Like many other embedded systems wireless sensor networks would benefit enormously from custom hardware accelerators - be it on the computation- or communication-side. Specialized sensor network system-on-chip (SoC) designs like the aforementioned PicoRadio [108, 107] and Spec [62] are well-known SoC-designs within the community. However, these are custom chip designs which had been very time consuming to develop and costly to build. Except for the initial publications on the first silicon nothing more has been published. It is likely that these projects had run out of time or failed their objectives - or a combination of both.

Therefore it is necessary to establish a *flexible design space exploration platform* which does not require a custom silicon chip and board re-spin every time the designer tries out new ideas or simply tweaks a few (design time) parameters.

Since power consumption is the crucial metric in wireless sensor networks it is a number one priority to have *high fidelity* power measurement support designed into the exploration platform. The power consumption feedback is a necessity in order to guide the exploration process. The results of the power assessment must be tractable so that the designer has a chance at spotting worthy optimization targets. Therefore, a design space exploration platform must be able to break-down the power consumption into *comprehensible categories* of functionality on a task-level.

Quanto has been a step into this direction but falls short in various aspects[40]. Besides being intrusive on the source code level, Quanto does *not* support hierarchical power tracking as mentioned before. Due to the inherent hierarchical composition of software (functions) Quanto looses a lot of expressiveness and flexibility by introducing this mismatch. Therefore a design space exploration platform must track *categorized* high-level tasks *hierarchically* and should also separate the *category specification* from the application- and system source code. This allows the designer to quickly switch between different power break-down perspectives without re-building the entire system.

Besides measuring the average power consumption of all involved hardware components it is desirable to assess the power consumption on a micro-architectural level since sensor nodes must be extremely power efficient in active- and especially in standby-mode. A *cycle accurate power measurement* setup allows even smallest amounts of energy to be measured within short time frames. The presented *capacitor based* measuring techniques reduce the measuring overhead but bring with them a number of constrains regarding the circuit behavior. If high-end laboratory equipment is integrated with a carefully designed custom measuring board then a less constrained *resistor based* measuring approach may be used to capture the full wave form of a single clock cycle[41]. This method reveals many more details like intra-cycle peak power

[40] Quanto requires - for example - that all hardware states are observable and have a constant power draw and that no hardware components switch simultaneously within the monitored time frame.

[41] "*Hyperion: A sensor node test bed for (high-speed) power measurements*" - Dominic Hillenbrand, Michael Mende, Timothy Armstrong, Jörg Henkel,[38],

consumption for example.

The discussion of challenges experienced in modeling radio communication has revealed that current radio- and channel models are limited and may deviate significantly from the ground truth. Hence, a design space exploration platform must have additional methods of aiding the designer in communication design such as interfacing to a real one. Since the design space of radio transceivers and antennas is vast a design space exploration platform must either provide a software defined radio or be extensible so that multiple (configurable) transceivers can be plugged into it to span the entire desired design space.

The current sensor network simulators model only existing sensor nodes. For a design space exploration platform it is necessary to derive the system from an *architecture specification*. This allows the designer to compose different versions of a baseline architecture and compare their performance within an acceptable time frame. An automatic system construction and code generation are means to cut down the time needed for each design iteration.

To further cut down the design iteration time it is necessary to *separate* the simulation and power assessment. Avrora/AEON combines simulation and power calculations into a single run. A separation would - however - allow the power consumption to be calculated repeatedly for varied power models without actually re-running the simulation.

A separation of simulation and performance assessment also allows the design flow to be extended with custom analyses without affecting the existing environment. Custom analyses are crucial if the sub-design space of individual hardware components must be explored. They help to identify optimization potentials which are best found by computing component specific metrics.

In this thesis a flexible design space exploration platform for advanced sensor nodes shall be presented. It combines software- and real time hardware simulation. The platform overcomes many of the limitations of prior isolated approaches and combines them into a comprehensive platform. To the best of my knowledge no comparable design space exploration platform for sensor networks has been conceived before.

The following chapter introduces the central application theme, the envisioned system, distinguishes between basic and advanced sensor nodes and presents the methodology followed in this thesis.

# 3 The Envisioned System and Methodology

This chapter describes the envisioned system (also referred to as *system vision* in this thesis) and the central application theme. The system vision introduces a flexible design space exploration platform for wireless sensor networks where parts of the simulation model can be shifted to real hardware. Models that interface with real hardware components enable accurate power- and timing-measurements of computational-, radio communication- and sensor-components.

Power consumption is one of the *crucial metrics* in the design of a wireless sensor network. The key components must be assessed during the design space exploration. It is also important that the power models are validated and parametrized on the basis of real measurements.

The first section introduces the *application vision*: a camera tracking sensor network. This central application theme extends into the following chapters. Camera tracking offers a wide network- and node-level design space and grounds the introduction of *advanced sensor nodes* in this chapter. A clear distinction is made between *basic* and *advanced* sensor nodes.

The chapter closes with an explanation of the methodology followed in this thesis.

## 3.1 Central Application Theme: ''A Camera Tracking Sensor Network''

Figure 3.1 visualizes the central application theme: a camera tracking sensor network.

A camera tracking network suggests a heterogeneous network architecture. Since a camera sensor node is relatively expensive it would be wasteful to have only one kind of sensor node in the network. Therefore it is reasonable to design at least two different types of sensor nodes. For example: a camera- and a radio-infrastructure node. This allows each sensor node to be matched to a task.

A camera sensor node is a good example for an *advanced* sensor node. It requires sufficient resources to handle a high bandwidth 2D-image signal and enough energy to power the valuable sensor nodes as long as the requirements need them to last.

Radio infrastructure nodes relieve the camera nodes from routine tasks such as time synchronization. They also coordinate the wake-up calls to sensor nodes which are close to the tracked target's future position.

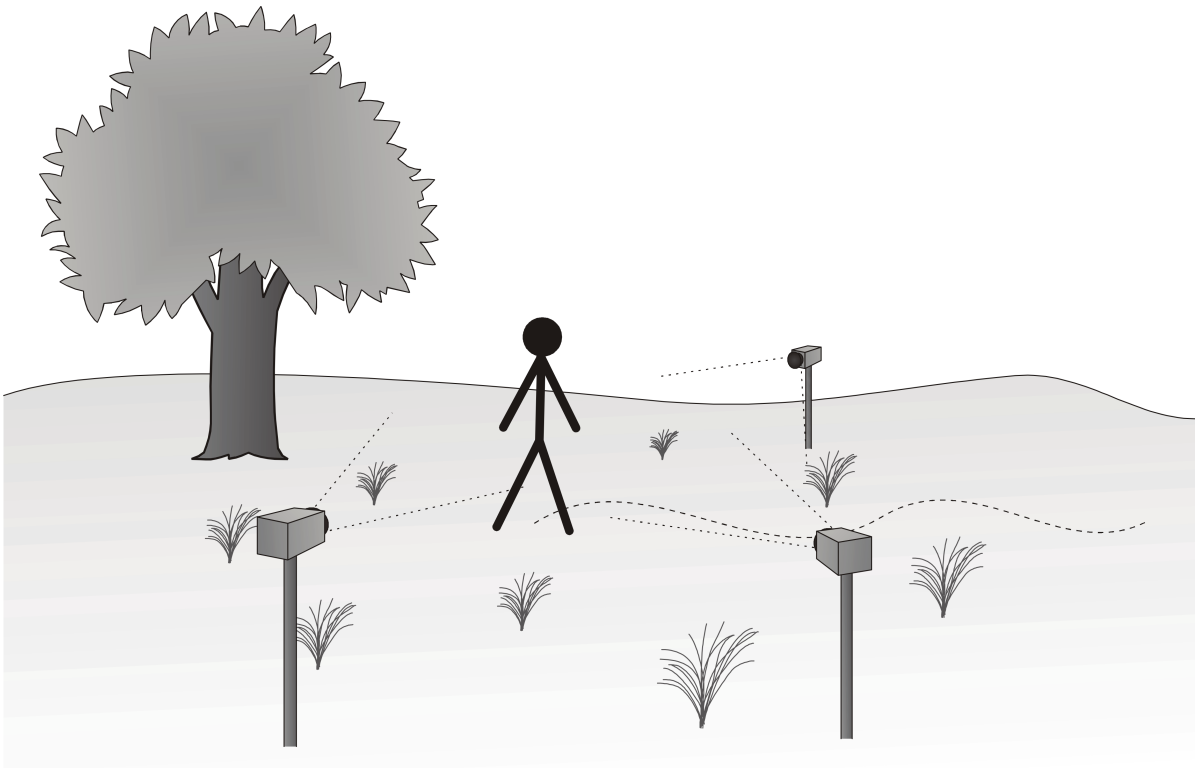


Figure 3.1: The central application theme in this thesis is a camera tracking sensor network. A person which enters the network's perimeter is sensed and tracked by one or multiple sensor nodes. The persons position is propagated through the network and may be queried.

A radio infrastructure node is a *basic* sensor node. It processes and generates signals and helps to maintain the network's connectivity. For this task a low data rate is sufficient in communication and computation. A sensor node like the Telos on page 14 is a good example. It combines a low power radio transceiver with an ultra low power microcontroller.

For position estimation of the target it is necessary that the camera nodes are aware of their location. This may be achieved by pre-programming the location into the camera nodes at deployment or by using built-in GPS receivers. GPS receivers - of course - provide more flexibility in exchange for an increased hardware overhead.

For a big number of radio infrastructure nodes both localization approaches are undesirable. For them it is sufficient to estimate their relative position in relation to the camera nodes. The radio infrastructure nodes can estimate their location based on the RSSI ( = Received Signal Strength Indicator ) value which is provided by the radio transceiver. The position estimates ought to be sufficient to realize a multi-hop network protocol with geographical routing.

The camera tracking application which has just been outlined is an *example* of a non-trivial and computational intensive wireless sensor network application. It may not be the best example but it is certainly sensible to stick to a sample application in a thesis and pick it up whenever appropriate - for example in chapter 6 where the case studies are presented.

Camera tracking sensor networks have spawned quite some interest in the sensor network community. Therefore I would like briefly reference related work for the sake of the interested readers.

Obviously, the most common use case is surveillance [56, 37]. Some authors focus on tracking humans [15, 33] and go as far as recognizing gestures [33]. Power consumption is especially important in camera tracking due to the high data rate of image sensors. This publication [152] looks intro distributed coordinated power management for example. A small microcontroller is quickly overwhelmed by the amount of acquired image data. This insight motivated the authors of this publication [36] to investigate FPGA based implementations. Accuracy in camera tracking requires calibration which has been looked into, in this publication[12]. Another interesting publication [12] takes a high level specification and derives the camera specification and parameters from it.

In this thesis the design space exploration is focused on the sensor node hard- and software and the assessment of their power consumption.

## 3.2 Distinction: Basic and Advanced Sensor Nodes

In this thesis a distinction is made between *basic* and *advanced* sensor nodes.

Basic sensor nodes only have a bare minimum of functionality whereas advanced sensor nodes have highly complex and specialized architectures. Figure 3.2 shows a scale with representative sensor nodes arranged by their capabilities:

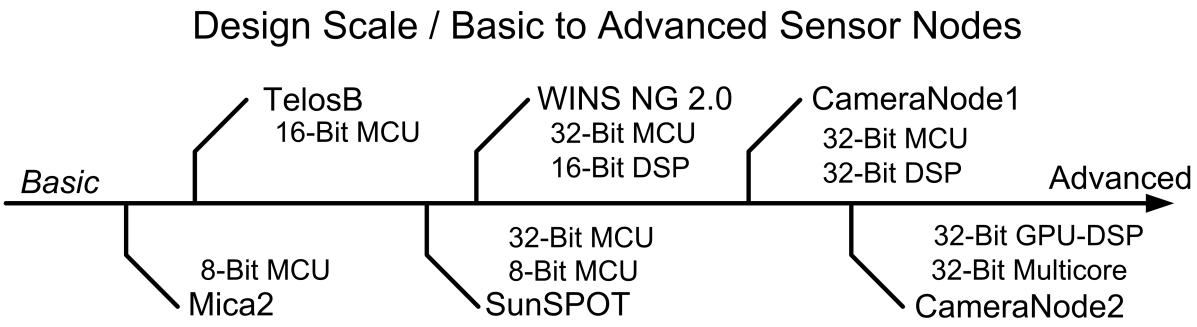



Figure 3.2: Basic sensor nodes have little computational power. Advanced sensor nodes may have multiple dedicated processing elements assigned to a specific tasks. Along with the computational power comes more memory, more complex operating systems, additional high data rate radios and elaborate power management.

The Mica2 and Telos-B (see pages 12 and 14 ff) are good examples for basic sensor nodes. They have little memory and computational resources. Subsequently, their system software has a strong focus on minimal code size and a small (run time) memory footprint (see page 20 ff).

The SunSPOT (see pages 15 ff), the WINs series and a hypothetical CameraNode1 sensor node are examples of today's advanced sensor nodes. Their hardware is capable enough to run embedded versions of the Java virtual machine or Linux. As a side effect large amounts of existing 32-bit software components are accessible.

To preserve power efficiency advanced sensor nodes may use:

1) Computational accelerators - Digital signal processors (DSP), application specific processors (ASIP) or field programmable gate arrays (FPGAs) for example. These accelerators can carry out signal processing tasks much more efficient than a microcontroller. Microcontrollers often lack elementary functional units like an integer multiplier.

Advanced sensor nodes may still use microcontrollers for simple tasks such as power management. Computational intensive tasks such as audio- and image-processing - for example - are best mapped to specialized digital signal processors, application specific processors or even custom hardware (as mentioned before).

2) The multitude of power modes in advanced sensor nodes allows the system software to incrementally power down memories and processors. Advanced sensor nodes can even attain power consumption levels comparable to basic sensor nodes when in (deep) sleep mode. The SunSPOT sensor node is a good example - see page 16 et seq.

The CameraNode2 is a place holder for future sensor nodes. It is not unrealistic to predict multi-core and GPU-powered sensor nodes. Upcoming consumer mobile phones will have these accelerators soon and thus lower the costs and propel architectural design advancements like die-stacking for example.

This thesis is concerned with the design space exploration of basic and advanced sensor nodes with a focus on latter. The modeling of future sensor nodes like the CameraNode2 would go

beyond the scope of this thesis and would have impeded a realization of the design space exploration platform. The system vision of the design space exploration platform is introduced in the following section.

## 3.3 The Envisioned System: A Flexible Design-Space Exploration Platform for WSNs

The envisioned system (also referred to as system vision in this thesis) is a flexible design space exploration platform for wireless sensor networks which supports the designer by assessing the performance and power consumption.

Figure 3.3 shows the envisioned system or system vision. The software components are shown in the middle. They encompass the application- and system-software components.

The system software consists of libraries, hardware device drivers and some kind of operating system. The hardware components are shown on the left side. They *interface* with the software components in the middle. A certain hardware component may have multiple types. There may be different types of radio transceivers, sensors and power sources for example. More important is the fact that *hardware* components may be simulated or real. Thus the designer has the freedom to choose the type and manifestion of each hardware component. The choice may depend on the planned evaluation or the stage the design has reached.

The right side of the figure shows the environment. Again some components are real and some are simulated depending on the simulation configuration.

At the bottom center of the figure is the evaluation domain. Depending on the configuration of the system the power consumption of a component is measurable or must be estimated.

The hardware- and environmental-components vary in accuracy and simulation speed. As indicate before, the designer has to decide which components are accurate enough for the planned evaluation. The data that is gathered from the real components can be considered as a *ground truth.* Based on this data simulation models must be parametrized and refined. This process is called simulation validation.

Real sensor readings may be saved and injected into non-real simulation components instead of synthetic data. Synthetic data may be necessary to model dangerous or rare phenomena - for example a fire or an explosion.

Some aspects of the environment are hard to model accurately in simulation or take a long time to compute. This is especially true for radio communication. The challenges of modeling radio communication have been discussed on page 37 et seq.

In the envisioned platform an additional method is offered. The system may interface with real radio transceivers. For some assessments this may be the preferred method to acquire credible results on wireless communication experiments.

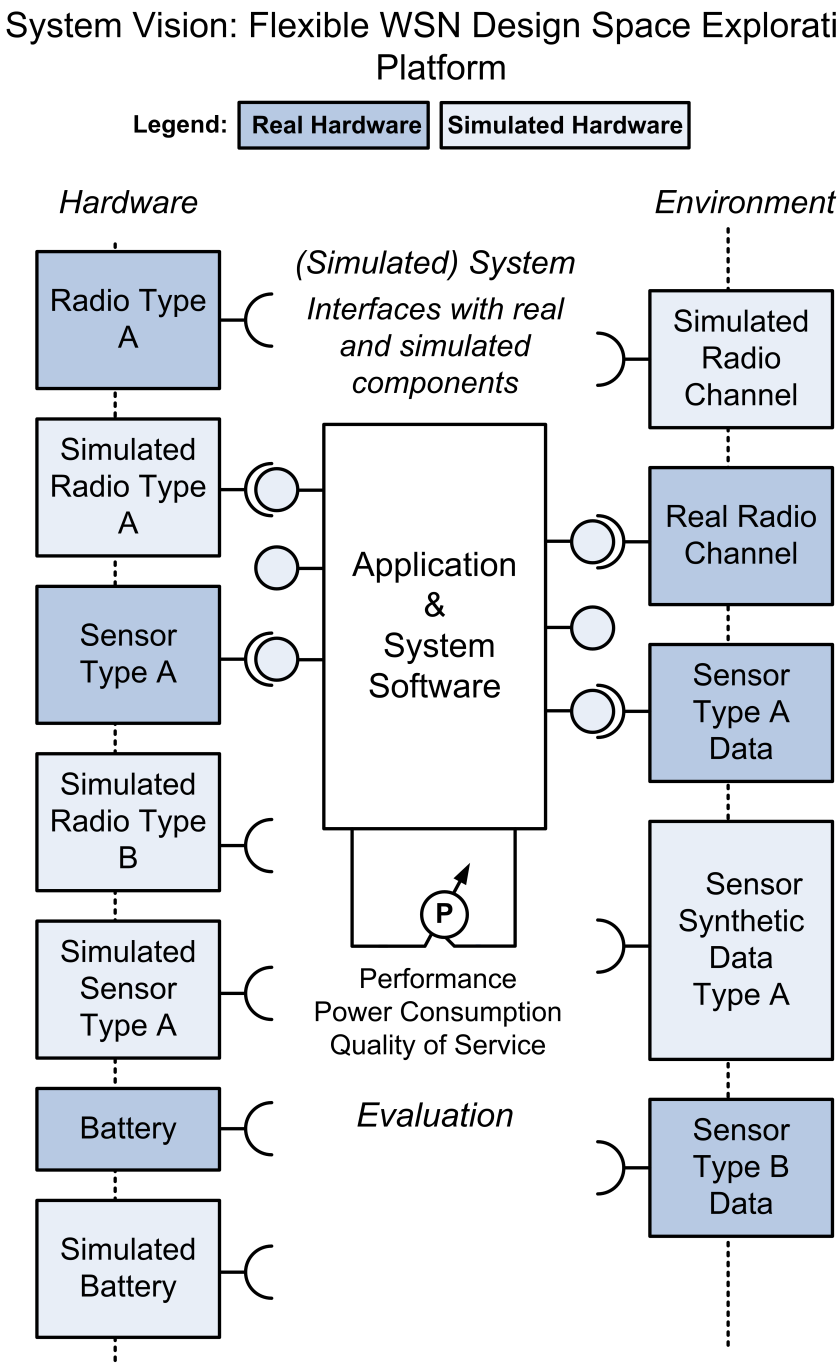



Figure 3.3: The figure shows the envisioned design space exploration platform for wireless sensor networks. Application- and system-software components may interface with real- and simulated hardware components. The environment where sensor data is acquired and radio waves propagate may be real or simulated. A sensor node prototype composed of real- and simulated components shall be assessable in its performance and power consumption. For real components by measurement and in simulation by estimation.

Virtual hardware components on the other hand are easier to create or get hold off. Unlike their real counterparts their design time parameters can be *sweeped* in simulation. Thus in the early stages of the design space exploration it is advisable to start off with non-real or virtual hardware components. In the later stages of the design when timing - for example - becomes important then it is desirable to interface with real hardware components. In a camera tracking sensor network this may be the case if time synchronization over the radio link must be evaluated. Time synchronization is especially important in a camera tracking sensor network because the acquired images must be time tagged before propagation.

Since sensor nodes often rely on limited power sources none of the sub-systems may introduce inefficiencies. *Hence a holistic design is required.* The design space exploration platform allows the designer to build up a system from simulated components which can be gradually refined with real components to increase the accuracy of the assessments.

The data supplied by the design space exploration platform must provide comprehensible feedback for the designer - especially for the pivotal metric, the power consumption.

## 3.4 Methodology

The system vision has been laid out in the previous section. This section outlines the methodology followed in this thesis.

Figure 3.4 visualizes the methodology. First the design space of wireless sensor networks is investigated. The investigation is structured into the following design domains: computation, communication, sensing and power supply. To limit the scope of the thesis and increase the usefulness of the planned design space exploration platform it has been decided to focus on two domains: *computation* and *communication*.

Since *wireless* communication is the distinguishing feature of wireless sensor networks it is given special attention.

Based on the requirements and findings a design space exploration platform for wireless sensor network has been conceived. It consists of a hard- and software driven platform. They are referred to as the *Hyperion*- and *Sensim* platform.

Hyperion and Sensim will be used to conduct several case studies to demonstrate their capabilities. The tasks range from (cycle accurate) power measurements, sensor node system-on-chip (SoC) optimizations to design flow extensions.

After the case studies the conclusion follows where the thesis is summarized. Then the chapter closes with an outlook.

This chapter has introduced the system vision and the methodology followed in this thesis. The following chapter investigates the design space of wireless sensor networks. Afterwards the reader will have a better understanding of the vast number of design choices involved in wireless sensor networks.

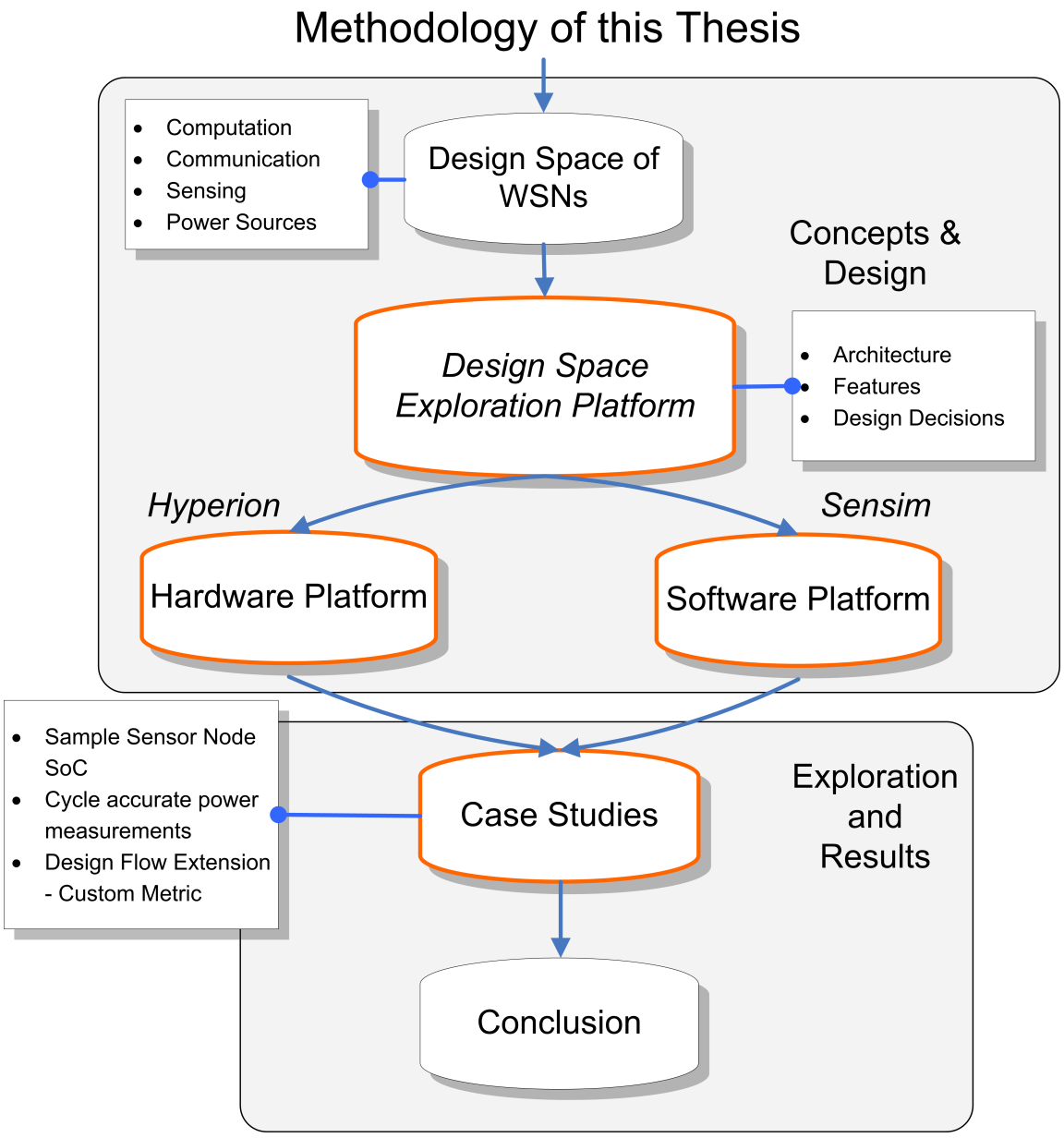



Figure 3.4: The figure shows the methodology followed in this thesis. The design space for wireless sensor networks will be investigated to outline the range of design choices. The design domains computation, communication, sensing and power supply will be introduced. The focus - in this thesis - is on computation and communication. Based on the requirements a design space exploration platform will be conceived which consists of a hardware- and software driven platform. Several case studies will be carried out. They encompass sensor node hardware architectures, architectural optimizations, power measurements and a design flow extension to compute a custom metric.

# 4 Design Space of WSN

The design space of wireless sensor networks is vast. This chapter gives an overview over the design space of wireless sensor networks which ranges from the network- to the node-level where hard- and software components for sensing, computation, communication and power supply have to be chosen.

Especially, advanced sensor nodes which have been introduced in the previous chapter enlarge the design space. Compared to basic sensor nodes they have additional hardware components like wide band radio transceivers and high-bandwidth sensors which require sufficient computational power and memory.

The focus in this thesis is on the design space exploration of computation and communication. For the sake of completeness other aspects like sensing and power supply are included as well. The communication layers are dissected and discussed in depth.

The chapter closes with a summary.

## 4.1 Overview: Design Space of WSN

Sensor networks have a wide range of applications as we have seen in the first chapter. Both the *network*- and the *sensor node*-design must match the application's requirements in *communication, computation*, *sensing* and *power supply*. The focus of this thesis is on the computational- and communication domain.

The design of the computational domain is an embedded, computer architectural challenge with a focus on low power design. The storage and buffering of data is also accounted to the computational domain in this thesis.

The distinguishing feature between *embedded systems* in general and *wireless sensor networks* specifically, is the *wireless* communication. Therefore, wireless communication is given special attention is this chapter.

### 4.1.1 Communication - Radio Transceiver

Figure 4.1 shows that a sensor network may be composed of different sensor node- and radio link-types. Certain properties of a radio link are decided at *design time* and some can be changed at run time through re-configuration.

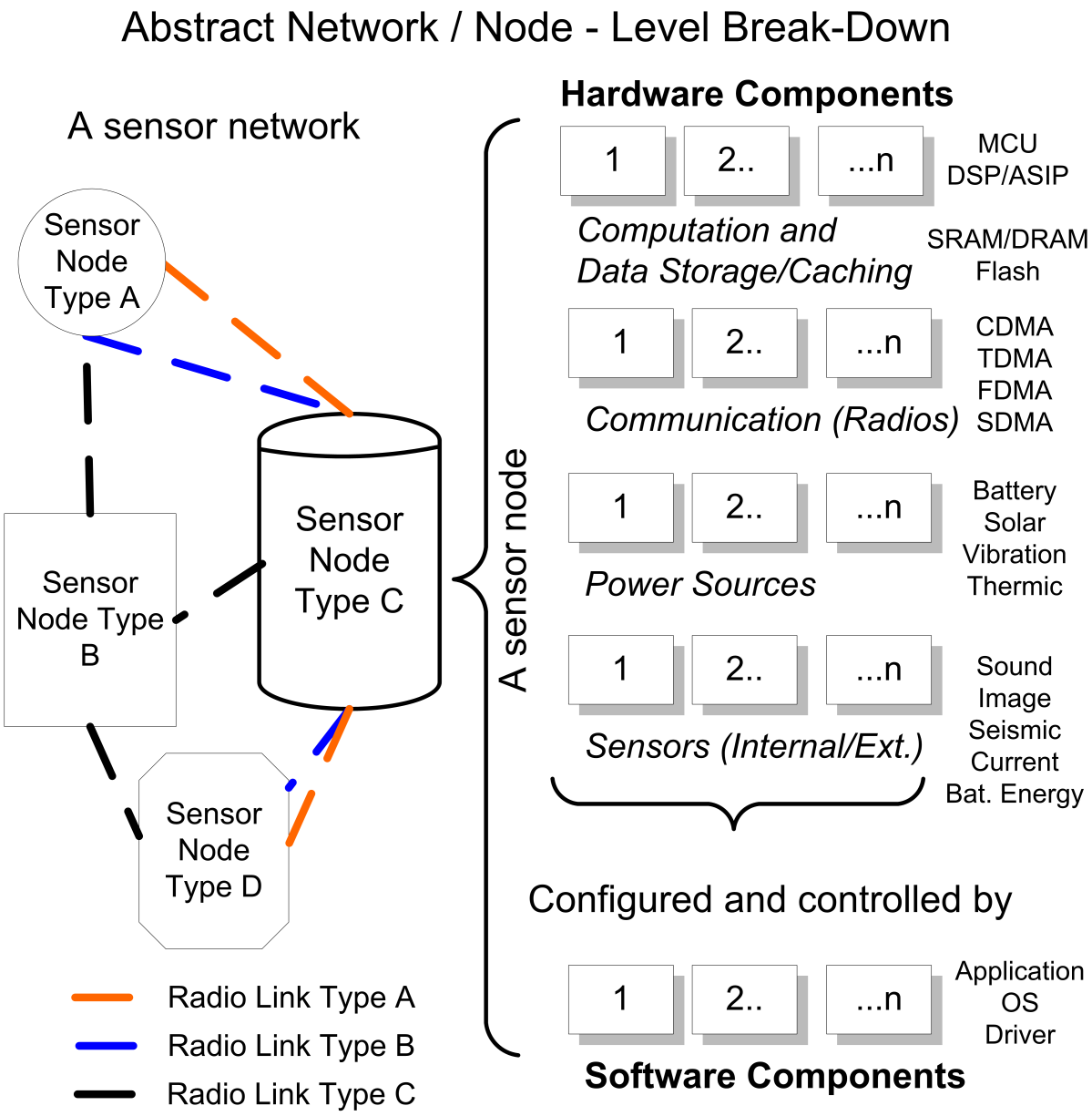



Figure 4.1: The design space of a sensor network and its nodes is vast. The network can be composed of different sensor node- and radio link-types. An individual sensor node can have various processing elements for computational tasks as well as memories to buffer and keep data. The different radio link types are realizable by multiple radios and their configuration settings. It is also conceivable that a software defined radio implements multiple radio link types - even simultaneously. Depending on the power needs and environmental conditions different power sources may be applicable. The choice of sensors is subject of the application requirements. All hardware components have in common that they are control- and configurable by software.

The flexibility of the radio transceiver determines the exact partitioning:

Low cost transceivers are often very rigid and can only vary a few parameters e.g. the output power. Due to the analog nature of the radio circuitry and the antenna it is often the case that the frequency range is constrained.

Only *software defined radio* [137] transceivers can vary most parameters at run time. This immense flexibility incurs extra run time overheads due to the software based signal shaping and filtering. Additionally, the hardware must be capable of receiving and sending signals over a wide spectrum. This involves special filters, amplifiers, high-speed A/D-D/A conversion and broadband antennas.

Thus upfront, software defined radio (SDR) transceivers are likely to be much more power intensive than conventional radio transceivers. Conventional radio transceivers can be built from optimized analog parts; for example highly selective antennas and filters.

Nevertheless, it is conceivable that a carefully balanced SDR radio which exploits its configurability to adapt to environmental conditions can amortize its higher power costs. SDR is interesting from a research perspective since computation costs can be traded for gains in radio communication performance [91].

More realistic - however - is a sensor network which standardizes around one or two conventional radio transceivers. For example: a narrow-band and wide-band radio transceiver. They need not be in every sensor node though.

Narrow band radios are suitable for control- or low data rate sensing-tasks. A wide band radio is appropriate if high data rate sensor data must be forwarded frequently. A good example is a camera tracking sensor network where images of the tracked object must be quickly exchanged between involved sensor nodes.

### 4.1.2 Computation - Processing Elements

Sensor- and arriving-data (e.g. radio packets) must be processed. For power efficiency it is important that the designer picks a suitable processing elements for each task.

Similar to software defined radios there is a choice between optimized but fixed (DSP / ASIP[1]) - and configurable processing elements (CPLD, FPGA).

For standard signal processing tasks like filtering, FFT and signal synthesis a wide range of digital signal processors (DSPs) are available. Many of them have special support for video- or audio-applications. It is imaginable to use FPGAs in sensor nodes but they will always be less power efficient than ASIC implementations.

FPGAs may be attractive for two reasons though: firstly, for a small amount of sensor nodes they are cheaper and secondly it may be possible to exploit their run-time reconfigurability

[1] ASIP=Application Specific Instruction Set / ASIPs are processors with application specific instruction set extensions. A DSP specialized at processing image data can be considered an ASIP processor.

to offset their high power consumption. Swapping coarse grained (fixed) function blocks in-an out of a small FPGA is a good example for saving power. Bigger gains are more likely if a fine-grained hardware can be assembled at run time to handle many (unforeseeable) operating conditions. In a camera tracking sensor network - for example - a hardware module could be composed at run time which matches a particular image pattern very efficiently.

FPGAs though very interesting as a reconfigurable computing fabric are out of scope since this thesis is concerned with design space exploration at design time - not at run time.

Currently, most sensor nodes are based on microcontrollers or DSPs. Depending on the sensor nodes capabilities they range from 8 to 32 bits (see pages 12, 14 and 15). Except for the tiniest sensor nodes it is common to have more than one processing element (e.g. SunSPOT page 15 and WINS-NG on page 68).

The task mapping is usually static: an application processor, a power management processor, a signal processing processor and a protocol processor. Most sensor nodes have a combination of two and not all of them.

In the related work a number of custom sensor node SoCs (system-on-chip) where referenced on page 60. These designs had a special focus on ultra low-power -data rate radio communication. Besides radio communication it is conceivable that future sensor node SoCs provide other high-level architectural features to support routine tasks in sensor nodes.

One example would be automatic coordination of sensor data acquisition and dissemination in the network without processor involvement. Another opportunity could be off-loading network maintenance tasks from the MAC- and routing-layer into dedicated hardware.

All operations require access to some form of memory. Therefore, a lot of challenges are to be found in memory access optimizations. For example: specialized low power caches for data- and instructions *or* automatic scratchpad memory management.

The following section looks into the memories itself and outlines their design space parameters.

### 4.1.3 Computation - Data Storage and Caching

A sensor node has various requirements to its memories. Non-volatile memory is required to store program codes, configuration data, event-logs, (accumulated) sensor readings and part of the system state.

Volatile memories are needed as scratchpad or cache memories during active phases and may be put into data retention modes to bridge (shorter) phases of inactivity (see page 25 for example).

A wide variety of memory types are available: e.g. SRAM, DRAM, NOR/NAND-FLASH,

MRAM[2], FRAM[3] or have been announced such as: PRAM[4], NRAM[5] and PMC[6].

Memories vary in power consumption for standby/active, read/write[7] accesses and have different power-on/off transition times, capacities and volatility. By cleverly managing the placement of data at run time, performance and power consumption may be matched with the characteristics of the different memory types.

Current technology already allows more than eight (memory) dies to be combined into a single package (MCP = Multi Chip Packages). Other approaches combine different memory types on the same die *or* on the top- and bottom-side[8]. The differences in packaging influence the physical dimensions and the performance of memory accesses (bandwidth, latencies) as well as power consumption.

Some novel memory types are *especially* interesting for sensor node designs. The *SRAM-like* memories (MRAM, FRAM or PRAM to mention a few) could be used to keep the state between *duty cycles*. Since those memories - unlike conventional SRAM - are non-volatile they can be easily powered down to save power.

In the related work the concept of *duty-cycling* has been introduced on page 14 et seq.

In the discussion of the SunSPOT design ("SunSPOT - Power Consumption and Efficiency") on page 17 it became obvious that a single, *volatile* pseudo SRAM memory (which keeps the entire system's state), exhibits the highest power consumption in sleep mode. A hardware- and software architecture which is able to exploit features of different memory types may be able to perform as good or even better.

### 4.1.4 Sensing

Generally, the choice of sensors and actuators which interface with the environment are *highly application dependent*.

Traditionally, general purpose sensor nodes have few or no on-board sensors (see pages 12, 14 and 15 ff) but they usually provide connectors for plugable sensor boards.

A variety of sensor technologies and types are available for sensor networks:

MEMS (Micro-electromechanical systems) based sensors are deemed crucial for the adoption of *large scale sensor networks* due to their expected advantages in size, power consumption and especially cost. Ultra-low power acceleration- and pressure sensors are already in mass

[2] MRAM = Magneto-resistive Random Access Memory `http://everspin.com/`
[3] FRAM = Ferroelectric RAM `http://www.ramtron.com`
[4] PRAM = Phase-change memory
[5] NRAM = Nano-RAM
[6] PMC = Programmable metallization cell
[7] For FLASH memory - for example - the number of writes is limited due to the wear.
[8] Samsung Flex-OneNAND and OneDRAM / BeSang

production [117, 118]. More futuristic applications of MEMS include optical communication with lasers and mirrors [143].

Smaller scale networks can still provide a wide coverage with more power-intensive and costly sensors. An extreme example would be a sensor network composed of satellites.

A more realistic example are camera tracking sensor networks (see page 67). Their image sensors have a variety of settings like frame rate, resolution and color depth. If an image sensor can be tilted and panned then the coverage and tracking performance can be improved even further. In this case even the sensor has a wide design space which has to be explored.

Unlike many simple sensors an image sensor provides a 2-dimensional signal and thus puts higher demands on the computational components and the memory architecture. If image data must be exchanged between nodes then this must be considered in the choice of radio communication as well.

There is virtually no limit in the type of sensors which can be used. Even laser scanners - for example - have already been used in mobile sensor nodes to monitor large areas of interest[9].

### 4.1.5 Power Sources

The designer has to carefully design the power supply system to match the power source(s) with the environment and the application's requirements. The network's life time and the quality of service (QoS) depend on a well-designed power supply system.

Shadrach Roundy (UC Berkeley) wrote in his PhD thesis[10] [114] that the challenge of powering wireless sensor nodes can be tackled in three ways:

1. Improve the energy density of storage systems
2. Develop novel methods to distribute power to the nodes
3. Develop technologies that enable a node to generate or "scavenge" its own power

(3) Shadrach (2003) has published results on solar- and vibration-powered radios [46, 16]. Especially in [16] he has shown that vibration powered sensor nodes are viable if the vibration spectrum which is encountered at deployment can be matched.

Micro generators like micro fuel cells (hydrogen, hydro-carbon or alkaline based e.g.), thermoelectric generators, micro-heat engines or even radio active power sources have been proposed as well. In [128] the authors have shown that it is feasible to power a sensor node with a thermoelectric generator which turns thermal- into electrical-energy.

(1) Currently, most sensor nodes are powered by consumer batteries. Recent advances (2008) in nano-technology may yield a ten-fold increase in battery capacity [18]. This broadens the scope for sensor networks which rely on batteries as their primary energy source.

[9] http://www.bosch-sicherheitssysteme.de/de/systeme/systemloesungen/gelaende/index.htm
[10] A book has been created out of the PhD-thesis [119]

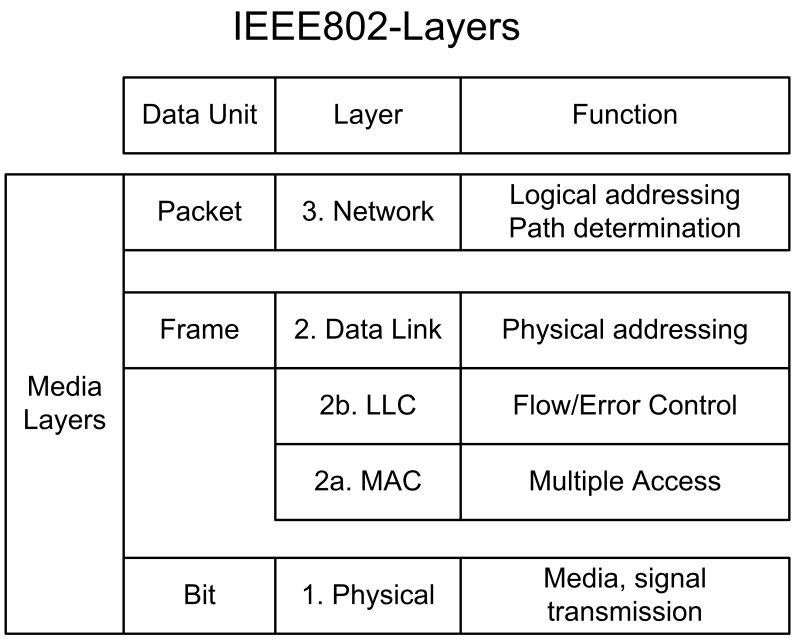



Figure 4.2: The IEEE802 standard divides the media layers into 3 sub-layers. The physical layer is responsible for the bit-level transmission. The data link layer in IEEE802 is split into logical link layer (LLC) and media access control layer (MAC). The MAC layer coordinates the media access whereas the LLC layer deals with flow control and error recovery or -correction. The network layer handles path determination (routing).

Ultra-capacitors have also been considered as power sources for sensor nodes. They have a higher energy density than normal capacitors but less then batteries. Unlike batteries, capacitors can be charged and discharged much more quickly and also more often (millions of cycles).

(2) At the MIT (Massachusetts Institute of Technology, Cambridge) a group of researchers has successfully transferred energy wirelessly over short distances (a 60W light-bulb was lighted over a distance of 2 meters). Their *efficient* approach is based on resonant coupling and does not spread the energy in the free space [9, 76] (2007-2008). A further advantage is that it does not require a free line of sight. Thus it is - for example - possible to power sensor nodes which are submerged in a tank where they monitor chemical reactions for example.

In [109] and [119] the various (design) trade-offs of energy sources and power scavenging are discussed in detail.

## 4.2 Communication Layers

The design of the communication system is crucial to the success since the performance of the radio communication determines the quality of service (QoS) and power consumption of a sensor network.

The design space of a wireless network stack may be structured according to the IEEE802-model as shown in Figure 4.2. IEEE802 encompasses the three lower layers of the OSI-model and splits the (2) second layer into two sub-layers, (2b) and (2a).

The lowest layer is the physical layer (1). It is responsible for the bit-wise transmission over a media (in this thesis wireless).

On top of the physical layer is the (2) data link layer (node-to-node communication) which is composed of the MAC (media access control, 2b) and LLC (Logical Link Layer, 2a). Latter provides multiplexing and flow-control. The MAC-layer *controls* the access to a *shared* physical medium. In a *contention based* network packet collisions may be avoided and detected and in a *channel based* (circuit switched) network multiplexing on the physical layer is employed. Many (real-life) networks have features of a channel- and contention based network .

The network layer (3) handles the source to destination packet routing.

All IEEE802 standardized communication system adhere to this layering (e.g. ZigBee, Bluetooth, WiMAX). Many publications in the sensor network community do not. Nevertheless, the classification is still helpful and necessary. The information compiled in the discussion of the individual layers has been compiled from a numerous number of sources. The interested reader is hereby referred to [73, 96, 47, 100, 144, 115, 30, 17, 65].

### 4.2.1 Physical Layer

Radio signals are inherent analog and must be *modulated* to carry digital signals. This can be done by varying phase, amplitude and frequency of a carrier signal (FSK, ASK, PSK[11]). Besides, these elementary modulation schemes are many more. They vary in spectral efficiency, sensitivity to noise and modulation / demodulation effort.

Before the signals are modulated they are suitably encoded for transmission: C*hannel coding* deals with forward error correction (FEC), error detection and interleaving[12]. FEC codes can be grouped into: convolutional-, block- and turbo- codes. Convolutional codes have lower latencies than block- and turbo-codes. Block codes are fixed length channel codes and can be efficiently implemented in hardware (BCH-codes e.g). Turbo codes are computational intensive but come close to the Shannon limit which means that they are most efficient at transmitting data over a noisy channel.

Communication on the physical layer can be full- or half-duplex. Frequency-division duplexing (FDD) allows two nodes to communicate simultaneously over two different channels (frequencies). Full duplex can also be emulated with time division multiplexing (TDM) and is known as: Time-division duplexing (TDD). Latter is commonly found in sensor networks due to simpler hardware realizations.

[11] Frequency / Amplitude / Phase - Shift Keying
[12] Interleaving makes error correcting codes less susceptible to burst errors.

A good example of a physical layer for wireless sensor networks is the IEEE802.15.4 standard [124]. It is used in ZigBee (see Section 4.2.4)

### 4.2.2 Data Link Layer

The media access control in the data link layer controls access to the shared physical medium. In a *contention based* network, packet collisions may be (1) avoided and detected and in a (2) *channel based* (circuit switched) network *multiplexing* on the physical layer is employed.

(1) In wireless networks the common *channel access* methods are CDMA (code division multiple access), FDMA (frequency division multiple access) and TDMA (time division multiple access. Theoretically, they have the same spectral efficiency. However, their implementation offer many different trade-offs:

In CDMA (coded) signals are *spread* over a frequency range and can be transmitted simultaneously. A desired signal can be recovered by applying a unique *code sequence* to the signal mixture. This requires that each signal has roughly the same power level at the receiver(s). Another challenge is the susceptibility to the near-far problem where a remote transmitter is blocked out by a close one. An advantage of CMDA is that it's spread-spectrum property makes it more immune against multi-path fading (see also 37). A further, advantage of (asynchronous) CDMA is that it does not require channel access coordination when (many) nodes transmit infrequently and overall interference remains low.

In FDMA the spectrum is divided into multiple channels. The challenge here is to leave enough space (guard band) in between due to non-ideal filters and frequency shifts (e.g. Doppler shift caused by moving nodes). The frequency separation makes FDMA less sensitive to the near-far problem but requires efficient filters to separate the channels sufficiently. OFDM (orthogonal frequency-division multiplexing) uses multiple (orthogonal)[13] channels to transmit data. Narrow band distortions can thus be compensated by channel interleaving.

TDMA uses a single channel and organizes the access by dividing the access in time slots which requires precise synchronization. A guard time is allotted between slots to compensate for small timing deviations. An advantage of TDMA is that a node may sleep between their slots.

In TDMA the time slots must be managed, in FDMA the channel usage and in CDMA the code assignment. All three methods may also be combined in various forms.

For the sake of completeness I would like to mention that the channel access can also be divided spatially which is known as SDMA (space division multiple access). This requires directional antennas or *beam forming*. Beam forming can be realized with antenna arrays.

(2) In a *contention based* network packet collisions may be avoided and detected. A well known and widespread protocol is CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Before transmission the sender listens on the channel and proceeds only if the channel is free. However, it is still possible that two nodes start transmitting simultaneously. If a collision

[13] This property allows OFDM to be efficiently implemented using the fast Fourier transform (FFT).

is detected then the involved nodes wait for a random period before they engage in the retransmission. CSMA/CA is often found in sensor networks alongside with channel multiplexing.

Examples: WiseMAC [45] is an energy efficient CMSA based MAC protocol for sensor networks. Like many other protocols nodes periodically listens for preambles. The novel idea behind WiseMAC is that the listening schedule is propagated between sensor nodes to dynamically scale the preamble length. SyncWUF [121] is based on WiseMAC and aims to perform better by cutting down the wake-up preamble lengths even further. If the wake-up schedule of a neighbor node is up to date then a short preample is sufficient. Otherwise SyncWUF uses multiple short signals. These wake-up signals - unlike a normal preample - carry more information to aid the wake-up process. S-MAC [112] - from the University of Southern California is not a pure MAC protocol since it also takes over functionality usually found in other layers. In addition to media access it engages in topology and synchronization. In S-MAC nodes listen for a certain time before they decide to become a synchronizer or follower. Latter, follow a schedule broad casted by the synchronizer. S-MAC saves power by periodically alternating the synchronized nodes between sleep and active mode. B-MAC [105] from the University of California, Berkeley is an adaptive MAC protocol which supports collision avoidance (CA). If a network is unloaded the duty cycle can be stretched to save power at the cost of increased latency and lower throughput.

### 4.2.3 Network Layer

On the network layer the data is packetized. When a packet is ready for transmission on the physical layer it is called a frame and consists of a preample, a synchronization word, a length field, an address field, a data-field and a check-sum (see for example [67]). In [155] it is shown that the packet reception rate is theoretically higher for smaller packets. This depends - however - on the error correction as well as the signal-to-noise (SNR) ratio. One disadvantage of smaller packets is the reduced ratio between (useful) payload and header fields.

Once a packet has been assembled it can be routed through a (multi-hop) network from its source to its destination. The challenge in routing is to find a suitable path. Routing can be centralized or decentralized. Latter is coherent with the characteristics of a sensor network but may lead to local and thus sub-optimal decisions. The topology is an important factor. If the network is fixed static routing can employed. Dynamic routing can handle networks with frequent node failures and mobile nodes. Therefore, dynamic routing requires a metric to guide the path selection (e.g. latency, load). Changes in the network can be propagated proactively or re-actively. Latter is better in networks which are mostly static.

Examples: Wendi Heinzelman from the Massachusetts Institute of Technology has published two well known routing algorithms in the wireless sensor network community: SPIN [58] and LEACH [59]. Deborah Estrin's (Professor at UCLA and director of the CENS[14]) students have contributed directed diffusion- and rumor routing [68, 14]. TEEN [92] is an example for a reactive routing protocol.

[14] Center for Embedded Networked Sensing `http://research.cens.ucla.edu/`

For a camera tracking network geo-routing protocols are an interesting choice [70]. LAR (Location-Aided Routing) [78] is a reactive protocol, LAAR (Location-based Adaptive Ad-hoc Routing) [151] forms a cluster based hierarchy and GOLI (Greedy On-Demand Routing Scheme Using Location Information for Mobile Ad-Hoc Networks) [98] - unlike LAR - does not flood the network and thus reduces the number of messages. GPSR (Greedy Perimeter Stateless Routing for Wireless Networks) [77] uses an greedy algorithm to make local routing decisions based the neighbors position. Thus it can quickly recover from topology changes. PSGR (Priority-based Stateless Geo-Routing in Wireless Sensor Networks) [150] is a volunteer forwarding routing algorithm, where the packet holder does not decide who forwards next. Instead nodes volunteer depending on their geographical position.

### 4.2.4 Example: ZigBee and IEEE 802.15.4-2007

The enumeration of radio design choices in the previous sections ought to have made clear that the design space for a wireless radio communication is vast. Thus wireless communication can be tailored in many details to the specific requirements of an application.

For general purpose, low-cost and low-data rate sensor networks the ZigBee specification suite[15] has been proposed:

It uses the IEEE 802.15.4-2006 standard [124] which specifies the physical- and media access control layer. The (channel) access method can be time slotted or CSMA/CA[16] based. The data rate, modulation and number of channels depend on the frequency range (868-868.8 MHz, 902-928 MHz, 2400-2483.5 MHz). The maximum data rate is 250 kbit/s.

For increased robustness DSSS (direct sequence spread spectrum)[17] is used to reduce the effects of narrow band distortions. In DSSS the signal is spread over a (wide) frequency range. Besides DSSS, two other spread spectrum technologies have been included in the latest version: ultra wide band (UWB) and chirp spread spectrum (CSS). These two enable localization to be realized as part of the radio link due to their special physical properties. For applications which require localization this may be the deciding factor.

Of course, RSSI based localization is still possible with the currently assigned frequencies [23, 24].

IEEE 802.15.4 specifies two topologies: star and peer-to-peer. The star topology requires a central node (= limited scalability). The peer-to-peer topology can have an arbitrary shape and requires routing. A routing algorithm is not (yet) specified in IEEE 802.15.4.

However, two routing protocols seem to be associated with ZigBee[18]: ad-hoc on-demand distance vector (AODV) [101, 103] and neuRFon[19]. They are not referenced in the current spec-

[15] `http://www.zigbee.org`
[16] CSMA/CA=Carrier Sense Multiple Access/Collision Avoidance
[17] DSSS is also used in CDMA.
[18] According to Wikipedia and various lecture slides on the WWW.
[19] `http://www.motorola.com/content.jsp?globalObjectId=290`

ification. *Thus the vendor-interoperability of multi-hop enabled sensor nodes is unspecified in ZigBee.*

It remains to be seen if the (interoperability) features and the customizability of ZigBee fit the user's needs in the long term. An undisputable advantage is that a standard enables 3rd-party tool support[20]. Examples of sensor networks that use ZigBee can be found in [19, 20, 148].

## 4.3 Summary

The previous sections (4.1.1-4.2.4) have shown that the design space for a sensor-node and -network is quite *vast:*

We have seen that in the computational domain the designer has to pick suitable processing elements and (on-chip) interconnects as well as memories for caching and storage. Especially memories come in a variety of different technologies (SRAM, FLASH, DRAM, MRAM[21] for example) - each with its on unique characteristics.

Signal processing requires processors with numerical units. For control simpler processing elements are often sufficient. Besides functional aspects it is often also important to consider non-functional aspects like the maturity of a compiler.

The design space of the communication subsystem has been divided into three layers: physical, data-link and network layer. The separation is not fixed. So called cross layer optimizations cross the boundaries and often promise power savings.

If a WSN-standard like ZigBee - for example - has been chosen then many parameters of the communication domain are fixed and cannot be subject of the design space exploration process. The advantage of a communication standard is (a higher chance of) interoperability. It depends on the application requirements if a standard has to be considered.

Another industry consortium[22] which shall be mentioned here too, even promotes the usage of the Internet protocol (IP) in control- and sensor networks. The Internet protocol has certainly not been designed with sensor networks in mind. Its ubiquity - however - suggests that sensor nodes can be integrated smoothly with existing infrastructure ( = functional requirement ). Again, the requirements may justify possible inefficiencies in the communication domain if a standard must be used.

The sensing domain is highly application specific and has implications on the computational and communication domain which must process and communicate the sensed data. A good indicator for the required performance (computational- and communication domain) of a sensor node is the sensor's data rate. Since the acquired data is likely to be stored, processed or streamed into the network at a similar data rate.

[20] `http://www.opnet.com` (ZigBee MAC-Layer)
[21] Magnetoresistive Random Access Memory
[22] `http://www.ipso-alliance.org/`

Last but not least, the power supply subsystem and its sources are important in regard to the network's performance and life time. Sometimes energy can be scavenged from the environment. In such cases the design may become dependent on the environmental conditions. A sensor network that scavenges solar energy - for example - could exhibit a day time and weather dependent performance profile.

A designer may easily be overwhelmed with design decisions. Not only due to their sheer number but also due to their manifold interdependence. There is a strong relationship between genericity, specialization and power efficiency in WSNs. The designer must find a balance to meet the requirements.

Unlike the automotive or mobile phone industry there are no well established segments and platforms (yet) which could serve as baseline.

The following chapter presents a flexible design space exploration platform for wireless sensor networks which aids the designer by providing feedback on power consumption among other features.

The design space exploration platform allows the computational domain to be freely defined without changing the hardware physically. Thus different hard- and software architectures can be explored within a reasonable amount of time.

The communication domain may be explored in simulation or by interfacing with real radio transceivers. If a real radio transceiver is used then it becomes possible to connect to a real sensor network. Subsequently, the communication performance can be assessed under (more) realistic conditions.

One of the most important aspects: the power consumption of a sensor node, can be *estimated* in simulation and *measured* when real hardware components are interfaced. The design space exploration platforms supports the exploration of the computation-, communication- and sensing-domain.

The concepts, algorithms and mathematical underpinnings of the design space exploration platform are covered in the following chapters.

# 5 A Flexible Design Space Exploration Platform for WSNs

In this chapter the design space exploration platform is introduced.

First, the *overall* design flow is presented. Then the design space exploration platform, which is integrated into the design flow. The platform consists of a hard- and software driven part. They allow the designer to build models of wireless sensor networks with real- and simulated hardware components.

The hardware driven platform is called *Hyperion*. It can be extended with exploration modules. The exploration modules are equipped with real hardware components. The memory exploration board *Theia* (for advanced sensor nodes) is a good example for a custom exploration module. Theia has been designed to enable the exploration of the memory sub-system.

The flexible hardware driven platform has a comprehensive power measurement support. The measurement setup is explained in detail as well as the mathematical background behind it. The power measurement sections cover average- and cycle accurate-power measurements. Cycle accurate power measurements are a special feature of Hyperion (the hardware driven platform) which allows the power consumption of a single clock cycle to be measured.

The software driven platform *Sensim* complements the hardware driven platform. First, Sensim's flexible design flow is introduced then its architecture. The architecture of the software driven platform splits node- and network-level simulation. A key feature of Sensim is the division of simulation and evaluation. The benefits are extensibility and flexibility. The energy estimator within Sensim - for example - is decoupled from the simulation component. The energy estimator computes hierarchical energy break-downs. The energy break-downs can be organized in designer defined categories. Categories may be routing or signal processing for example.

This chapter is the basis for the case studies in the following chapter. It ends with a summary.

## 5.1 Overall Design Flow

Before we delve into the architecture, concepts and design decisions related to the hard- and software driven exploration platform, it is necessary to understand the overall design flow. Figure 5.1 visualizes the iterative design space exploration process. The designer is mainly con-

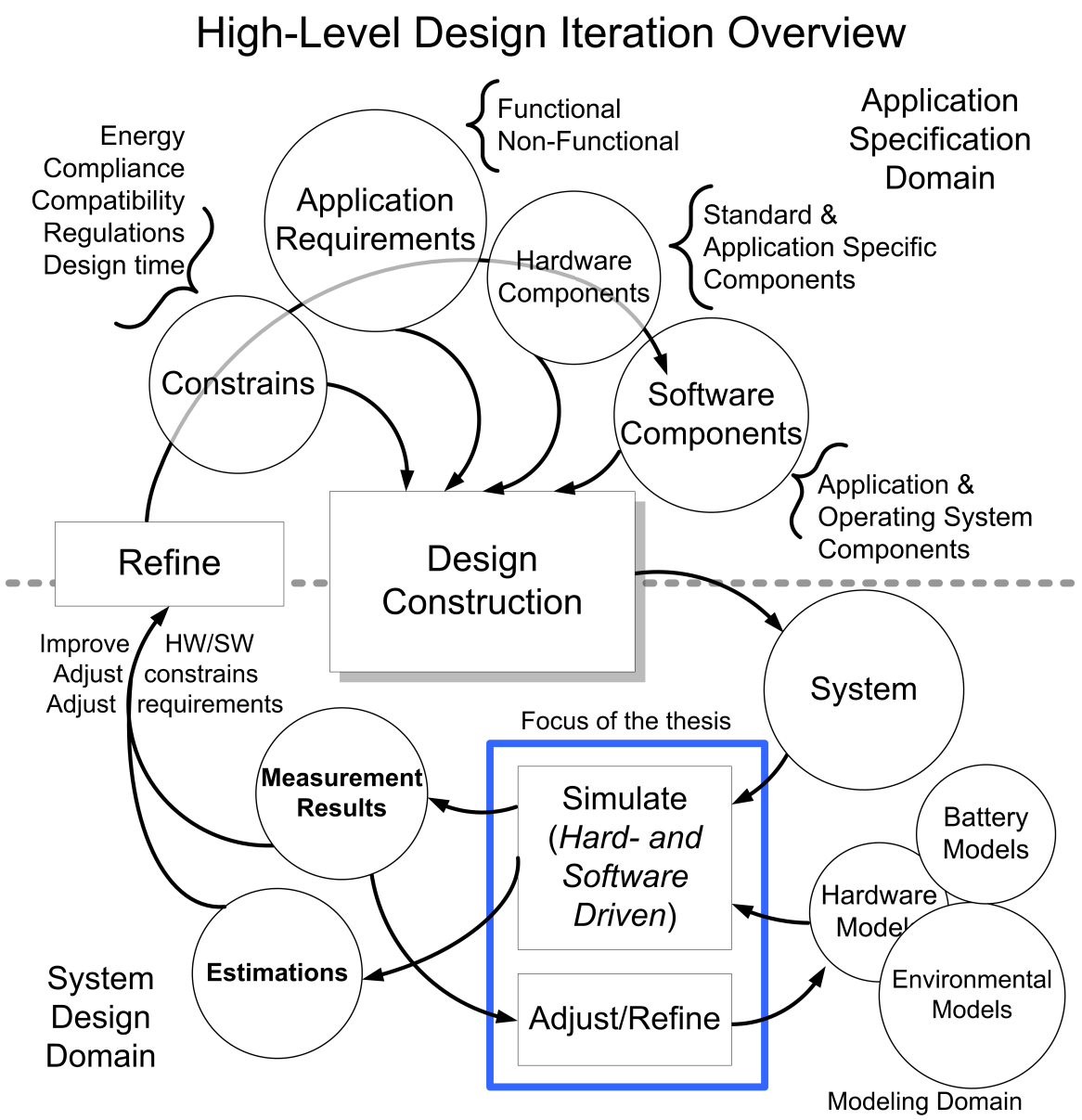



Figure 5.1: The figure illustrates the high level design iteration process. The designer is mainly concerned with two domains: the *application specification*- and *system design*-domain. If *new* hard- or software-components are introduced then it is necessary to adjust and refine the simulation models (*modeling* domain). Once the application has been specified a system may be constructed and can be assessed by the hard- or software driven exploration platform. Both approaches have trade-offs in accuracy, flexibility and simulation speed. The designer has to decide which aspects need to be evaluated at a certain design phase. The measurements obtained with the hardware driven platform are used as input for the software driven simulation platform. These inputs are needed to adjust parameters and to refine simulation models. The designer has to adjust the constrains and parameters of the application or may also refine the hard- and software components.

cerned with two domains: the *application specification*- and *system design* domain. The *modeling* domain encompasses simulation models of sensor node components (processor, battery - for example) and environmental phenomena (terrain, sun ray intensity, radio wave propagation - for example).

In the application specification domain the designer must provide the system constrains like available energy - for example. But also the application requirements and specifications of hardware- and software components. This includes interface and behavioral specifications.

In this thesis it is assumed that the *energy constrains* are key to the success of a wireless sensor network design. Constrains imposed by regulations like limited transmit power *or* compatibility with the IP-protocol/ZigBee *or* available design time *or* usage of existing infrastructure are *not* a concern in this thesis.

Application requirements can be divided into *functional* and *non-functional* requirements (see Figure 5.2 to see where various requirements may map to).

Non-functional requirements are for example: cost, safety, security, reliability, performance / power efficiency and scalability. Many of the self-x properties apply to sensor networks as well[1].

In hardware non-functional requirements often require trade-offs between area and performance and in software components between performance and memory consumption.

The functional requirements specify what the user wants the system to accomplish. For example to track objects like in the central application theme on page 65 et seq. Another example is the required indoor/outdoor-range of the wireless transceiver.

To differentiate between the requirements it is advisable to prioritize them so that a designer or design tools can consider trade-offs. A so called *traceability matrix* can be constructed to visualize the mapping of requirements to design artifacts. This helps to identify interdependence and completeness.

Back to the figure: Components in the application specification domain can be hardware- or software-components or both:

Hardware components specifications in the figure are either descriptions of physical chips or non-tangible IP-cores like processors, interconnects or memories. Software components specifications describe the behavior and interfaces of software modules. Common sensor node software modules provide compression, packet handling, routing, application logic and system software. Often hard- and software components form an unit when hardware components come with accompanying software drivers for configuration and control.

An operating system is also a component. It acts as a *run-time* integration platform (see page 25 ff). TinyOS or SOS (page 20 et seq.) are good examples for operating systems in wireless sensor networks.

[1] self-(organization, configuration, optimization, healing) and context awareness

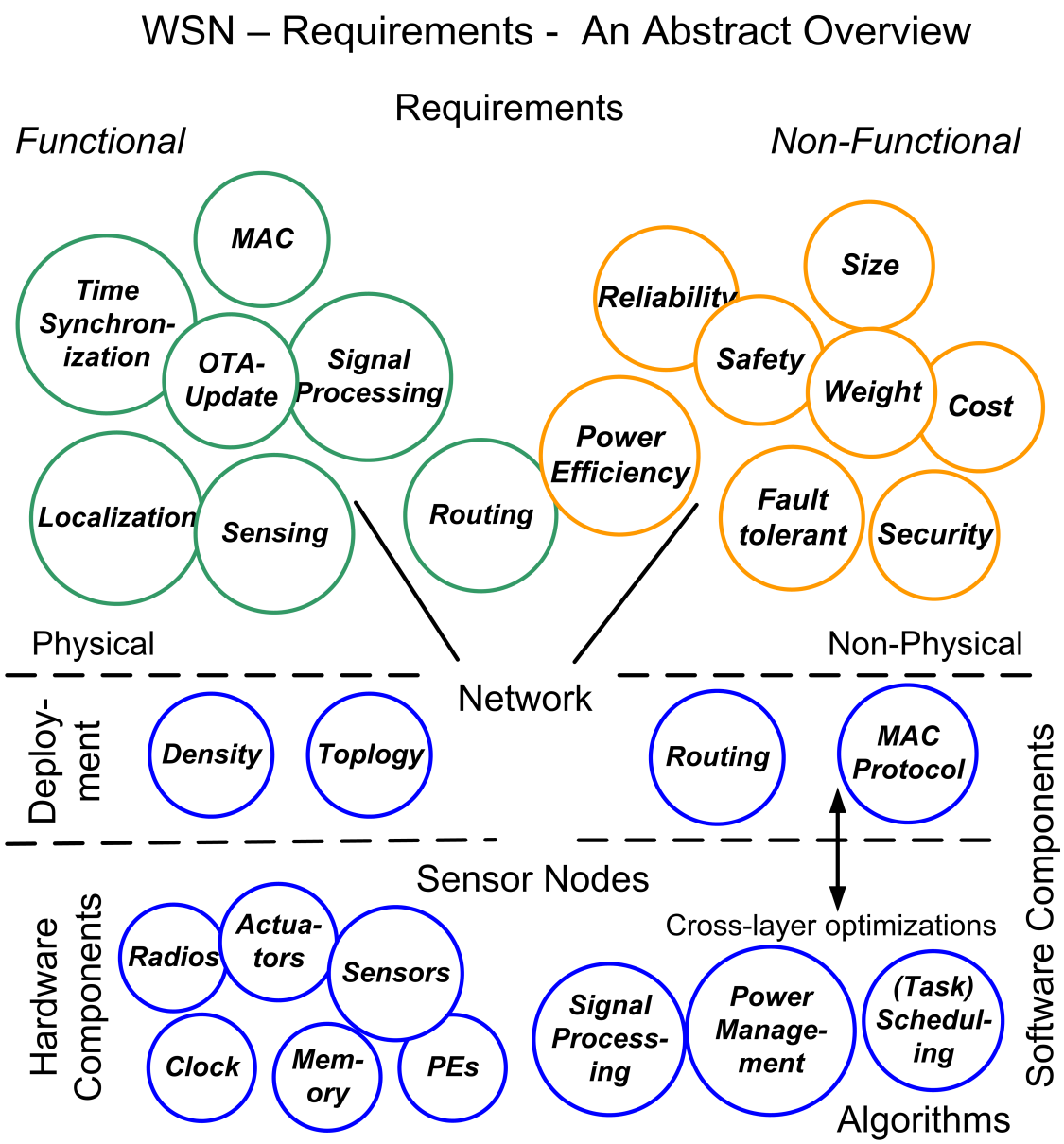



Figure 5.2: The figure shows functional- and non-functional requirements of a wireless sensor network. The requirements may map to the *network*- or *node*-level and may have *physical* or *non-physical*-manifestations. Physical manifestations are the sensor nodes (their hardware components) and the network (topology) they form. Examples are a star or mesh topology. The node density is also an important feature. Examples of hardware components are: radio transceivers, sensors or processing elements (PE). The non-physical manifestations of requirements are software components which take over tasks on a node- or network-level. Cross-layer optimizations may cross the node-network boundary. For example when local power management influences routing decisions or the media access control (MAC) - layer (adaptive algorithms).

The Xilinx EDK[2] is an example of a *design time* integration platform. It assembles parametrizable pre-built and custom - soft- and hardware components into a fully functional hard- and software-system.

Some design time platforms generate hard- and software source code components from higher level specifications. Examples are tools such as LabVIEW[3],MATLAB / Simulink / Real-Time Workshop[4] and Esterel/SCADE[5].

To assemble a system out of components it is important to have component interface specifications. Examples for software component specifications are API-interfaces and for hardware components signal-interfaces.

Back to figure: The (automated) design construction finally yields a concrete *system* which may or may not meet the application requirements and design constrains.

The evaluation of the system is the focus of this thesis. Since power consumption is the crucial metric in wireless sensor networks it is necessary to obtain detailed information about power consumption.

Figure 5.1 shows that the designer may either use hard- or software driven platform to evaluate a wireless sensor network. The system vision on page 69 suggests a simulation based on virtual- and real hardware components. In this thesis the assessment of a wireless sensor network is therefore split into a software- and hardware driven platform.

The software driven design space exploration platform performs a full-system simulation and the hardware driven design space exploration platform uses *real time* hardware-in-the-loop simulation. The hardware driven design space exploration platform can interface with real hardware components.

The system specification abstraction level differs between the hard- and software platform. The software driven platform needs a *functional* hardware specification whereas the hardware driven platform strictly requires a *cycle accurate* hardware specification. It is possible to use a cycle accurate specification for both platforms *or* alternatively the designer may choose to generate a cycle accurate- and functional specification from another high-level specification. This is possible with MATLAB/Simulink specifications for example.

Unlike the hardware specification which may either be functional or cycle accurate there is no difference in the software components - regardless of the platform.

After prototyping (Figure 5.1) the *measurement results* may be used as feedback to re-parametrize or refine the simulation models. If the models are satisfying but the system performance is *not* then the designer must proceed by adjusting the requirements, constrains or improving the hard- and software components.

[2] http://www.xilinx.com/ise/embedded/edk_docs.htm
[3] http://www.ni.com/labview/
[4] http://www.mathworks.com/products/simulink/
[5] http://www.esterel-technologies.com/

The design iterations are carried out until the design space has been fully explored. Or - more likely - a satisfying solution has been found.

## 5.2 Part I : Hardware driven Design Space Exploration Platform

The following section introduces the architecture of the hardware driven design space exploration platform *Hyperion* which has been conceived in this thesis. It enables cycle- and timing accurate simulation of sensor node system-on-chip (SoC) designs without re-spinning a chip- or board design. Unlike a software driven simulator it can interface with real hardware components.

*Hyperion* has actually been built and successfully tested. The schematics, layout and design notes can be found in the appendix starting on page 145.

### 5.2.1 Hardware Architecture of the Platform

The system vision which has been outlined on page 69 et seq. pictured a flexible design space exploration platform that allows the performance and power consumption of a wireless sensor network to be assessed. It furthermore emphasized a *duality* between simulated- and real hardware components.

The hardware driven design space exploration platform *Hyperion* which has been developed as part of this thesis supports the real time simulation of a sensor node system-on-chip (SoC) design. It can interface with hardware components and therefore interact with its (real) environment.

The software driven design space exploration platform (page 106) on the other hand simulates hardware components and environmental phenomena.

Hyperion has been designed to assess the *computational-, communication- and sensing-domain (*see page 73 et seq. for a definition). The focus of this thesis is on the first two domains.

The programmable logic and an (optional) memory exploration module enable the evaluation of the computational domain. Radio transceiver- and sensor exploration modules are for the evaluation of the communication- and sensing domain.

The hardware driven platform does *not* support the exploration of the power supply domain. A programmable and configurable power supply exploration platform - if feasible - would be quite interesting indeed. The exploration of the power supply domain could still be realized within the constrains of the software driven design space exploration platform (page 106 ff).

A high level architecture diagram of Hyperion is shown in Figure 5.3:

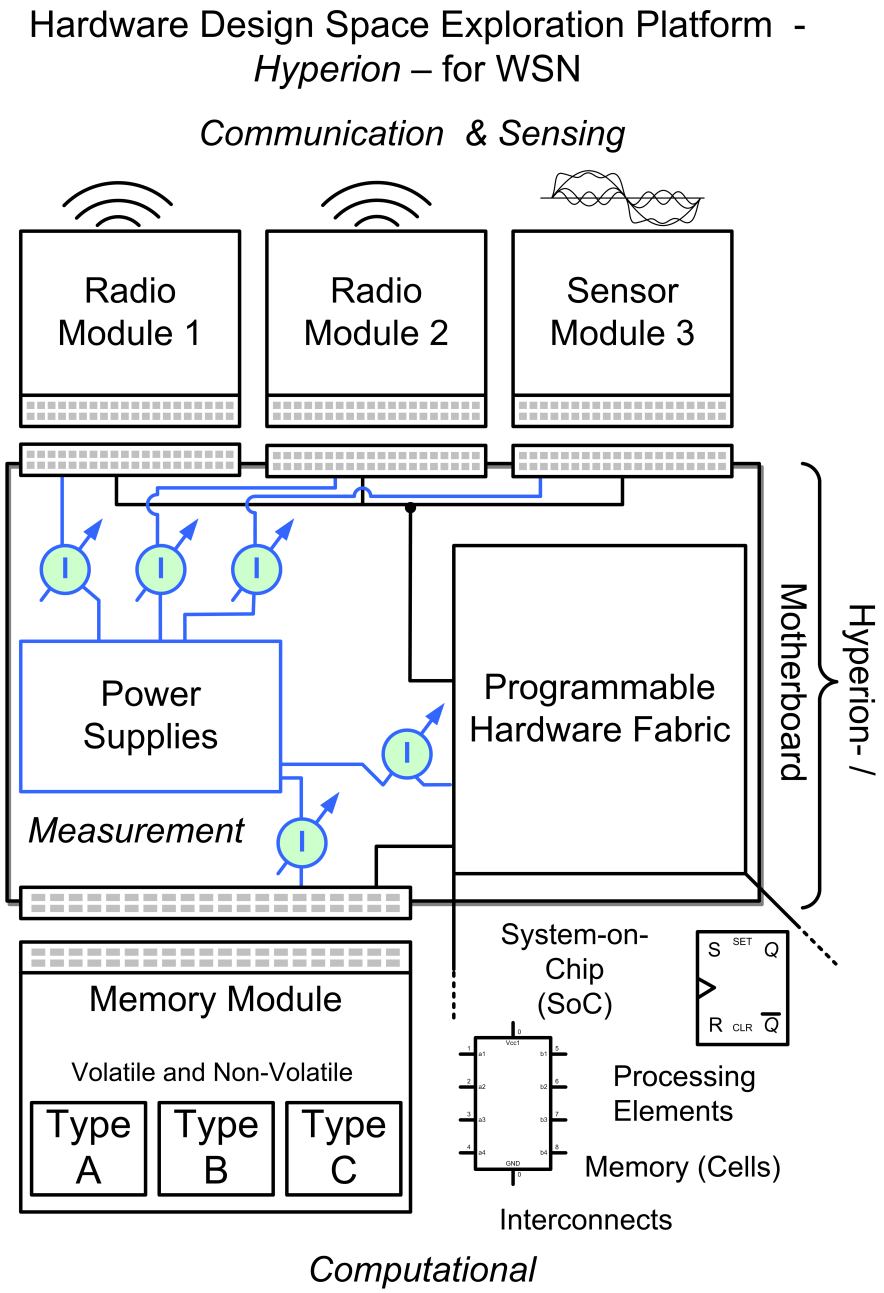



Figure 5.3: The figure shows the high level architecture of the hardware driven design space exploration platform *Hyperion* for wireless sensor networks. The Hyperion board (motherboard) has a programmable hardware fabric. A sensor node system-on-chip (SoC) design may be loaded into the fabric within seconds. There is no need to re-spin a chip or board. Hyperion has extension ports which may be used for radio transceiver-, sensor- and memory exploration modules. The programmed SoC can be executed in real time and interface with exploration modules (*hardware-in-the-loop concept*). Compared to software driven simulation Hyperion offers more *realistic* wireless communication, accurate timing, *real* sensor data and power measurement for *each* component - instead of estimations. More information about Hyperion can be found in the appendix (page 145 et seq.).

Hyperion has a programmable hardware fabric. The fabric holds the sensor node SoC (system-on-chip) design under evaluation. The designs can be changed within seconds instead of weeks or months needed for a silicon chip or board re-spin. The fabric is wired to the extension ports where radio transceiver-, sensor-, and memory- *exploration modules* may be plugged in.

Since power consumption is the crucial metric for the success of a wireless sensor network it must be possible to measure and trace power consumption during the design space exploration. Each component - be it on-board or off-board - has its own power supply to enable undistorted *average* power measurements. The board has measurement resistors and plugs between the power supplies and hardware components so that external laboratory equipment can be attached in multiple ways.

Depending on the wiring configuration it is possible to conduct voltage- or current based power measurements. It is also possible to disable power measurements.

Since every important hardware component can be measured individually it is feasible to obtain an accurate break-down of the system power consumption.

In addition to average power measurement Hyperion supports the *cycle accurate* power measurement of the programmable hardware fabric. Cycle accurate power measurement gives insights into the microarchitectural-level. This is necessary if parts of a sensor node system on chip (SoC) design must be optimized (*computational domain*). An in-depth coverage of cycle accurate power measurement can be found in the related work on page 103 et seq.

A limitation of the programmable logic is that its hardware structure is not comparable with an ASIC. Therefore, the measured power consumption figures must either be scaled or may be compared relatively. For credible scaling it is necessary to acquire sufficient measurements from (a) real sensor node SoC-chip(s).

If a real sensor node SoC-chip is available then it can replace the programmable logic and be measured directly. The measurements would then constitute a *ground truth*.

Due to time- and cost constrains no real sensor node SoC-chip has been realized. The power measurements done with the programmable fabric have proved that the on-board circuitry and the measurement setup work as expected.

If a sensor node SoC *or* parts of it were to be implemented in programmable logic then Hyperion could obviously be used *as is* to assess the power consumption.

In Figure 5.3 the upper part shows the communication- and sensing-domain. Here the radio transceiver- and sensing-exploration modules may be connected to the Hyperion board. For the designer it is important to assess the communication performance under realistic conditions since simulation of radio communication is still not accurate enough to omit measurements.

A detailed discussion of the challenges encountered in radio communication can be found in the related work on page 37 et seq. The vast design space of wireless communication domain has been outlined on page 73 and a detailed discussion of the communication layers can be found on page 79 et seq.

Depending on the radio exploration module plugged into Hyperion the designer has a certain set of run-time configuration choices which can be set by the application- and system software components. Since each connector of the Hyperion board has its own power supply and measurement circuitry it is possible to measure the power consumption caused by communication directly.

A flexible ultra low power narrow-band transceiver has already been successfully used with Hyperion. More information on the current radio transceiver exploration module can be found in the appendix on page 146.

Like for radio transceiver exploration modules, it is possible to connect almost arbitrary sensing exploration modules to the Hyperion board and measure their power consumption too.

The design space exploration platform has been targeted at *advanced sensor nodes* (see page 67 for a definition). A camera tracking application has been chosen as a *central application* theme (see page 65 ff). Therefore a sensor exploration module with an image sensor has been interfaced with the Hyperion board. The experiments with the camera exploration module have shown that non-trivial sensors with high data rates can be successfully handled by the platform. More information about the image sensor exploration module can be found in the appendix on page 146.

A detailed table which compares the hardware driven design space exploration platform and standard sensor nodes can be found on page 167 - also in the appendix.

The computational domain which has been outlined on page 75 et seq. also includes memories. An important point which had been made is that almost all operations require access to some form of memory. It is obvious that memory access optimizations are crucial to the power efficiency of microarchitectural design.

The following section covers the memory exploration module *Theia* which has been designed to facilitate the design space exploration of different memory types in advanced sensor nodes.

### 5.2.2 Hardware Architecture of the Memory Exploration Module

The design space of memories belongs to the computational domain (see page 73 for an overview of the domains) which is besides the communication domain, in the focus of this thesis. Thus the computational domain encompasses processing elements (e.g. processors) and memories. The processing elements are handled by the programmable logic of Hyperion which has been introduced in the previous section.

The *Theia* memory exploration module (for advanced sensor nodes) holds multiple memories and features a programmable wiring fabric to connect them.

The architectural diagram of the memory exploration module Theia is shown in Figure 5.4. Two versions have been built. They have been equipped with different memory types. The schematics and other material related to Theia can be found in the appendix on page 158 et seq.

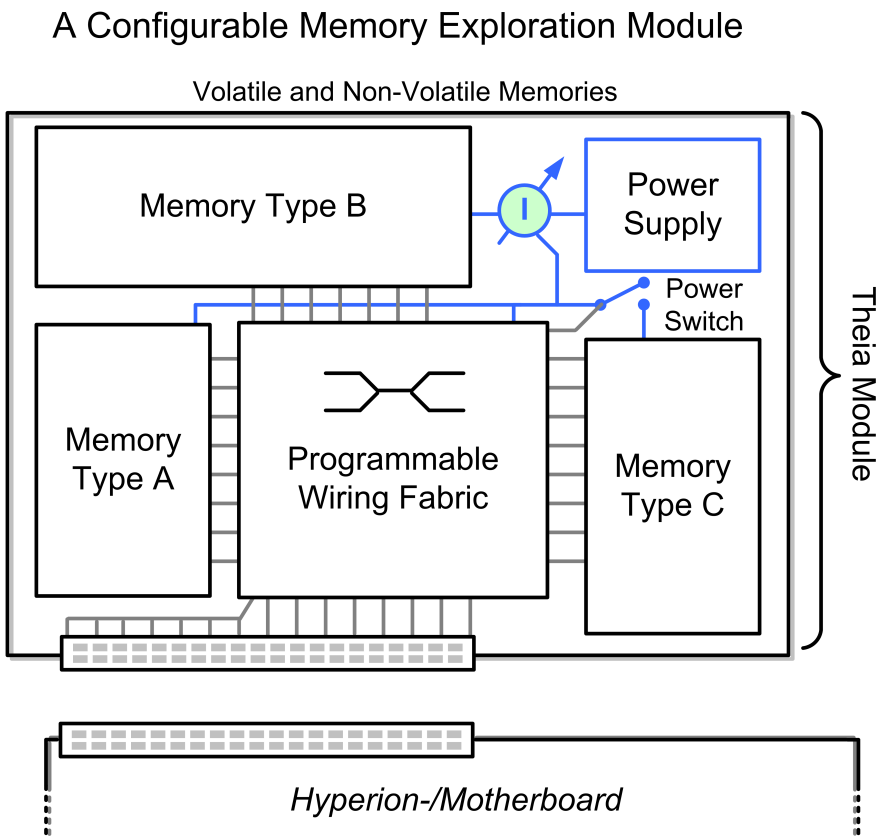



Figure 5.4: The figure shows the Theia memory exploration module. Theia can be equipped with three different memory types depending on the number of control-, bus- and address-lines. Two versions with different memory types have been built. They have MRAM, FRAM, NOR-FLASH and low-power SRAM (see the appendix on page 158). The memories are connected to a central *programmable wiring fabric*. This allows the designer to evaluate different wiring topologies without a chip or board respin. Theia has on-board circuitry to support on-board and off-board power measurement with laboratory equipment. The power supply is on-board so that Theia can be used with arbitrary motherboards - not only Hyperion. Some of the memories can be powered down with *off-chip switches* if comparable performing switches are not available in the memory chip itself.

A Theia memory exploration module can hold up to three different memory types. The focus is on (*upcoming*) non-volatile memory types like FRAM, MRAM and (high density) NOR-Flash since those can be selectively powered-down in a sensor node to save power. Some memories can be switched off with external power switches if no adequate internal switches are available.

In order to quantify the power savings Theia has support for on-board and off-board power measurements. The off-board power measurements are carried out with professional laboratory equipment. The on-board power measurement allow the motherboard to monitor the power consumption in real time and may be used to design adaptive memory scheduling algorithms.

The in- and out-going signals of the memory chips are connected to a central *programmable wiring fabric* which in turn is connected to the external connector. This flexibility allows the designer to realize different topologies within seconds instead of months, needed to re-spin a new board.

All memories can be powered-down and -up at run time which is especially interesting if non-volatile memories are used. Since the signals may float if a memory chip is powered down it is advisable to consider this in the design of the wiring topology.

A limitation of the programmable wiring fabric is that its electric properties do not match those of a custom chip or circuit board wiring. Since the programmable wiring chip consumes very little power it is imaginable that it is used in a real sensor node as well.

If this is undesirable then the results of different designs can either be compared relatively against each other or they must be scaled. Latter requires that real chips are built and measured against the programmable wiring fabric so that the scaling model can be parametrized correctly.

The power consumption of the programmable wiring fabric could also be assumed as constant due to its simple structure.

Compared to the memories the power consumption of the wiring is easier to model. The memory chips often have multiple power states and complex protocols which cause varying power consumption depending on the prior state and last recently issued command. Especially FLASH-memories tend to have very complex logic to handle wear and tear in a transparent way.

Last but not least, the designer may also choose to translate the wiring design into a SPICE-model and simulate it for different technologies.

A discussion of the design space which can be explored with the Theia module and references to examples from the related work can be found on page 76.

The key idea behind Theia is that the different characteristics of the memory chips (e.g. latency, bandwidth and power consumption) are matched with the data access patterns exhibited at run time. For sensor networks a common pattern which could be exploited is *duty cycling* where a sensor node is idle most of the time and wakes up infrequently for short bursts of activities.

### 5.2.3 Power Measurement

Hyperion (*a.k.a.* hardware driven design space exploration platform) is shown on page 93 in Figure 5.3. A distinguishing feature of Hyperion is the extensive support for power measurement. The platform can measure the average power consumption of every attached exploration module as well as major on-board hardware components.

Additionally, the power consumption of the programmable hardware fabric can be measured with clock cycle accuracy. This allows deep insights into the sensor node system-on-chip (SoC) under evaluation.

The following sections introduce the power measurement setup and cover the important aspects of *average-* and *cycle accurate power* measurements in depth. First - however - a few mathematical underpinnings:

### 5.2.4 Deviations and Repeatability of Power Measurements

For measurements it is important to assess their accuracy and ensure repeatability. Two measurement series A and B are assumed *significantly* different if the difference of the absolute averages $\bar{x}_A$ and $\bar{x}_B$ is bigger then the sum of the average absolute deviations $\triangle x_A$ and $\triangle x_B$:

$$|\bar{x}_A - \bar{x}_B| > \triangle x_A + \triangle x_B$$

where

$$\triangle x = \hat{\sigma} = \sqrt{\frac{\sum_{i=1}^{n}(\bar{x} - x_i)^2}{n-1}}, \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5.1}$$

are an estimator for the standard deviation (= average absolute deviation) and the average of the data series respectively. The relative error can be obtained with:

$$r(x) = \frac{\triangle x}{x}$$

The variability of the measurements is an indication of the quality of the measurement setup.

### 5.2.5 The Power Measurement Setup

Figure 5.5 shows the measurement setup which has been developed for the hardware driven design space exploration platform Hyperion. Besides Hyperion it includes external laboratory equipment and a PC.

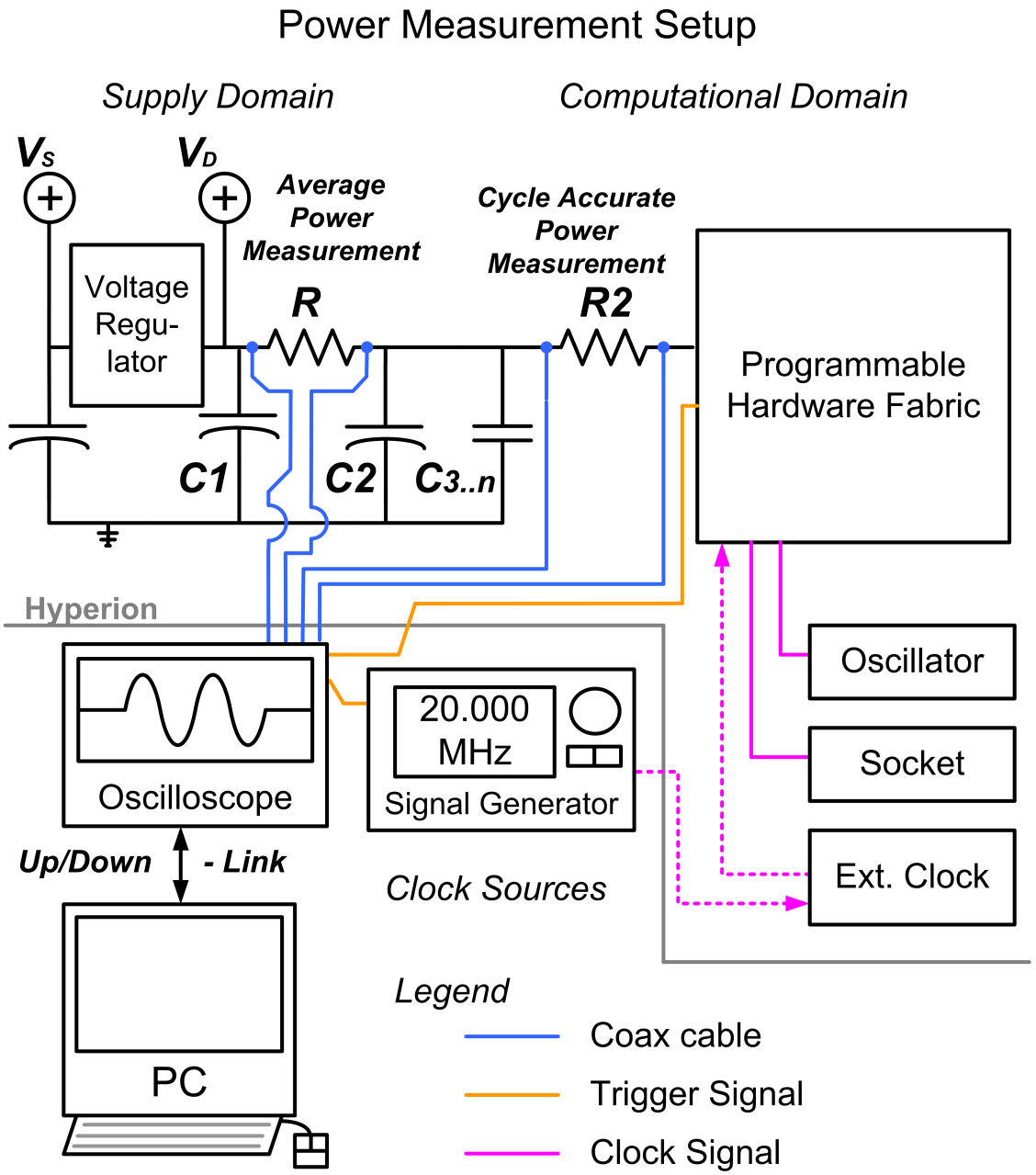



Figure 5.5: The figure shows the power measurement setup of the hardware driven design space exploration platform Hyperion (= board + exploration modules). The gray line indicates the boundary between Hyperion and the laboratory measurement equipment. The average- and cycle accurate power measurement of the computational domain (=programmable hardware fabric) is shown as an example. The external voltage $\mathbf{V_S}$ is regulated down to $\mathbf{V_D}$. The supply current flows through **R** which is the measurement resistor for average power. **R2** is for the cycle accurate power measurement of the programmable fabric. The voltages across the resistors should be measured differentially. For example with an oscilloscope. Custom signals from the hardware fabric or clock signals may be used as triggers. The PC gathers the measured data for evaluation. Besides the signal generator the clock can be sourced from an on-board soldered and socketed oscillator.

The thick gray line indicates the boundary between Hyperion (upper part) and the environment. The diagram shows the average- and cycle-accurate power measurement circuitry for the computational domain. The computational domain is represented by the programmable hardware fabric chip.

The average power measurement circuitry for the exploration *modules* is not shown since it is conceptually similar. There is no cycle accurate power measurement circuitry for exploration modules since the measurement circuits must be placed on the same board as the chip - under evaluation - for fundamental physical reasons.

The source voltage $V_S$ is fed into the linear voltage regulator which outputs a lower but stabilized voltage $V_D$. Preferably, $V_S$ is sourced by a battery and not a laboratory power supply. Even laboratory power supplies often have noise and and spikes in their output power signal which they pick up from the grid. A linear voltage regulator - although very inefficient - has the advantage that the output signal is smooth compared to a more efficient switching regulator. All regulators on Hyperion are linear regulators.

The capacitor C1 stabilizes the voltage by smoothing out supply variations. It is a single capacitor with roughly $100\mu F$.

R is a measurement resistor[6]. The voltage across its terminals is proportional to the average power consumption. Measurement resistors are specialized components which exhibit extremely small variations over a wide range of currents, frequencies and temperatures. The size of the measurement resistor depends on the measurement range and the experienced voltages. There is an inherent trade-off between the accuracy of the voltage measurement and the undesirable voltage drop caused by the measurement resistor. The measurement resistor should be at least an order of magnitude smaller than the resistance of the circuit under evaluation. If an ampere meter is used instead of a less accurate oscilloscope than R can be dimensioned smaller. Currently, Hyperion uses measurement resistors of $100m\Omega$ .

The capacitor C2 is a low-pass filter and smooths out high frequency noise to prevent measurement errors. The decoupling capacitors C3 to Cn (with $n$ easily bigger than 10 or 20) provide energy to the chip when it switches and must therefore be placed nearby. Their size is usually around $100nF$. To reduce their resistive losses it is beneficial to have many of them in parallel. Compared to the $100\mu F$ capacitor they discharge much quicker and stabilize the supply voltage even if the chip's logic switches at high frequencies.

The resistor R2 has been placed directly next to the chip to facilitate cycle accurate power measurements. The supply layers and -traces (= conductor path) have been carefully engineered to reduce (low pass filtering) capacitances to a minimum and still ensure a stable supply voltage. The PCB-layout can be found in the appendix on page 154.

For measuring the voltage across R2 it is strictly necessary to use a very high sampling rate because the signal is not averaged (= low pass filtered) by capacitors. Since the transistors in a chip switch at frequencies of 500 MHz and higher it is necessary to sample fast enough to obey the Nyquist–Shannon sampling theorem (= at least twice the signals bandwidth). The

[6] The Isabellenhuette kindly provided the measurement resistors. `http://www.isabellenhuette.de`

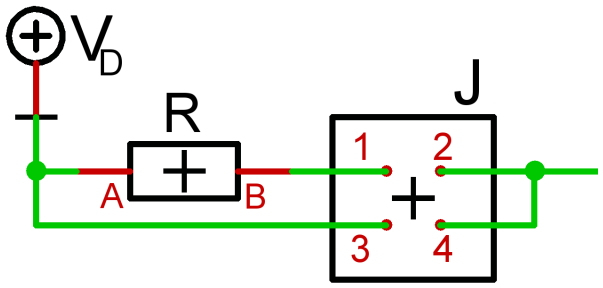



Figure 5.6: The schematic diagram helps to illustrate how average power measurements (see Figure 5.5 on page 99) can be configured on the Hyperion board. It is possible to assess the power consumption either by current or voltage measurement. An ampere meter can be directly connected to **3** and **4.** Alternatively, the voltage drop over a measurement resistor **R** can be measured over the connectors **1** and **3** - if **1** and **2** are connected. A connection of **3** and **4** can be chosen if no measurements are carried out.

oscilloscope in our laboratory is a mixed signal Agilent MSO6054A which can sample at a rate of 4 giga samples per second. The cables leading to the Hyperion board are coaxial cables. They can carry high frequency signals.

The oscilloscope can either be triggered by the rising/falling clock edge or by custom trigger signals generated by the sensor node system-on-chip (SoC) which has been loaded into the programmable hardware fabric. This is a very powerful technique but requires small SoC design changes.

The laboratory PC is connected with the oscilloscope. A custom measurement application configures the oscilloscope, starts the measurement, downloads the acquired data and pre-processes it automatically. This allows many measurements to be carried out efficiently with a high confidence.

To stretch the switching activities of the programmable logic fabric it is possible to generate slower clock signals with a signal generator. It is even possible to halt the circuit completely.

The on-board soldered oscillator or socketed oscillator are additional clock sources which may be used in standalone operation. For example to simplify the measurement setup or when a field test is carried out.

One last remark: if the voltages are measured differentially with an oscilloscope then it is advisable to use either a differential probe. Otherwise special precautions must be taken to prevent ground loops.

### 5.2.6 Average Power Measurement

The hardware driven design space exploration platform Hyperion allows the average power of the programmable hardware fabric and the exploration modules to be measured separately. The

average power is measured *after* the (linear) voltage regulators because the focus of this thesis is on the computational-, communication- and to a lesser degree on the sensing-domain. It is *not* on the power supply domain (see page 92 et seq. and pages 78 ff. for an outline of the power supply design space). The power supply system of the Hyperion board has been optimized to deliver a *clean* and *undistorted* power signal at the cost of voltage conversion efficiency. Since Hyperion is a design space exploration platform for sensor nodes and *not* a deployable sensor node it has been necessary to make different trade-offs.

The average power power can be determined by voltage- or current measurement:

Figure 5.6 shows the details. The current can be measured directly. For voltage based power measurements it is necessary to measure the voltage drop $V_R$ over an on-board resistor $R$.

For the designer it is often interesting to know how much *energy* a certain period $T$ costs. Thus the power $P = UI$ has to be multiplied by $T$ as shown in equation 5.2. The voltage drop caused by the load (for example an exploration module or the programmable fabric) is calculated and multiplied by the current through the measurement resistor $R$.

$$E = UI \cdot T = \overbrace{(V_D - V_R)}^{U} \overbrace{\left(\frac{V_R}{R}\right)}^{I} \cdot T \tag{5.2}$$

All parameters of equation 5.2 are prone to measurement errors. If the inputs are independent and the errors are sufficiently small then it is possible to calculate the *maximum* error of $E$ by:

$$\begin{aligned} \triangle E &= \left|\frac{\partial E}{\partial V_D}\right| \Delta V_D + \left|\frac{\partial E}{\partial V_R}\right| \Delta V_R + \left|\frac{\partial E}{\partial T}\right| \Delta T + \left|\frac{\partial E}{\partial R}\right| \Delta R \overset{V_R>0,\ V_D>0,\ R>0,\ T>0}{=} \\ &= \frac{V_R T}{R} \Delta V_D + \frac{T|V_D - 2V_R|}{R} \Delta V_R + \frac{|V_D V_R - V_R^2|}{R} \Delta T + \frac{T|V_D V_R - V_R^2|}{R^2} \Delta R = \\ &\overset{V_D \gg V_R}{\approx} \frac{T}{R} \cdot V_D V_R \left(\frac{\Delta V_D}{V_D} + \frac{\Delta V_R}{V_R} + \frac{\Delta T}{T} + \frac{\Delta R}{R}\right) \end{aligned} \tag{5.3}$$

where $\Delta V_D$, $\Delta V_R$, $\Delta T$ and $\Delta R$ are the errors of the inputs.

Thus the relative error is

$$\frac{\Delta E}{|E|} \approx \frac{\Delta V_D}{V_D} + \frac{\Delta V_R}{V_R} + \frac{\Delta T}{T} + \frac{\Delta R}{R} \tag{5.4}$$

the sum of the relative errors of the inputs.

Since the supply voltage over the load has to be stable it must be ensured that the measurement resistor $R$ (in series with the load) causes only very small voltage drops in relation to the load. Therefore it must be ensured that $V_D \gg V_R$.

A trade off regarding the size of $V_R$ has to be made. $V_R$ is more difficult to measure the smaller it gets (= less accurate). *But* it influences the load more, the bigger it gets (for CMOS circuits:

$P_{Avg} = \alpha \cdot V_D^2 \cdot C_{LOAD} \cdot f_{CLK}$ where $\alpha$ is the number of power consuming transitions per clock cycle; $f_{CLK}$ depends on $V_D$).

If the load varies a lot between different measurements it is advisable to adapt the size of the resistor $R$[7].

The absolute summands $\left|\frac{\partial E}{\partial x_i}\right|$ in Equation 5.3 are all positive and inhibit a cancellation of measurement errors. Therefore the (linear) error propagation tends to overestimate the measurement uncertainty - unless some errors vary considerably in size compared to others. This may be the case if different measurement instruments are used for $V_R$ or $V_D$ for example.

The Gaussian error propagation considers error cancellation and is suitable if the measurement errors have the same dimension.

The equation for $\Delta E$ - if $V_D$ and $V_R$ are considered - is:

$$\begin{aligned}\Delta E &= \sqrt{\left(\frac{\partial E}{\partial V_D}\Delta V_D\right)^2 + \left(\frac{\partial E}{\partial V_R}\Delta V_R\right)^2} \\ &= \sqrt{\frac{V_R^2 T^2}{R^2}\Delta V_D^2 + \left(\frac{T \cdot V_D}{R} - \frac{2V_R \cdot T}{R}\right)^2 \Delta V_R^2} \\ &\overset{V_D \gg V_R, R>0, T>0}{\approx} \frac{T}{R}\sqrt{\left(V_R \Delta V_D\right)^2 + \left(V_D \Delta V_R\right)^2}\end{aligned} \tag{5.5}$$

and with Equation 5.2, we get

$$\frac{\Delta E}{E} \overset{V_D \gg V_R}{\approx} \sqrt{\left(\frac{\Delta V_D}{V_D}\right)^2 + \left(\frac{\Delta V_R}{V_R}\right)^2}$$

Thus the relative error $\frac{\Delta E}{E}$ is the square root of the sum of the squares.

### 5.2.7 Cycle Accurate Power Measurement

The power consumption of the programmable hardware fabric (on the Hyperion board) can be measured with cycle accuracy. Figure 5.7 shows conceptually the expected outcome. The purpose of the cycle accurate power measurement has been discussed on page 92. The main limitation of the programmable hardware fabric is that its structure is different from a real chip and thus the cycle accurate power measurements do not (directly) represent a real chip's performance.

However, if a pin-compatible ASIC-chip (based on a previously programmed sensor node SoC) is manufactured and placed on the Hyperion board then the results are directly usable and

[7] Plug-able measurement resistors would be desirable.

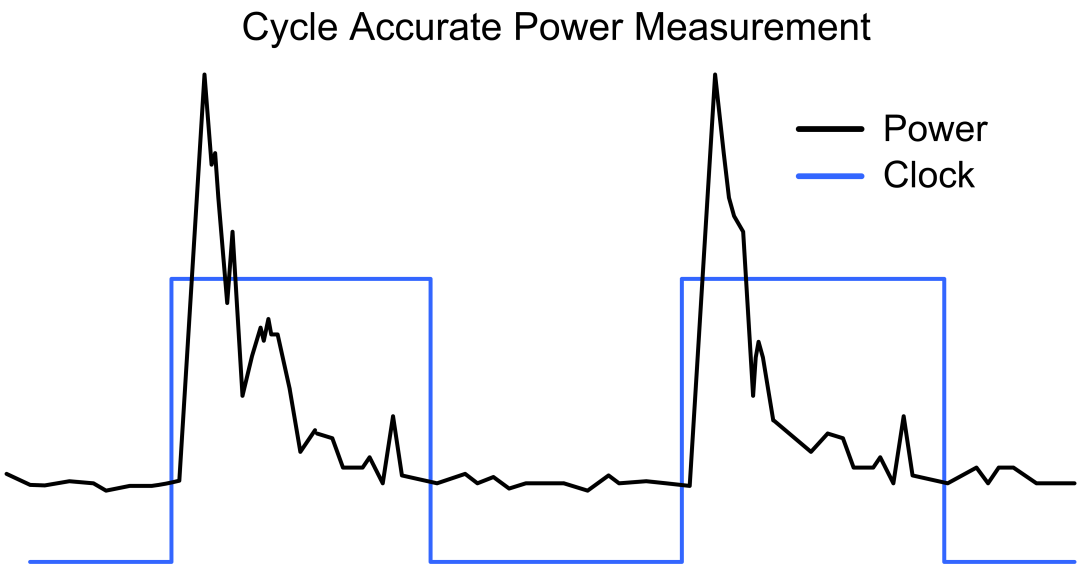



Figure 5.7: The figure shows the conceptual wave form which can be expected from cycle accurate power measurements. When the clock edge rises many gates start to switch which causes a sharp spike. Shortly afterwards the switching activity drops sharply and flattens towards the end of the clock period. If the clock speed is reduced then the flat line is extended. On the other hand if the clock speed is increased continuously then the activities will start to overlap at some point and the circuit will cease to function properly. The steepness of the spikes depends on the switching speed of the transistors.

need not to be scaled. Due to time and cost constrains this has not been done. Nevertheless, experimental measurements based on the programmable hardware fabric are sufficient to prove the performance of the measurement setup itself. This has been a major reason to include circuitry for cycle accurate power measurements on the board.

The various pros and cons of resistor- and capacitor based cycle accurate power measurements can be found in the related work on page 103 et seq. For Hyperion a resistor based power measurement has been chosen since it is more robust and allows the complete wave-form of a cycle to be captured. The capacitor based methods can "only" calculate the consumed energy per cycle.

To acquire the wave form a digital oscilloscope with a sampling rate of 4 GS/s and 8 bit A/D resolution is sufficient. At this data rate only a short period of time can be captured. Therefore, it is important to trigger the measurement precisely. This can either be done on the rising- or falling edge, or with custom trigger signals generated by the sensor node system-on-chip. Latter is much more flexible and the preferred method.

For technical reason it is necessary that the trigger fires shortly before the region of interest to allow the oscilloscope to settle in and prepare the acquisition. Otherwise, the acquired wave form may suffer from artificial distortions.

Obviously, the additional trigger signals require modifications to the original design. The signal additions should not impact the power measurements significantly because the triggers do not fire often. Usually only once per acquisition. The required logic should also be relatively small unless an excessive number of trigger signals is generated or the measured hardware design is trivial.

The voltage across the measurement resistor R2 (Figure 5.5) is measured differentially. The shielded coaxial cable is directly soldered onto the board to avoid signal reflections and filtering effects caused by connectors.

The clock signal for the logic is sourced from an external clock generator over coaxial cable to avoid the chip internal clock generation. Latter causes too much noise on the power rails and must therefore be avoided.

Unlike R, R2 is physically close to the programmable hardware fabric chip and there are no (decoupling) capacitors in between the two. Even the layer capacitances have been removed to avoid low-pass filtering. To minimize supply voltage fluctuations and ensure accurate measurement it had been necessary to invest a lot of engineering effort into the design of the Hyperion board. The interested reader can find the PCB-layout and schematic diagrams in the appendix on page 145.

## 5.3 Part II : Software driven Design Space Exploration Platform

The software driven design space exploration platform *Sensim* realizes the full-system simulation and exploration of wireless sensor networks. It consists of two major components: the full-system simulator and the energy estimator.

The first section locates the two components within the design flow. Like in the hardware driven design space exploration platform a major goal has been the assessment of the power consumption of a wireless sensor network.

The design flow is extensible. Therefore, it is possible to integrate 3rd-party products.

After the design flow the (full-system) simulator architecture is introduced. The simulator can also be broken into two components: the configurable sensor node system-on-chip (SoC) - simulator and the environment simulator.

The SoC-simulator simulates all aspects of a sensor node (computation-, sensing- and power-supply-domain - for the design space domains, please see pages 73 ff) whereas the environment simulator e.g. models the radio communication in accordance to (exchangeable) error models (= communication domain).

In the last section the capabilities of energy estimator are laid out. The energy estimator analyses the simulation traces generated by the full-system simulator. A novel feature of the energy estimator is the energy break-down into user-defined categories, like routing or compression for example. Categories span both: hard- and software components of a sensor node platform.

### 5.3.1 A Flexible and Extensible Exploration Design Flow

The software driven design space exploration platform Sensim has been designed to obtain insights into the power consumption of wireless sensor networks. Since power consumption is the crucial metric it is important to understand where the design has potential for further optimizations.

A single analysis tool cannot provide all imaginable information or outdo every previously existing tool. Therefore, the design flow and Sensim itself is modular and extensible.

Certain aspects of performance evaluation are inefficient or challenging to model accurately in simulation and are better explored on the hardware driven design space exploration platform Hyperion. For everything else the software driven design space exploration platform provides a flexible exploration environment.

Figure 5.8 visualizes the design flow. The two Sensim components for full-system simulation and energy estimation are shown in the lower right part. Together they form Sensim. On the

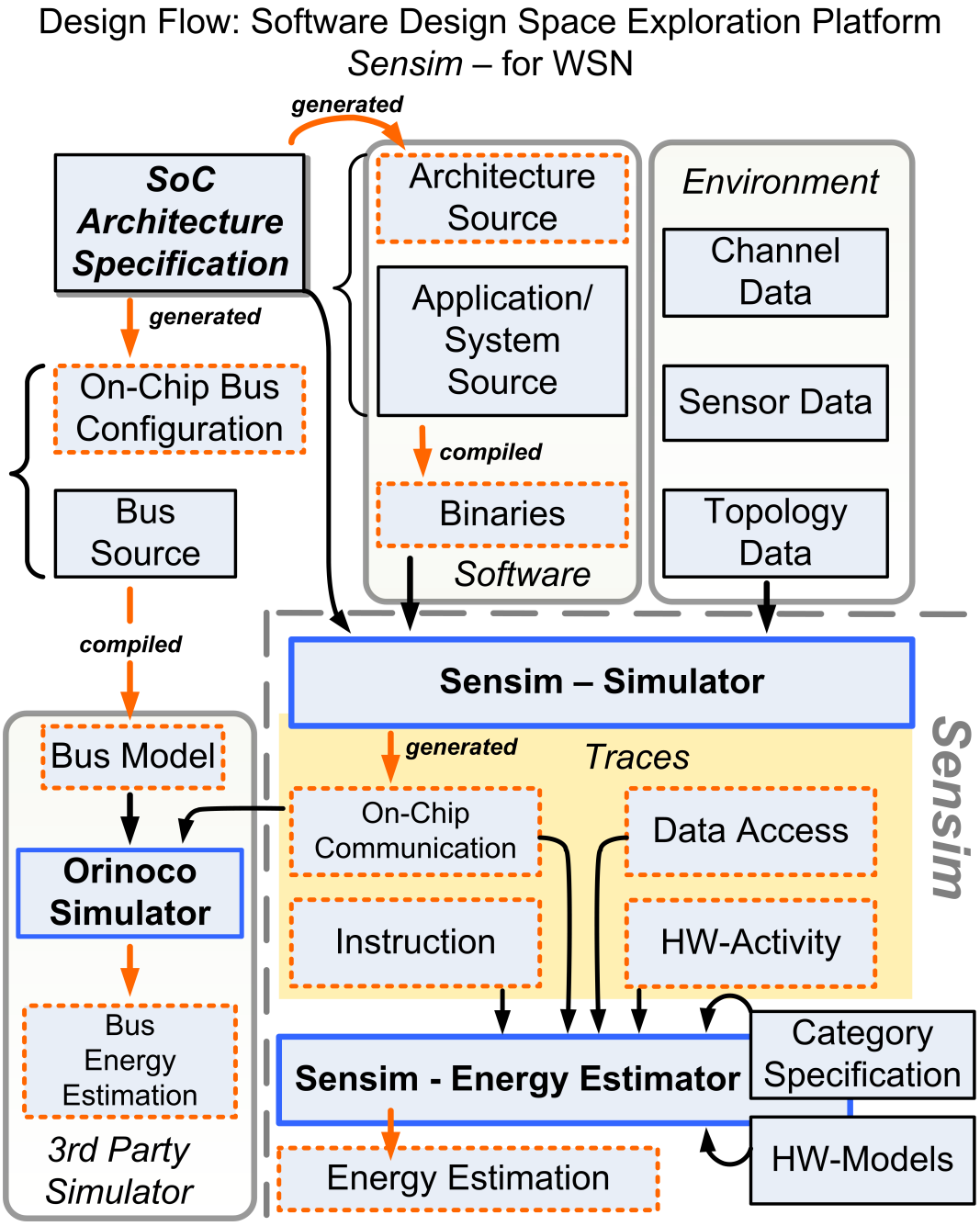



Figure 5.8: The figure shows the design flow of the software driven design space exploration platform Sensim. Sensim consists of two components: the simulator and energy estimator. 3d-party simulation/estimation platforms can also be integrated (e.g. Orinoco for on-chip communication). The sensor node SoC-architecture specification is necessary to generate architecture dependent source and configuration files for simulation and system construction. The environment data inputs for the simulator may be synthetic or derived from the hardware driven platform by measurement e.g. During simulation a number of selectable traces are generated and can be fed into the energy estimator to obtain a power break-down by categories. The hardware power-models can be re-parametrized and the estimator re-run to carry out what-if-analyses without restarting the (time consuming) simulation.

lower left a 3rd-party simulation and estimation tool (called Orinoco[8]) is shown. Orinoco is a good example of how an existing exploration tool can be integrated into the design flow.

The starting point for the hardware simulation is a system-on-chip (SoC) architecture specification. The network (topology) and environmental settings (channel- and senor data e.g.) are direct simulator inputs. Sensor data can either be synthetic or pre-recorded. The sensor data may have been derived from experiments with the hardware driven design space exploration platform which uses real hardware components.

The SoC-architecture specification is required to generate architecture specific source- and configuration files. It contains information about the number and kind of processing elements, memories, radio transceivers and peripherals but also how they are interconnected and parametrized. For connectivity buses, bridges and/or switches e.g. may be chosen.

A custom application- and system binary is compiled for the specified architecture. The obtained binary will be identical on the software- and hardware driven design space exploration platform as well as on the real sensor node.

The full-system simulator uses the SoC-architecture specification to instantiate and parametrize the hardware simulation models.

The SoC-architecture specification is also necessary for the construction of cycle accurate hardware models (on-chip buses e.g.) which may be used with the hardware driven design space exploration platform or 3rd-party simulators like the aforementioned Orinoco.

Once the application- and system-software has been constructed and information about the environment is available the simulation of the wireless sensor network can be initiated.

After full-system simulation so called simulation traces are available for forthcoming analyses. Examples of commonly generated traces are: memory traces (instruction stream and data access), hardware activity traces which logs hardware state transitions and on-chip communication traces. Latter may be fed into Orinoco to analyze on-chip communication power consumption for example. The other traces may be fed into Sensim's energy estimator.

The energy estimator has power models for various hardware components and determines the energy taken by each user-defined category (e.g. routing, compression, signal processing).

The break down into categories provides a *coherent* view on the system power consumption (see page 111 ff).

### 5.3.2 Software Architecture of the Platform

The main design objective of the Sensim full-system simulator has been to provide a *flexible* and *extensible* platform for the software driven simulation of a wireless sensor network. The key architectural decisions based on these requirements are the separation of simulation and

[8] `http://www.chipvision.com/`

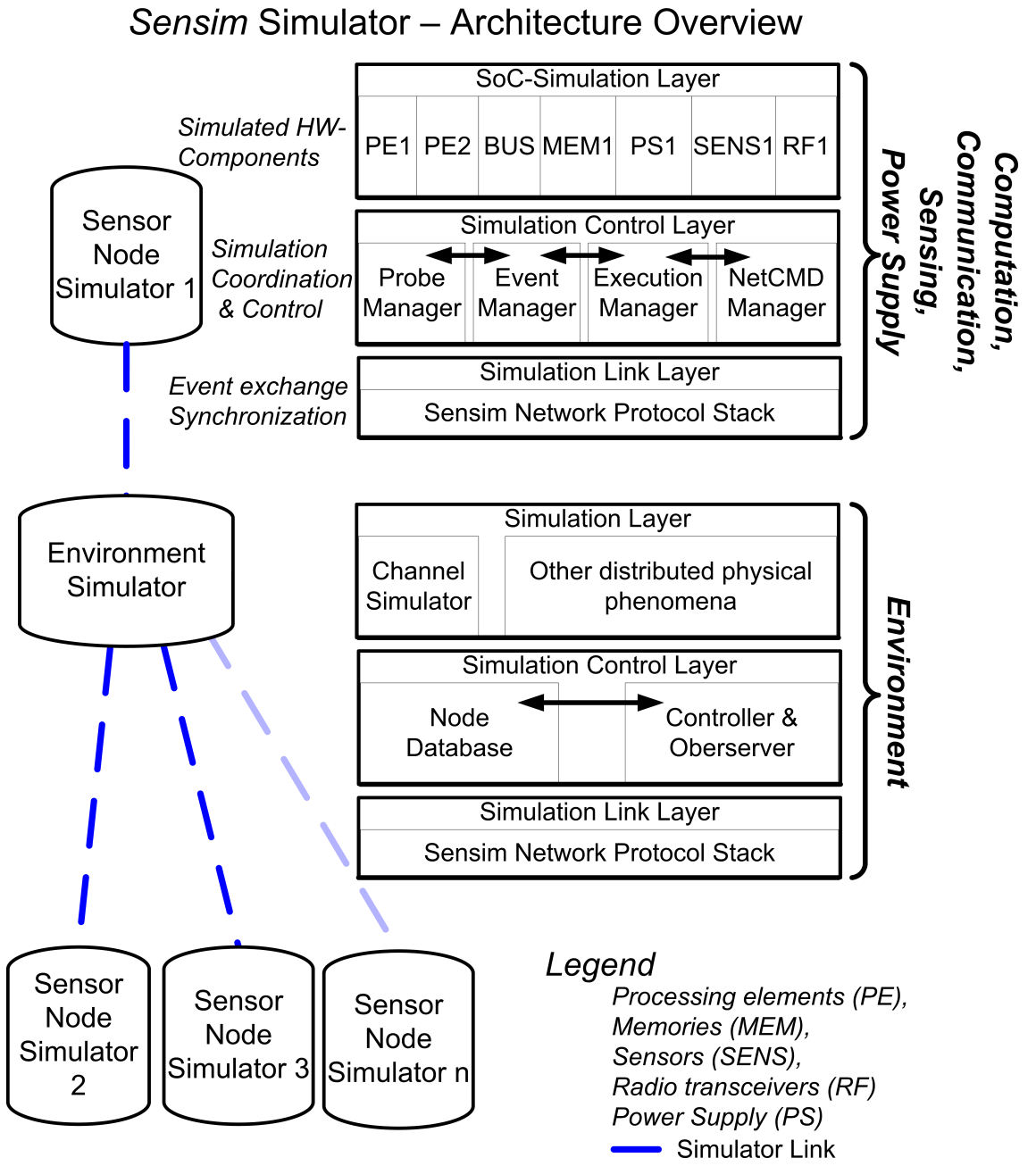



Figure 5.9: The figure shows the Sensim full-system simulation architecture. The configurable sensor node simulator instantiates simulated hardware models according to a SoC-specification. The environment simulator is connected to all sensor node simulators. The environment simulator models all shared and distributed physical phenomena (for example the radio channel). Both simulation components: the node- and environment simulator have a layered architecture. The link layer connects the sensor node simulators to the environment simulator which synchronizes and coordinates the network's simulation from within the simulation control layer. The corresponding layer in the node simulator has managing components for probing and managing (local) events and (remote) execution control. The top layer holds the simulation components for hardware- components and environmental phenomena.

(power)-analyses by using trace files - as well as - the division between node- and environment simulation.

The architecture of the Sensim simulator is shown in Figure 5.9. For each sensor node a sensor node simulator is instantiated and connected with the environment simulator.

The sensor node simulator assembles a simulation model according to the architecture SoC-specification. The simulated hardware components could be processing elements (GPP, DSP), memories (SRAM, DRAM, Cache), radio transceivers, power supplies (battery) and sensors. Thus all elements of the design domains outlined in Chapter 4.

The environment simulator simulates the wireless channel and may be extended to simulate other distributed (*shared*) physical phenomena. The simulation component for wireless communication has a simple range and collision model. The challenging design of accurate wireless communication simulation (pages 37 et seq.) has not been the focus of this thesis. The radio simulation component may be swapped for a better (and probably slower) simulation component if necessary. The proposed method is to use the hardware driven design space exploration platform whenever realistic radio communication must be part of the assessment.

The node simulator uses functional models of the simulated hardware similar to the Avrora/ AEON simulator which has been introduced in the related work. The hardware peripheral interfaces are modeled bit accurately and behave according to the state machine specifications in the data sheets. Probes may be inserted to log state changes and events in hardware models. For common tasks predefined probes are available.

For sensors it is possible to read-in prerecorded data if the sensed phenomena need not be modeled globally. The virtual image sensor - for example - may load images previously acquired by a real image sensor. The image data could thus come from a camera exploration module which had been plugged into the hardware driven design space exploration platform.

The radio transceiver model conveys transmitted data to the node simulator which forwards it to the environment simulator where the transmission propagation is subject to range- and error constrains imposed by the configured wireless channel. The node database provides the necessary information (e.g. node locations). The radio transmissions may also be observed at simulation time and logged.

The sensor node- and environment simulator has a 3-layer architecture:

The bottom layer (“Simulation Link Layer”) realizes the exchange of simulator event- and synchronization messages. Each simulator runs independently on either a dedicated or shared simulation computer. Therefore it is possible to scale the simulation to multiple machines.

The simulation control layer as its name suggests controls the execution of the simulation:

Within the environment simulator the “node database” keeps track of the participating sensor node simulators. The entire simulation is coordinated by the “Controller & Observer ”-component.

A novel feature of Sensim which is based on this control component is the concept of network wide (event-triggered) breakpoints. They may halt the simulation under certain conditions. These conditions may involve events on the network- or node-level. Once halted the node simulators can be single stepped and their states may be inspected until simulation is resumed. For repeated experiments it is possible to script the simulation. Usually, sensor networks are inherently difficult to debug due to their distributed nature and fault modes.

In the node simulator the simulation control layer consists of four managing components. The probe manager keeps track of the probes which monitor certain events. The event manager queues up and delivers local- and remote-events. The execution manager controls the simulation of the local hardware components and may be remotely controlled by the participating -manager. Latter receives commands from the environment simulator controller which coordinates the entire simulation.

The top layer ("simulation layer") contains the hardware- and environment simulation components.

For Sensim, the Avrora/AEON-simulator (pages 43 et seq.) has been a source of inspiration. Avrora/AEON has been conceived as a scalable *Mica* sensor node simulator (see page 12) at the University of California, Berkeley.

Sensim borrows the "probes" concept and some low-level utility routines from Avrora / AEON. Probes may be attached to the simulator core or simulated hardware components and are activated when certain events occur.

Some of the analysis components - of this thesis - where first developed for Avrora and then later integrated into Sensim to facilitate the exploration of (configurable) sensor node platforms.

### 5.3.3 Power Estimation

The Sensim full-system simulator records traces which are fed into the Sensim energy estimator (after simulation) to obtain a detailed energy break-down (see pages 106 et seq. for the design flow). Thus the full-system simulation and energy-analysis are decoupled.

The advantage is that the simulator output (traces) can be analyzed repeatedly with differently parametrized hardware power models. *What-if* -style analyses can then be carried out without re-running the simulation. Additionally, selected traces can be fed into future analysis tools to compute custom performance metrics. An example is given on page 125 et seq. A disadvantage of traces are their storage overhead.

The power within a sensor node is consumed by its hardware components which are ultimately controlled by software. An energy break-down into hardware components would not reveal the energy costs of high-level activities which utilize shared hardware components over extended periods of time.

Therefore, Sensim lets the designer define categories which represent high level activities. Typical categories in a sensor network are for example: sensing, signal filtering, compression, net-

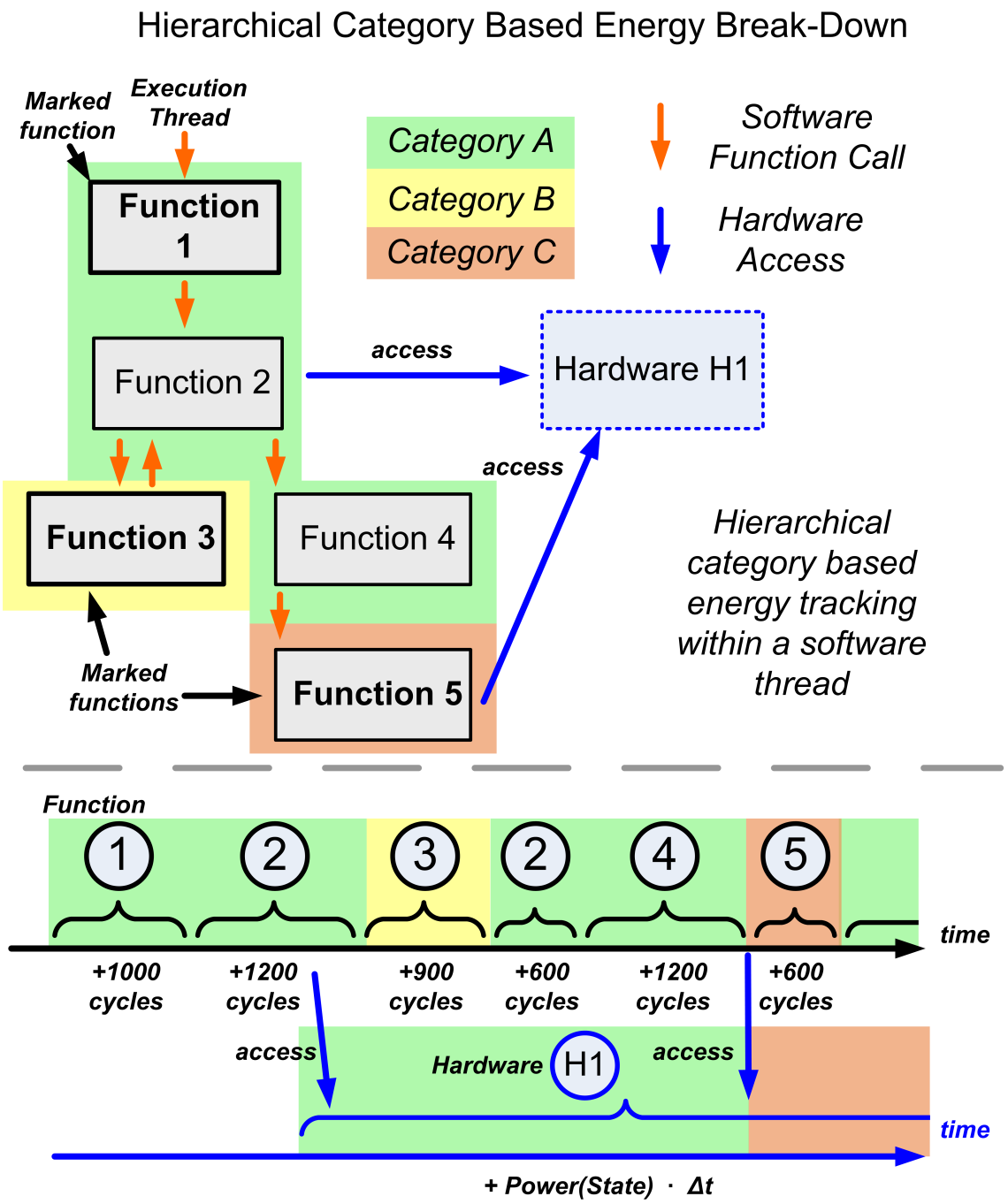



Figure 5.10: The figure illustrates the hierarchical category based energy accounting in Sensim. Function 1 is marked as category A and invokes function 2 which has no predefined category. Therefore it assumes the currently active category A. Function 2 calls function 3 which is marked as category B. When function 3 returns to function 2 the active category is restored to A, although function 2 is not marked. The hardware access by function 2 - prior to calling function 3 - causes hardware H1's energy to be accounted to category A until function 5 accesses H1. Besides the user defined categories A,B and C there is also a special category ("No Category") in cases where no category is active. For sensor nodes useful categories may be: routing, signal filtering/compression or sensing.

work stack, MAC-layer and routing. The Sensim energy estimator uses these or other designer defined categories in its energy break-down statistics.

The Quanto energy profiler (see pages 49 ff) uses a similar methodology but does not support *hierarchical* energy tracking and has a different (tracking) scope.

The hierarchical category based energy break-down in Sensim tracks the energy usage along the *function*-call chain of a software task. Sensim allows the designer to *mark* software functions (= series of processor instructions). When such a marked function is invoked its category *becomes active* until the function is finished. In this case it *restores* the previous category. If a marked function calls other functions then there are two possibilities:

Either the other function is also marked - in which case its category becomes active - or the called function is not marked. Then the category of the calling function is assumed.

If a thread accesses hardware components (besides the processor(s) it runs on) then the energy of these hardware components is accounted to the active category as well. This is done until another category becomes active and accesses the same hardware component(s). Or, if the hardware component is deactivated. For example due to a timeout.

The category "No Category" is used as a default when no other category is active. For example directly after a new task has been spawned.

Figure 5.10 illustrates the category tracking along the call chain with an example.

The power models of the Sensim energy estimator are comparable with those in Avrora/AEON (see page 43 ff). They usually assume a finite number of hardware states which exhibit a constant power consumption. The parameters must be obtained by measurement *or* are subject to design space exploration experiments. Due to the flexible nature of Sensim it is possible to extend or change the power models without modifying the simulator.

## 5.4 Summary

This chapter has introduced the flexible design space exploration platform for wireless sensor networks. The platform consists of two sub platforms: the hardware- and software driven design space exploration platform. They are called Hyperion and Sensim respectively.

Both platforms have focus the assessment of power consumption which is the crucial performance metric of wireless sensor networks. They can explore the computational-, communication- and sensing domain as defined in Chapter 4. The focus in this thesis is on the computation- and communication domain.

The two *complementing* platforms are integrated into the design flow. The design flow has two evaluation paths for design space exploration: hard- or software driven simulation.

The hardware design space exploration platform Hyperion provides the hardware driven simulation. Hyperion uses a programmable hardware fabric to simulate the sensor node system-on-chip (SoC) cycle accurately and in real time. Hyperion can interface with real hardware

components on exploration modules (= extensible). The Sensim full-system simulator uses virtual hardware components instead.

The design of Hyperion has been optimized for undistorted power measurements. The platform can measure the *average* power consumption of each exploration module. The power consumption of the programmable logic can be measured with cycle accuracy. The applicability and limitations of the cycle accurate power measurement have been discussed in depth. The power measurement setup for the average- and cycle accurate power measurements has been presented. As well as the mathematical theory necessary for the evaluation of the measurements.

The Theia memory exploration module for advanced sensor nodes is an example for Hyperion's extensibility in respect to design space exploration. Theia provides access to multiple novel and upcoming memory types. It allows the designer to match the memory characteristics to the application requirements. To keep the high standard of flexibility it has been equipped with a programmable wiring fabric.

The flexible and extensible software driven design space exploration platform Sensim complements the hardware driven platform. Sensim can estimate the energy consumed by a wireless sensor network. It is a platform which supports repeatable experiments with many sensor nodes. The designer may define arbitrary environmental conditions in simulation. Data which can be acquired in simulation may be hard to acquire in a real system in some cases.

The Sensim platform has two major components: the Sensim full-system simulator and the energy estimator.

Due to the flexible design flow it is possible to integrate 3rd-party analysis and estimation platforms - if necessary.

The environmental inputs and parameters for the sensor network simulation may be obtained through the hardware driven platform. Once the system has been constructed according to its sensor node SoC-architecture specification and when all inputs are compiled, then it is possible for the simulation to commence.

The Sensim full-system simulator consists of two parts: a sensor node- and environment simulator. Latter simulates shared distributed physical phenomena like radio wave propagation for example. The environment simulator coordinates the simulation on a network level and offers rich introspection capabilities.

Through simulation various traces can be obtained and used for subsequent analyses. The traces contain - for example - information about hardware state changes and memory accesses (instruction and data).

The Sensim energy estimator is one consumer of these traces. The estimator has power models for each hardware component in a sensor node. Hence it can estimate the consumed energy.

The energy break-down provided by the Sensim energy estimator is *categorized* to account the energy to specific (high-level) activities. The designer defines the different categories (e.g. sens-

ing, routing, network stack). The *hierarchical* activity tracking follows the execution of software tasks along their call chains. The tracking includes accesses to hardware components.

For validation purposes it is possible to use data which has been acquired with the hardware driven design space exploration platform. This is a good example for a case where the platforms complement each other.

Now, that the design space exploration platform (Hyperion plus Sensim) has been dissected and discussed in great depth, the reader ought to be well prepared for the case studies in following chapter.

# 6 Case Studies

This chapter introduces several case studies that demonstrate the capabilities of the design space exploration platform and the design flows conceived in this thesis.

First Plasma, an example for a sensor node system-on-chip (SoC) is introduced. The SoC has been designed for a camera tracking sensor node since camera tracking had been chosen as the central application theme. Subsequently, the SoC is used to demonstrate the cycle-accurate power measurement capabilities of the hardware driven platform.

In the second case study an exemplary, custom metric is integrated into design space exploration platform. The analyzer that computes the metric uses the traces generated by software driven design space exploration platform. This case study is a good example for the extensibility of the design flow and the software driven design space exploration platform. The custom metric is applied to two mathematical kernels, in order to demonstrate the usefulness of custom metrics for sub-design space explorations.

## 6.1 Hardware driven Platform: A Sample Sensor Node SoC

In this case study a sample wireless sensor node system-on-chip (SoC) is presented. The cycle accurate design targets the hardware driven design space exploration platform Hyperion. The complementing software driven design space exploration platform Sensim has a functional model of the same SoC but is not discussed in this case study.

More specifically, this case study introduces a sensor node SoC for a wireless camera tracking sensor node. This application has been chosen in accordance with the central application theme of this thesis which had been introduced in the "System Vision" on page 65 et seq. Based on this sensor node SoC design the reader may explore alternatives or extend the design.

The SoC design is used to demonstrate the cycle accurate power measurement capabilities of Hyperion. The designer will be shown what can be expected from power measurements on a very fine grained level. The discussion of the cycle accurate power measurements - in this case study applies to all pipelined circuits.

The power measurement methods and the setup have been described in detail on pages 98 et seq.

Commercially available sensor node prototypes like the Telos, Mica or SunSPOT (see related work on pages 12 et seq.) have fixed hardware circuits and are therefore restricted to software

programming. The hardware driven design space exploration platform Hyperion (see pages 92 et seq.) has a programmable hardware fabric and can be re-programmed. Thus many different sensor node SoC designs can be assessed conveniently without re-spinning a new silicon chip or board.

Unlike software simulation the programmable hardware fabric offers real time hardware simulation and access to physical devices like sensors or radio transceivers.

The following section introduces the camera tracking sensor node SoC.

### 6.1.1 SoC Architecture of a Camera Sensor Node

A sensor node consists of computational- (memory and processors), communication- (radio transceiver e.g.), sensing- and power-supply components - see pages 73 et seq. for an overview. Except for the power-supply components the design space exploration platform Hyperion can be configured freely.

The main focus - in this thesis - is on the computational- and communication domain.

For computation a 32-bit RISC core is a sensible base processing element in an advanced sensor node. A 32-bit platform has the advantage that it can be massively scaled up in terms of processing power and memory. Most existing software components have also been written for 32-bit.

An improved design might add special instructions or accelerator cores for image processing. In a camera tracking sensor node a significant amount of image data must be transferred. Therefore, it is advisable to relieve the processor core from data transfers when possible.

Regarding communication, it is possible to interface with arbitrary transceivers on exploration modules. For the camera tracking sensor node a flexible and programmable low-power transceiver has been chosen. The radio transceiver exploration module is shown in the appendix on page 146.

The rich design space of radio transceivers has been outlined on page 73 and dissected into different layers on pages 79 et seq.

As mentioned before, the central application theme is a camera tracking sensor network. Therefore an exploration module with image sensor has been included (appendix on page 146).

Image sensors have a relatively high bandwidth compared to more “traditional” sensors which measure temperature or humidity. Thus the sensor node must have an adequate memory hierarchy and sufficient computational power, so that big chunks of image data can be processed efficiently.

The design space of various sensors and especially image sensors has been illuminated on pages 77 et seq. Important parameters of an image sensor are resolution, color depth and frame rate for example.

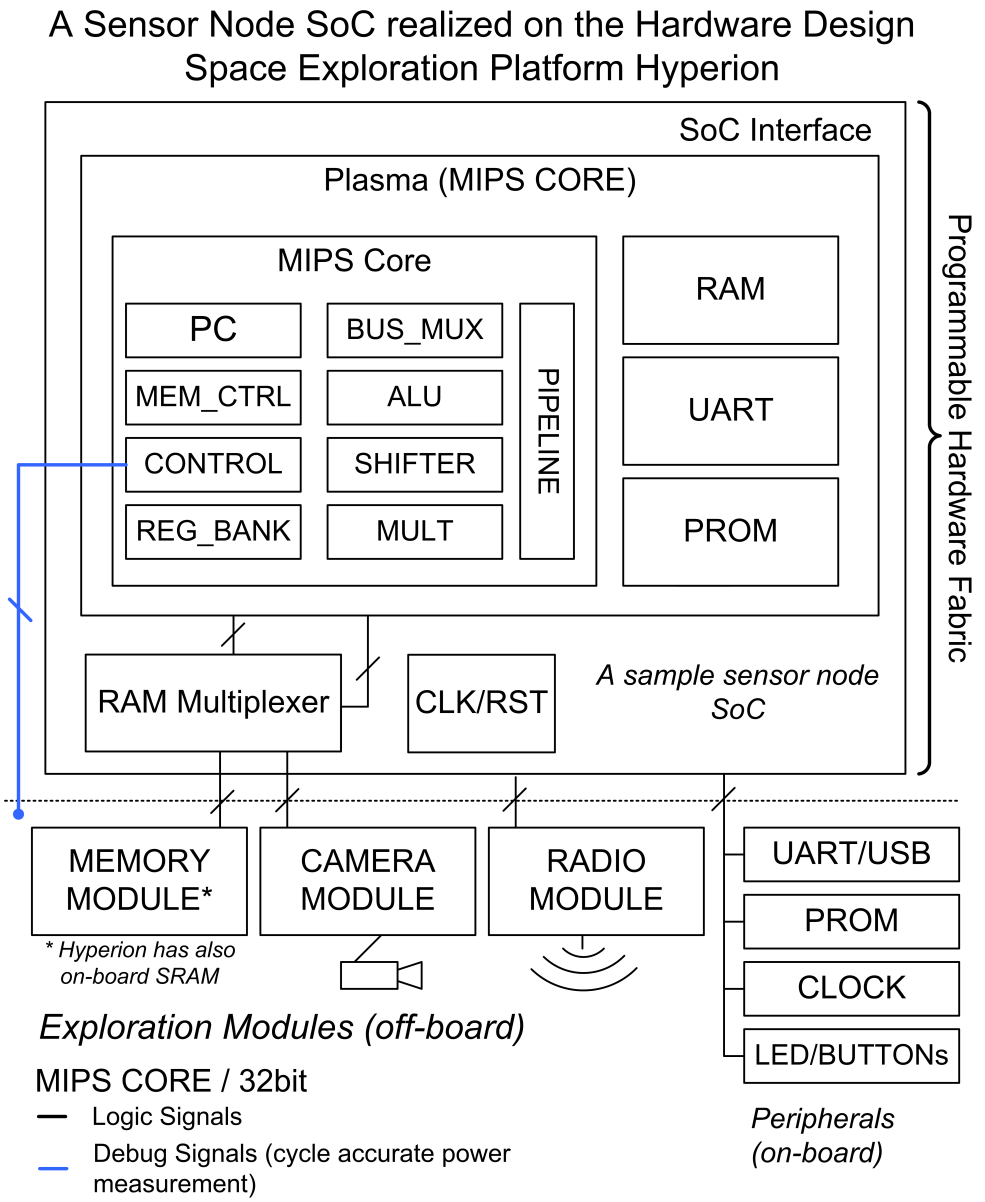



Figure 6.1: The sensor node system-on-chip (SoC) is a parametrizable MIPS 32-bit core (`www.opencores.org`) with custom extensions. The design is loaded from the PROM into the programmable hardware fabric of the Hyperion platform. The core has access to internal and external SRAM memory. The RAM, UART, PROM and CLOCK hardware logic interface with external physical chips. The RAM-multiplexer allows the external camera module to stream image data directly into the external SRAM where it can later be accessed by the MIPS core. The memory- (pages 95 ff), camera-, and the radio exploration modules belong to the computational-, sensing-, and communication domain respectively. Besides the regular logic signals, the SoC has debug signals to aid cycle accurate power measurements (also see pages 98 ff).

A comparison table which can be found on page 167 compares the Hyperion based SoC with the Telos and SunSPOT.

Figure 6.1 shows the hardware architecture of the camera tracking sensor node which has been realized with Hyperion. The sensor node system-on-chip (SoC) is loaded into the programmable hardware fabric.

A customized 32-bit MIPS processor based on the Plasma[1] core from `OpenCores.org` is the central processing element. The micro-architecture customizations include modified clock signal generation, PROM boot support, an enlarged memory space, trigger signals, RAM access multiplexing and an extended peripheral memory map. The core has been configured to have four pipeline stages. An earlier version of Plasma has a wishbone SoC-interface which is not shown in the figure.

All peripherals are accessible through a configurable memory map. Some devices like the radio transceiver can also generate interrupts. The camera sensor node has a small amount of internal SRAM memory (16 kb configured - inside the programmable hardware-logic) and a large external low-power SRAM (3906 kb = 32 MBit).

The RAM multiplexer allows the camera module to stream digital image data directly into the SRAM where the processor can access it. This customization allows the sensor node to acquire images much faster.

The clock signal may be sourced from a configurable on-chip clock manager, a (plugable) oscillator module or an external laboratory signal generator. Measurements can be triggered on the rising/falling-clock edge or by custom debug signals (see also pages 98 et seq. for an overview of the measurement setup).

Besides the exploration modules Hyperion has a small set of on-board peripherals: the UART-USB transceiver, the PROM chip, the plugable oscillator, push buttons and LEDs. The UART-USB interface allows applications to be uploaded and data to be exchanged between the sensor node and a controlling PC. The LEDs are useful for status indication and the push buttons are the only user-interface in field tests if the UART-USB cable is not attached to a notebook for example.

The software stack which runs on Plasma is called Senos. Senos has a small kernel with dynamic task- and memory management. Furthermore, there is a small C- and floating point math-library available. The tool chain is centered around the ELDK-GNU-C compiler.

In the following subsection the camera node SoC is the subject of cycle accurate power measurements.

[1] The most recent Plasma core and documentation can be found here `http://opencores.org/project,plasma`

### 6.1.2 Cycle-Accurate Power Measurement of the Sensor Node SoC

Cycle accurate power measurements are one special feature of the hardware driven design space exploration platform Hyperion. Cycle accurate power measurements allow deep insights into the micro-architectural power consumption. This case study demonstrates what can be expected from cycle accurate power measurements and how they can be conducted.

The different methods of cycle accurate power measurements have been introduced in the related work on the pages 103 et seq. Hyperion uses a resistor based approach. Between the power supply of the *programmable hardware fabric* and its (power supplying) decoupling capacitors a measurement resistor is placed. This is a very delicate spot to place a measurement resistor. Therefore a lot of engineering effort went into the electrical design to ensure a functional system and a high resolution.

The overall measurement setup is shown on the pages 98 et seq. The cycle accurate part can be found on the pages 103 et seq. where also the conceptual limitations of measuring the power consumption of programmable logic are discussed.

Basically, the measurement results cannot be directly mapped to an ASIC design. However, if an ASIC has been built for validation purposes e.g. then it is possible to plug it into the Hyperion board and measure it directly[2].

Figure 6.2 shows a sequence of 11 instructions executed on the MIPS core inside the programmable hardware fabric. The scope screenshot shows the voltage measured across the measurement resistor. The peaks are caused by the 3rd stage of the MIPS core where the opcode is converted into a 60 bit VLIW instruction which encompasses 92 control- and data-signals. In addition to these come 32 signals between the register bank and the bus multiplexer, 64 signals between the bus multiplexer and the pipeline logic as well as a number of other signals.

The programmable hardware fabric is a *XC3S400* Spartan-3 FPGA (Field Programmable Gate Array) from Xilinx. For FPGAs it is not uncommon that the majority of power is consumed within the wiring fabric [120].

The relationship between signals that toggle and power consumption can also be seen in the "Immediate Operands" Figure 6.3. The more bits of an immediate ( = fixed value encoded into an instruction) flip the higher the power consumption becomes. For this measurement the pipeline of the MIPS core was filled with the same instruction. The zero immediate poses as a reference for the other immediates which were varied for each plot. The register was set to R0 which is hardwired to zero. Thus the test application has more control about which bits get toggled in a cycle by adjusting the immediate values and keeping the opcode constant ( = encodes the instruction type).

Table 6.1 shows the energy cost of the instruction ADDI ( = add intermediate ) with a filled pipeline. It can be seen that the energy cost can vary by more than 20% depending on how many bits change in the immediate value.

[2] Granted - of course - that the pin-layout and package is the same.

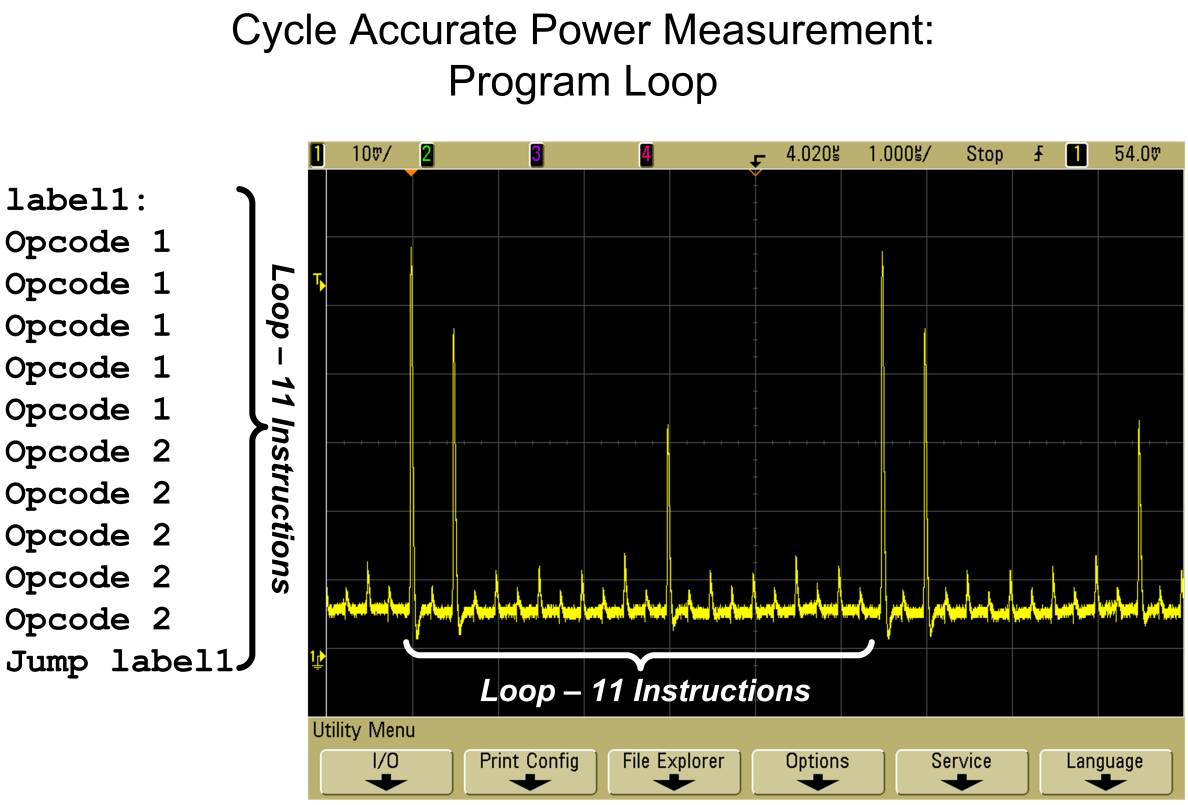



Figure 6.2: The test program has a loop with 11 instructions. The screenshot of the oscilloscope shows 11 peaks which repeat: the sequence has two big peaks followed by four small ones, then one big peak and another four small ones.

| Instruction | Energy |
|---|---|
| ADDI $ro $ro oxoooo | 6.50 nJ |
| ADDI $ro $ro oxoooF | 6.69 nJ |
| ADDI $ro $ro oxooFF | 7.13 nJ |
| ADDI $ro $ro oxoFFF | 7.76 nJ |
| ADDI $ro $ro oxFFFF | 8.36 nJ |

Table 6.1: Per instruction energy cost at 2 MHz for a filled pipeline

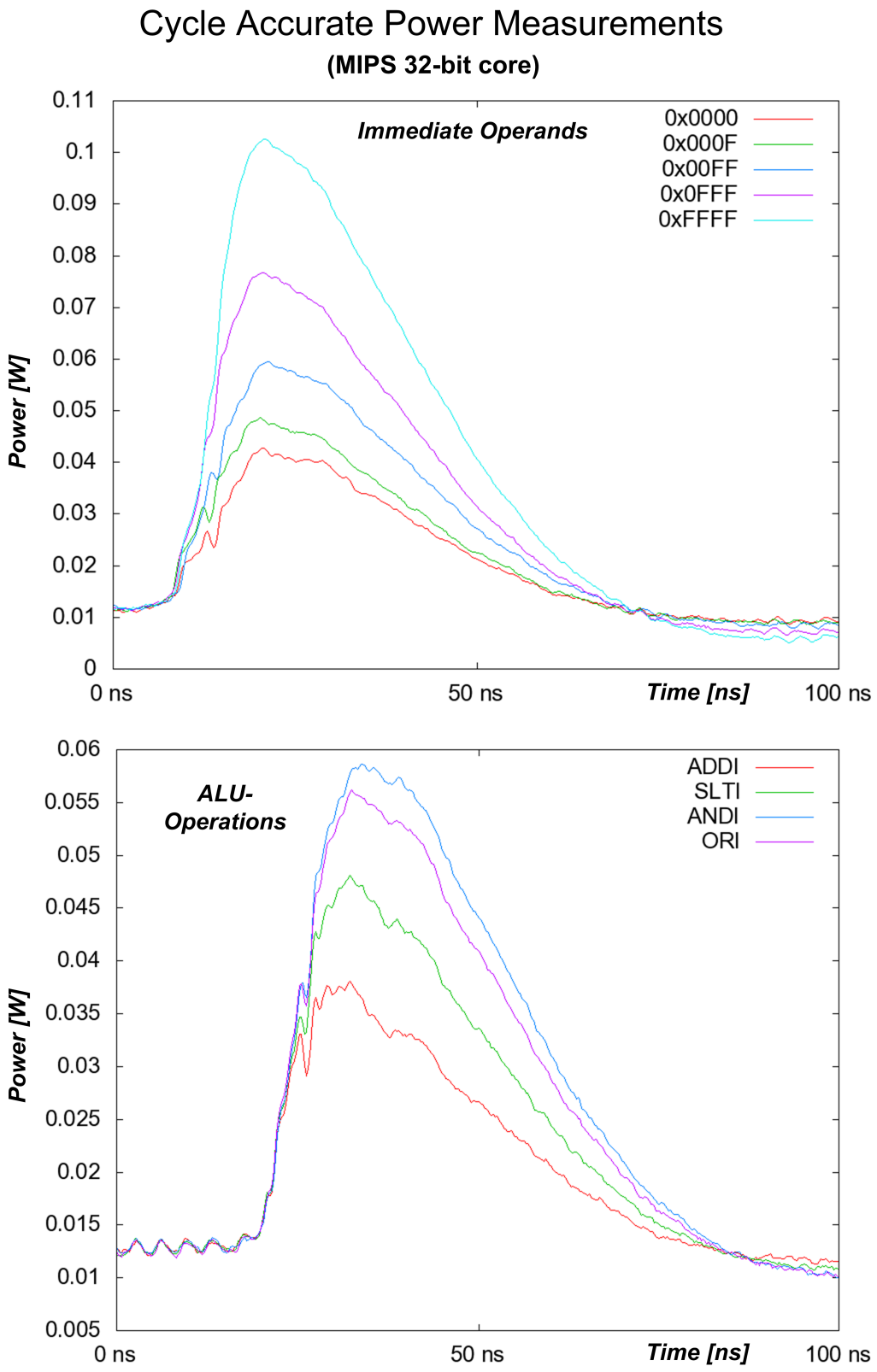



Figure 6.3: Both figures show averaged measurement results of cycle accurate power measurements of the MIPS-core. The measurements showcase the time resolution which can be achieved with the Hyperion platform. The top figure shows that the more bits get toggled by an immediate operand the more power is consumed. The lower figure compares the power consumption of different ALU-operations.

The energy cost depends not only on the immediate value as we can see in the second plot "ALU-Operations" - also in Figure 6.3. This time the immediate values are kept constant but the opcode is changed. Thus different ALU-operations are performed by the processor (ALU = Arithmetic Logic Unit). The power consumption differs among the different instructions but the differences are not so pronounced when compared to the varied immediate values of the upper plot.

All in all it becomes clear that a simple, fixed energy per instruction metric does not model the real power consumption accurately. A correct model must factor in the pipeline state, the instruction operands and immediate values.

### 6.1.3 Discussion of Power Measurement Distortion Sources

When measuring the power with cycle accuracy even seemingly small distortions can have a strong effects since the measured voltages are small and must be acquired with a very high frequency to satisfy the Nyquist–Shannon sampling theorem (independent of the clock speed!).

It was noticed that the internal clock generation of the programmable hardware fabric (an FPGA) induces noise on the power rails of the programmable logic. Therefore, the clock signal was supplied by an external laboratory signal generator.

Another signal distortion occurs when a digital trigger signal toggles. In this instant the captured analog signal is distorted. Thus trigger signals must be timed slightly ahead of time to preserve the signal shape of the power supply. Alternatively, different measurement equipment may be worth a test.

A more fundamental problem is that any change of the sensor node SoC requires a new logic synthesis and mapping which may lead to different results.

Either all measurements are performed with one SoC-design or the designer must provide sufficient constrains to the CAD-tool in order to prevent drastic layout changes when only minor design changes have been carried out.

### 6.1.4 Summary

The reader should remember from this case study that Hyperion can measure the power consumption of a sensor node system-on-chip (SoC) within a clock cycle. Custom digital signals must be generated by a SoC to trigger and mark important system changes.

Furthermore, it should be remembered that certain logical operations take multiple cycles and thus a careful power analysis is necessary.

If an ASIC of the sensor node system-on-chip (SoC) is installed on the Hyperion board then the micro-architectural power consumption can be analyzed straight away and a validated power model can be postulated. Such a validated power model could also be integrated into the software driven design space exploration platform Sensim.

## 6.2 Software driven Platform: A Custom Performance Metric

This case study explains - by example - how specialized analyses can be integrated into the design flow. Why they are needed and what they can look like. To guide the design space exploration a custom metric is first described, then an algorithm is presented to compute it efficiently. Finally, the metric is applied to two mathematical kernels which concludes this case study.

### 6.2.1 Integration of a Specialized Analysis into the Design Flow

Once the *power consumption* of a wireless sensor network has been assessed with the design space exploration platform (= Sensim and Hyperion) it may turn out that a few or even one component stands out.

For a camera tracking sensor network (the central application theme in this thesis) it is likely that components from the computational domain are among them due to the large amounts of 2-dimensional image data which must be processed in real time. Hence, when a designer has decided to optimize a specific (computational) hardware component then he will need dedicated analyses and *custom metrics* to explore the sub-design space.

This case study gives an example of a custom metric for the computational domain (= focus of this thesis) and how it can be derived using the design space exploration platform.

The design space exploration platform and the accompanying design flow are designed to be extensible so that they can accommodate customizations.

To facilitate *customized* exploration the design flow provides standard traces and means to generate custom traces. Once a trace is available it can be fed into external analysis applications to compute (custom) metrics.

Figure 6.4 shows that the Sensim full-system simulator generates traces which record hardware activity, data- and instruction memory access and user-defined (custom) traces.

The hardware activity trace can be used to generate a utilization report. A memory access trace may be fed into a CPU-cache simulator for example. A typical metric computed by a cache simulator is the miss-rate which can be further broken down into compulsory-, capacity- and conflict misses. Depending on the outcome the designer may choose to change a cache's associativity, size or replacement policy. Without these cache specific metrics it would be difficult to guide the optimization of the cache hardware component. A cache is a good example to justify custom metrics.

Sensim has already a cache simulator and is able to integrate 3rd-party cache simulators as well (Dinero IV for example).

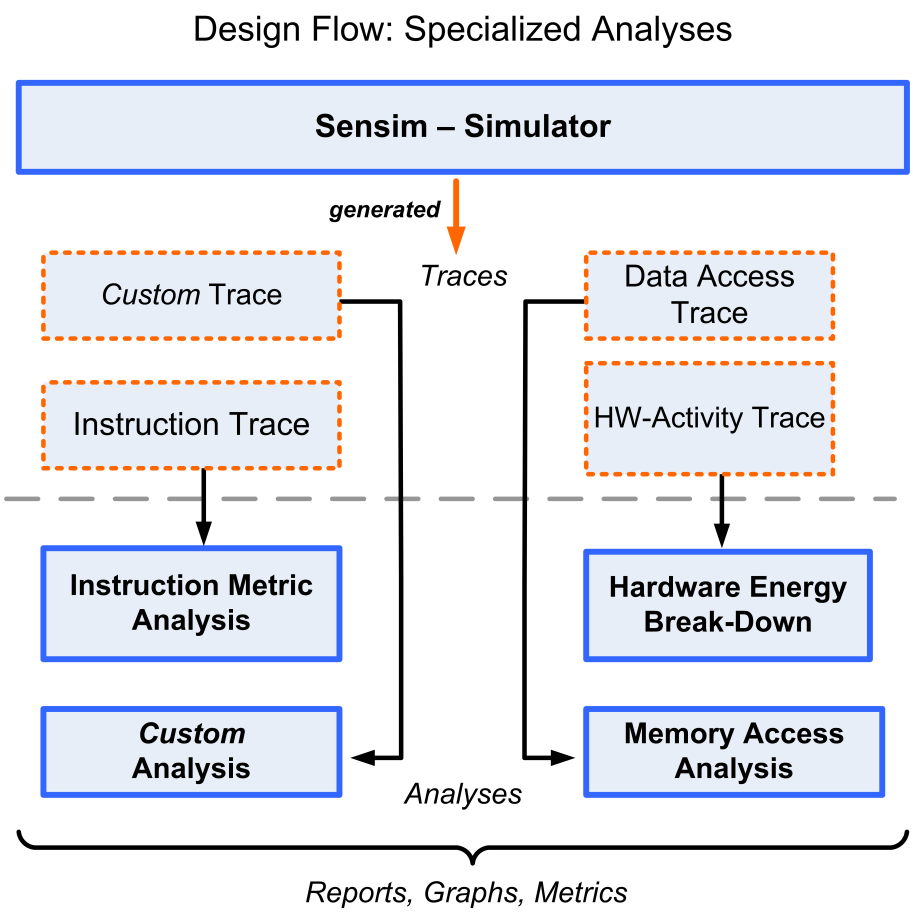



Figure 6.4: The traces which are generated by the Sensim simulator may be fed into specialized analysis components besides the Sensim energy estimator. Examples are memory access or instruction metric analyses. When Sensim identifies hard- and software-components of the sensor node which cause a significant power consumption then specialized analyses are crucial to obtain component specific performance metrics. The designer may extend the simulator to acquire custom traces in addition to the standard traces ( = data access or instruction trace ).

Besides low latency and high memory bandwidth it is important that an image processing application (→*camera tracking sensor network theme*) has access to sufficient computational power. Therefore this case study focuses on a metric for exploring processing optimizations (= computational domain - see pages 73 et seq. for an overview).

In the "Design Space"-Chapter on page 75 various processing elements were enumerated. Common processing elements in the embedded domain are digital signal processors (DSP), general purpose processors (GPP), reconfigurable computation fabrics (FPGAs - for example) and application specific instruction set processors (ASIPs).

ASIP processors have *special instructions* to accelerate applications of a certain application domain. For a sensor node this may encompass instructions for image processing, packet handling, compression and cryptography - not necessarily all in one processor.

A more coarse grained approach would be to integrate dedicated *accelerator cores* into the sensor node system-on-chip (SoC) . Accelerator cores often attach to on-chip interconnects - for example - buses where they can interface autonomously with memory and be shared by multiple (general purpose) processors.

A special instruction is much smaller and less complex than a dedicated accelerator core and therefore *more generic* but less efficient in direct comparison. One distinguishing feature of accelerator cores is that their asynchronous operation- relative to the requesting processor comes natural whereas ASIPs usually invoke special instructions synchronously and thus block temporarily.

Special instructions must be known to the compiler tool chain to utilize them. Accelerator cores - especially if interfaced via a *shared* on-chip interconnect are better invoked by library calls and scheduled by an operating system.

Independent of the granularity, the challenge is to identify potential instruction sequences which are worthwhile to be accelerated. Thus a metric which aids this task would be helpful. Such a metric would have to factor in the complexity of the accelerator ( = its cost ) and its pay off. The hardware complexity could be measured by the number of instructions which are replaced and the pay off could be based on the frequency the accelerator is used. Obviously, most instructions are executed within loops. Therefore the metric must recognize loops independent of their static code structure.

### 6.2.2 The Theory behind the Custom Metric

The proposed *instruction trace metric* keeps track of a limited instruction window (similar to high performance processors) in order to constrain the accelerator's complexity and scope.

A window is be filled up when it contains $n$-unique instruction addresses. Instruction addresses which are repeated in a loop grow a window - granted the loop does not loop over more than $n$-different instruction addresses. If a window is filled-up then a new window is opened until the entire instruction trace has been consumed.

To reflect the frequency of the usage the metric has to be proportional to window's length $w$. By limiting the number of different instruction addresses to $n$ the complexity of the accelerator can be constrained. The metric could be formulate as: $\text{Metric}_n = w$.

Thus an infinite sequential instruction address trace ( = no loops ) would generate infinite amounts of windows of size $n$. To compare the metric for different values of $n$ it is beneficial to normalize to $n$ which leads to the following metric

$$\text{Metric}_n = \frac{w}{n} \begin{cases} W \subseteq T & \\ w := & \text{card}\{W\} \\ n := & \text{card}\{k_i : k_i \in W, k_i \neq k_j \text{ for } j > i, i = 1..\text{card}(W)\} \end{cases} \tag{6.1}$$

where $T$ is the instruction address trace set, $W$ the instruction address trace window, $w$ its length and $n$ the number of unique instruction addresses in the window. Thus the following condition holds:

$$1 \leq n \leq w \leq \text{card}(T)$$

The normalized metric ($\frac{w}{n}$) yields the value 1 for windows with sequential instruction sequences independent of $n$.

Figure 6.5 shows a simple example with a loop. The metrics $\text{Metric}_3$ and $\text{Metric}_5$ have been computed. The application's MIPS assembler source code on the left initializes the register R2 with 8 and decrements the value in a loop (lines 3 to 5). The **or $0,$0,$0** instruction is a no-op. The instruction trace is shown on the right side where the loop is unrolled. The instruction addresses are shown inside the square brackets and the line numbers on the right.

The $\text{Metric}_3$ fills three windows. The first- and last-one contain sequential sequences of instruction addresses and have a value of 1. The second window has a length of 23 instructions with 3 unique instruction addresses and has its center at trace position 14. The value of its metric is $7.67 = \frac{23}{3}$.

For the $\text{Metric}_5$ only one window can be filled. The window contains 25 instructions, 5 unique instruction addresses and has a metric of $5.2 = \frac{26}{5}$.

Many profilers count the number of instructions executed within a function and accumulate the frequencies throughout the call-graph. This allows the designer to spot hot functions immediately. The instruction trace metric however reveals the precise code flow inside and across functions.

### 6.2.3 An Efficient Algorithm to Compute the Metric

Figure 6.6 shows an algorithm to compute the metric from Equation 6.1. The instruction address trace is the input (1).

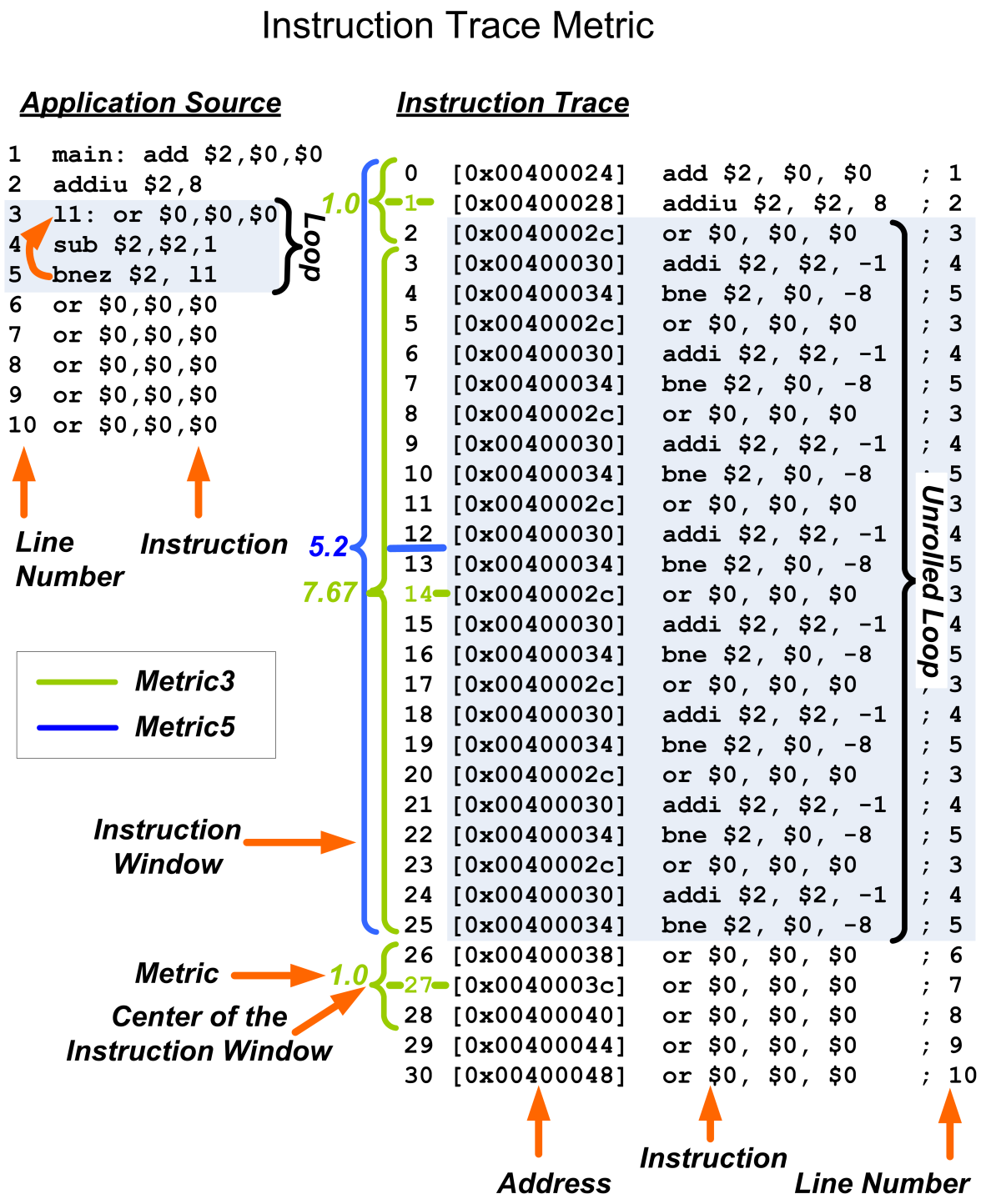



Figure 6.5: The figure illustrates how the metric is obtained. The application source code (MIPS assembly) is shown on the upper left. It runs a simple loop with 8 iterations. The instruction trace on the right shows the flow of instructions when the instructions are actually executed. Depending on the metric ($\text{Metric}_n$) an instruction window may not hold more than $n$ unique addresses. The $\text{Metric}_3$ and $\text{Metric}_5$ are shown in green and blue. The lengths of the instruction windows are indicated as well as their centers where the computed metrics are written. Due to the normalization by $n$ a sequential code flow within a window has always a metric of 1.0.

Algorithm: Instruction Trace Metric

```
/* Input */
addrs                : List of addresses = [ /* address trace */ ]

/* Declarations */
window               : List of addresses = [ /* empty */ ]
window_set           : Set  of addresses = ( /* empty */ )
window_max_addr      : Integer  = 3 /* parameter 2..n */
window_start         : Integer  = 0

/* Loop over the instruction address trace */
for i, next_addr in enumerate( addrs ) {

  /* window full and "next_addr" new ? */
  if len( window_set ) == window_max_addr and
     next_addr not in window_set {

     metric = len( window  )  / float( window_max_addr )

     window_center = window_start +
                     ( ( i - 1 ) - window_start ) / 2.0

     /* print window */
     print window_center, window, metric

     /* new window */
     window_start = i
     window       = []
     window_set   = ()
  }

  /* add address to window */
  window.append(  next_addr )
  window_set.add( next_addr )

}
```

Loop

1 2 3 4 5

Figure 6.6: The instruction trace metric takes a list of instruction addresses as input (1). The instruction trace **window** is a variably sized list which holds **window_max_addr** *unique* addresses (2). **window_set** is a heap which holds each unique address once as opposed to **window**. **window_start** (2) is an index into the input address list. The algorithm loops (3) over the addresses. Each iteration (5) an address is added to **window** and **window_set**. Latter is only changed if the address had not been in the set before. If a window becomes full (4) (= **window_max_addr** unique elements in the **window**) then the metric is calculated and a new window is started. The algorithm outputs a window's instruction addresses (**window**), its center index (=middle) into the input address list (**window_center**) and the metric (**metric**).

In (2) the variable declarations are shown. For a $\text{Metric}_n$ "window_max_addr" has to be set accordingly. In the figure it has been set to 3 thus the $\text{Metric}_3$ is configured. The instruction address trace window is defined as a list of instruction addresses. The window set (a heap data structure) is necessary to determine the number of unique elements - inside a window - within $\mathcal{O}(\log(w))$.

The algorithm loops (3) over the entire instruction trace and adds instruction addresses to the window until it is full (5).

If a window is full (3) then the metric for the window is computed and a new one is created. Besides the metric the algorithm also determines the window's center position within the trace. The position is helpful when the windows are plotted (see Figure 6.7).

Except for the window set which is a heap data structure all operations within the loop are in $\mathcal{O}(1)$. Inserting and finding an element in a heap has a complexity of $\mathcal{O}(\log(w))$ where $w$ is the window's length. Thus the algorithm's complexity is given by:

$$\begin{aligned}
&\sum_{i=1}^{n}(c \cdot \mathcal{O}(1) + 2 \cdot \mathcal{O}(\log(w))) = \\
&\quad \sum_{i=1}^{n}(\mathcal{O}(1) + \mathcal{O}(\log(w))) \subseteq \\
&\quad\quad \sum_{i=1}^{n}\mathcal{O}(\log(w)) \subseteq \\
&\quad\quad \mathcal{O}(\sum_{i=1}^{n}\log(w)) \subseteq \\
&\quad\quad \mathcal{O}(n \cdot \log(w))
\end{aligned}$$

where $n$ corresponds to the length of the instruction address trace ( = $\text{card}(T)$) and should not be confused with the $n$ of the metrics's equation on on page 128 which denotes the number of unique elements inside a window.

### 6.2.4 Application of the Metric to Mathematical Kernels

The first example on page 129 (Figure 6.5) is a trivial case to which the metric had been applied for the sake of comprehensibility. In Figure 6.7 two mathematical kernels (benchmark A and B)[3] with millions of instructions have been analyzed with the metric.

The instruction trace has been plotted against the x-axis. The metric values can be read on the y-axis. The data points correspond to window center positions.

For the benchmark A it can be seen that a tight loop shortly after the start spans nine different instruction addresses (red peak). The loop is not worth optimizing since it spans only a

[3] Mandelbrot set and n-body problem.

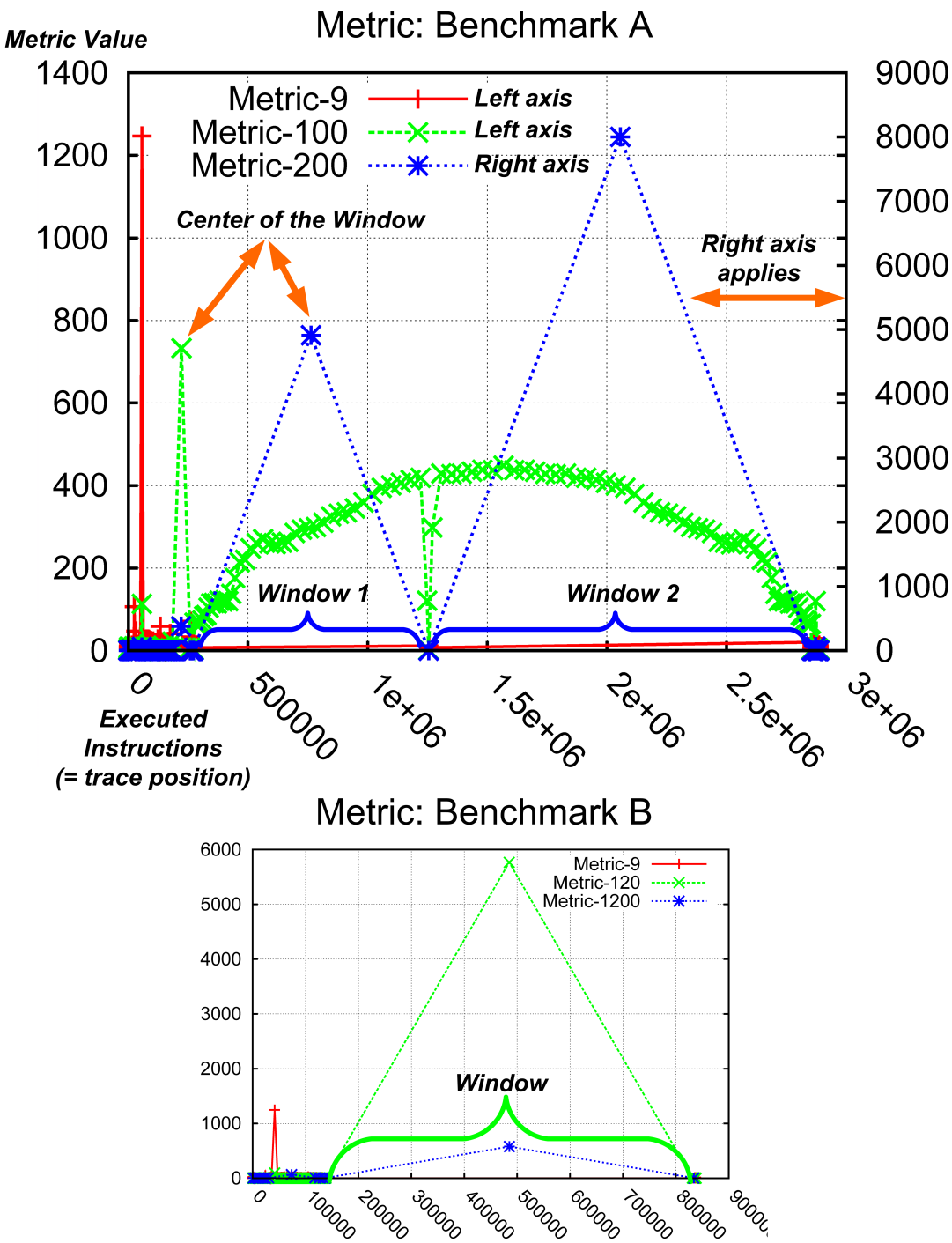



Figure 6.7: The two figures show plots for different $\text{Metric}_n$ where $n$ is either 9, 100, 120, 200 or 1200 (= number of unique instruction addresses in an instruction window). Both benchmarks (A and B) compute mathematical kernels. The x-axis shows the trace position ($\approx$ time). Each data point represents an instruction window's center and its metric (y-axis). For the benchmark A, the $n = 200$ metric has been plotted against the right y-axis. The peaks indicate large instruction windows ($\rightarrow$ loops with at most $n$ different instruction addresses). In benchmark A, window 1 and 2 ($n = 200$) span most of the program's executed instructions; whereas in benchmark B it is a single window with $120 \leq n \leq 1200$. In both benchmarks a tight loop with 9 instructions can be seen shortly after the start (red colored peak).

short range of instructions in the trace. For the $\text{Metric}_{100}$ (green) the main loop fits into 100 instructions for up to 400 iterations. The number of loop iterations follow the shape of a concave function ($f(x) = -x^2$) which has its cause in the algorithm of the benchmark. If the instruction window is extended to 200 instructions ($\text{Metric}_{200}$ - blue plot, right y-axis applies) then the main loop is almost covered completely except for a program phase change between window 1 and window 2.

For the benchmark B a tight loop with 9 different instruction addresses can also be be seen right after the start (red peak, $\text{Metric}_9$). It is probably a library initialization that both benchmarks share. The main loop of benchmark B fits into 120 instructions ($\text{Metric}_{120}$). Further increases of the window size just cover multiple loop iterations and lead to a lower metric value ($\text{Metric}_{1200}$).

Thus the benchmark A has a higher loop complexity than the benchmark B.

In Figure 6.8 the total number of instruction address windows is shown on a logarithmic scale for the benchmark A. The metric has been swept for $n = 8..200$. It can be seen that the number of windows drops non-linearly by two orders of magnitude. Once for $\text{Metric}_{50}$ and again around $\text{Metric}_{120}$. Unlike in Figure 6.7 it is not possible to see that only two windows of $\text{Metric}_{200}$ span almost the entire trace (= benchmark run).

### 6.2.5 Summary

This case study has shown that the design space exploration platform can be extended with custom metrics. Either by using standard- or custom-traces. Latter can be generated by probes (see also on page 110 ff). The presented metric aids the identification of non-contiguous code portions which are frequently executed. These hot spots may be turned into special instructions or accelerator cores ( = computational domain). Custom metrics are necessary to explore the sub-design space of components which have been found to consume a significant amount of power.

The case studies have demonstrated the capabilities of the design space exploration platform. They cover the exploration by software driven simulation and (real time) driven hardware simulation. Latter can interface with real radio transceivers, sensors and memory chips. This enables power measurements under realistic conditions which are a *essential* to guide the exploration process and to validate the power models.

The following chapter concludes the thesis with a summary and a list of challenges left for future work.

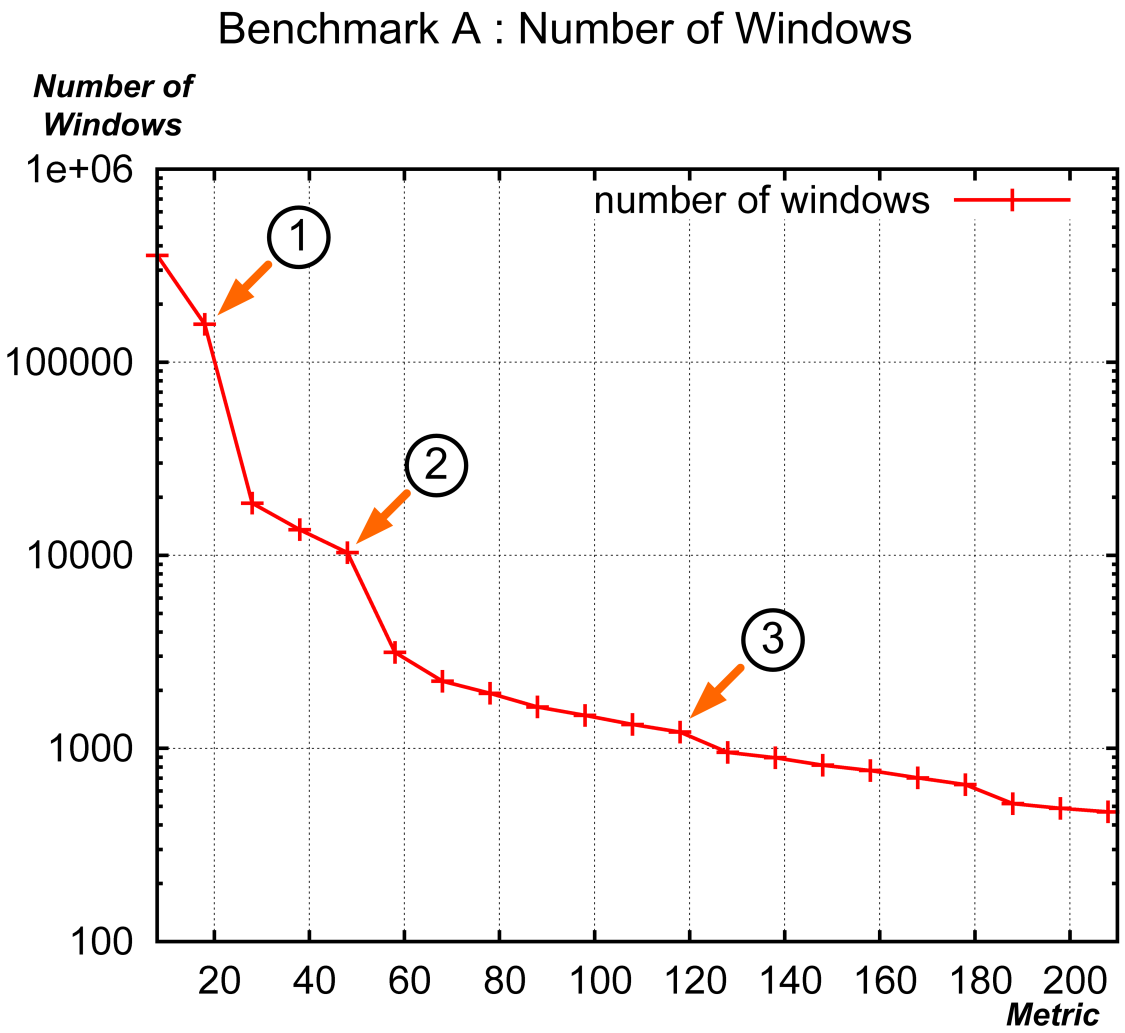



Figure 6.8: The figure shows the number of windows for different $\text{Metric}_n$ (see x-axis) - ranging from 8 to 200. The number of windows drops by two orders of magnitude between $\text{Metric}_{20}$ (1), $\text{Metric}_{50}$ (2) and at around $\text{Metric}_{120}$ (3). The y-axis has a logarithmic scale.

# 7 Conclusion

This chapter concludes the thesis with a detailed summary which highlights the major differences between the state-of-the-art before and after the conception of the flexible design space exploration platform and its extensible design flow. Then the closing remarks section presents important thoughts surrounding the thesis, wireless sensor networks and their exploration. Finally, the outlook outlines unexplored aspects of wireless sensor network design space exploration and makes suggestions to advance the field.

## 7.1 Summary

Sensor network applications are extremely diverse. As a consequence the aperture, density and topology of a sensor network may vary considerably. Of course the sensor node design depends also on the application.

To aid the exploration of the vast design space an exploration platform has been conceived. The platform has a hard- and software driven part. They allow designs with real- and virtual hardware components to be assessed. The designer can start off with a functional model and refine it to a cycle accurate model. The cycle accurate model has the advantage that it can be simulated in real time on the hardware driven architecture. The hardware driven platform can also interface with real hardware components. The power consumption of the real hardware components can be measured accurately. On the software driven platform the power can be estimated with power models.

The design flow allows the software driven platform to integrate 3rd-party simulation components and custom analyzers. In addition to a per hardware component energy break down it is also possible to perform a category based break-down. The categories are designer defined and may encompass activities like routing, signal processing or radio communication.

### Requirements - Life time and QoS

In sensor networks the functional- and non-functional requirements map either on the node- or network-level. On the network level the requirements determine physical aspects of the deployment like density or topology. Good examples of non-physical aspects are algorithms and protocols of the communication layers like routing or media access control (MAC) . On the node level the requirements map to hard- and software components.

The requirements of each project are more or less unique. The design space exploration platform aids the designer in validating whether the functional- and non-functional requirements have been met. One of the most important non-functional requirements in sensor networks is obviously a minimal power consumption.

The design space exploration platform aids the designer in assessing the functional- and non-functional requirements. Some limitations apply though. It is not possible to determine the cost, size or weight of the final design - for example.

The life time of a sensor node or the network as a whole depends on many parameters. Given a certain statistical distribution of power consumption among the senor nodes and limited energy resources it is obvious that the network's performance (QoS) will degrade and eventually fail. Thus the requirements and the achieved power efficiency ultimately determine the capacity and cost of the power sources. Since the requirements are application dependent the designer has only the freedom to improve the power efficiency of the design.

Well known platforms like Avrora/AEON and Power-TOSSIM assume a fixed hardware platform and can *only* be used to explore the design space of software components. T*he design space exploration platform conceived in this thesis explicitly supports both.*

## Design Space of Off-the-Shelve Sensor Nodes

Commercially available sensor node platforms like the Mica, Telos and SunSPOT are conceptually simple as we have seen. They combine off-the-shelve hardware components like microcontrollers, low power radios and (plugable) sensors. The predominant power sources are still (re)-chargeable batteries.

The standard hardware components are usually wired on a printed circuit board which is inefficient in terms of power (distribution) and signaling when compared to a system-on-chip (SoC). A customization of the hardware architecture is possible with standard hardware components but constrained. Every change requires also a PCB-board re-spin.

Wireless sensor networks do - maybe even more than other embedded system - benefit from custom hardware architectures. Especially for tasks that fall into the computational- and communication-domain which are in the focus of this thesis.

## Sensor Node System-on-Chip (SoC) Design Challenges

Since the current (commercial) development platforms emphasize the exploration of the runtime changeable aspects and neglect the hardware side very few publications explore customized architectures for wireless sensor nodes. A few system-on-chip (SoC) designs have already appeared though:

PicoRadio and Spec are well-known SoC-designs within the community. Unfortunately, they never seem to have been really used after the initial laboratory tests and thus few results are

available. The development of custom silicon is obviously time consuming and very costly. As a result none of the custom chips has been accessible or even been reproduced by other researchers to the best of my knowledge.

For comparison: the publication of the Mica and Telos designs which are based on off-the-shelve hardware components has inspired the creation of many clones. Subsequently many researchers have used variants of these platforms for their own experiments.

*To foster the exploration of wireless sensor network system-on-chip (SoC) designs a flexible design space exploration platform has been conceived in this thesis.* The platform uses hardware-in-the loop simulation with programmable hardware logic and plugable exploration boards as well as software simulation to allow the designer to explore the design space without re-spinning a chip or board.

Since the platform itself is based on standard components it can be re-produced comparatively easily and spread throughout the community. Thus custom hardware architectures can now have a much wider exposure and be scrutinized and extended by other researchers.

*This work has therefore the potential to raise the level of research in this area significantly.*

## Power Consumption Exploration

Power consumption is the crucial metric for the performance of a wireless sensor network. Therefore the design space exploration platform has been designed to include precise power measurements of all hardware components. The (average) power measurements encompass but are not limited to: processing elements, memories, sensors, radio transceivers and the external (serial) interfaces.

In addition to the average power measurements it is possible to delve into the microarchitectural level where the power consumption of a single clock cycle is measurable. *This functionality is very rarely found and none of the sensor network development platforms have it to the best of my knowledge.*

Besides the power measurement of the real hardware components, it is possible to estimate the power consumption of virtual hardware components. Power measurements of real hardware components are necessary to refine and parametrize the power models and acquire sensor data.

The (power) assessment and simulation are separated to facilitate *extensibility* within the design flow. Custom metrics - for example - can be integrated without breaking the established design flow.

*To the best of my knowledge this design space exploration platform is the first one with a comprehensive and documented design flow.*

Another advantage of the separation is that simulation results can be re-evaluated with varied model parameters and without re-running the entire simulation.

| Radio Exploration Module | Radio Exploration Module | Telos/SunSPOT |
|---|---|---|
| Frequencies [MHz] | 300-348 or 400-464 or 800-928 | 2.4 GHz |
| Output Power [dBm] | -30..10 | -25..0 |
| Data Rate [kBaud] | 1.2 to 500 | 250 |
| Modulation | 2-FSK, GFSK, MSK, OOK, ASK | O-QPSK |

Figure 7.1: The current radio transceiver (module) of the design space exploration platform is more flexible than the standard transceivers of the Telos and SunSPOT sensor node platform. A more complete comparison and references can be found on page 167 in the appendix. Since the radio exploration module is plugable there is actually no limit to what radio transceivers could be interfaced.

*Thus it is possible to carry out what-if style analyses.* Avrora/AEON - the closest competitor- does *not* have this capability.

An important feature *not found* in other wireless sensor network simulator platforms is the ability to track a task's high-level activities *hierarchically*[1]. In the design space exploration platform software modules can be mapped to designer-defined categories[2]. Care has been taken to allow the specification of categories separately from the application logic. This allows different mappings to be applied without introducing breakage (unlike the Quanto profiler for example).

Categories are assigned to software modules but the platform tracks the accessed hardware components as well. The energy needed to run the software of a certain category together with the energy caused by its hardware accesses is combined into the break-down.

Current simulator platforms break down the energy consumption by hardware component. If a hardware component is shared (for example a processor or a memory) then these simulator platforms *cannot* disambiguate the energy usage.

## Exploration of Radio Communication

As the discussion of radio- and channel-modeling has shown it is non-trivial to simulate radio communication credibility. As we know radio waves propagate in a three-dimensional space where they can be absorbed, reflected, experience refraction, diffraction and scattering. The cellular community has advanced models but they must first be adapted to the special conditions of wireless sensor networks.

[1] The tracking algorithm follows the call chain of the software stack.

[2] Good examples for user-defined categories are routing, packetization, filtering, compression or MAC-layer control.

The effort of creating an accurate model should not be underestimated. It involves the radio transceiver, the antenna, the radio-wave propagation and the terrain of course. The computational overhead is not non-negligible either.

*The design space exploration platform has an additional method of aiding the designer in communication exploration - it can interface with real radio transceivers.*

Thus the design space exploration platform can be integrated into a real sensor network if realistic network communication must be considered in an assessment.

Currently, the design space exploration platform has one radio transceiver exploration module[3]. The radio transceiver is within bounds configurable in many aspects: modulation scheme, data rate, frequency, packet size, error correction algorithm and frequency - see also Figure 7.1. The simulated radio model supports simple ranging and is meant as a place holder for external models.

## The Capability to Explore Advanced Sensor Node Designs

A distinction has been made between basic and advanced sensor nodes. Both types of sensor nodes can be found in the central application theme introduced in this thesis. The central application theme is a camera tracking sensor network which requires basic sensor nodes for communication infrastructure purposes and advanced sensor nodes for image acquisition and processing. They vary in computational-, communication-, and sensing capabilities.

More resources and capabilities entail a higher component power consumption. Since the power sources do not scale[4] as easily it is important - especially in regard to advanced sensor nodes - to explore the design space rigorously in order to identify potential optimizations.

The design space exploration platform is powerful enough to explore both types of sensor nodes. Basic sensor nodes are used in deeply embedded sensor networks. A wide spread assumption is that deeply embedded sensor networks[5] are the only likely scenario where sensor networks are applied since many early and influential publications had been written under this assumption. In deeply embedded sensor networks a common optimization goal is to reduce the sleep mode power consumption. The assumption is that a sensor node is not active most of its life-time ( = concept of duty cycling ). A typical example would be a sensor node which measures the temperature every hour and propagates the results infrequently through the network.

In this thesis a wider span has been chosen. For sensor nodes with high performance processing-, communication- and sensing-components the design space is much larger and many trade-offs must be explored simultaneously. The quantity of variables involved does not permit

[3] A photo is shown on page 146 in the appendix.

[4] Many sensor nodes must rely on batteries or use some form of energy harvesting if the environmental conditions allow it.

[5] Deeply embedded sensor networks are associated with large scale and dense - networks which are composed of physically small and very constrained sensor - nodes - a.k.a.

an exploration based on intuition nor is it possible to optimize a single aspect like the power consumption in sleep mode.

The previously mentioned system-on-chip (SoC) designs PicoRadio and Spec are basic sensor nodes. They have been heavily optimized for low-power and -bandwidth radio communication. A system-on-chip (SoC) for an *advanced* sensor node would encompass broadband radio, powerful application processors and a non-trivial memory subsystems. In most research laboratories it is not possible to realize such designs since the costs for custom silicon is prohibitive[6].

*The design space exploration platform is the only platform which is capable to explore advanced sensor nodes to the best of my knowledge.*

### A Memory Exploration Module for Advanced Sensor Nodes

The memory exploration module Theia[7] (for the design space exploration platform) targets advanced sensor nodes. The different memory types on the board vary in power consumption for standby/active, read/write-accesses, power-on/off transition times and volatility. This allows the designer to experiment with different memory mappings and scheduling algorithms.

The focus is on non-volatile memory types: FRAM[8], MRAM[9] and NOR-Flash. Since those can be selectively powered-down to save power. The wiring between the memories can be changed by uploading a new hardware configuration into the re-programmable wiring fabric. Thus different topologies can be assessed without re-spinning a new board.

*To the best of my knowledge Theia is the first reconfigurable memory exploration module for sensor nodes and the only one to support upcoming memory types.*

## 7.2 The Case Studies

In the two case studies various capabilities of the design space exploration platform have been demonstrated:

In the first case study a custom sensor node system-on-chip (SoC) design for a camera tracking sensor node ( = central application theme ) has been introduced. The SoC has control over a digital image sensor, memory and a radio transceiver among other things. The digital image data is directly streamed into the main memory where the soft-core processor can access the image data.

[6] A top-notch mobile SoC in the current technology has roughly a double figure million dollar price tag - not including the design costs.

[7] The schematics of Theia and a photo can be found on page 158 et seq.

[8] Ferroelectric RAM

[9] Magneto-resistive Random Access Memory

| Hardware Utilization Summary | Used | Available | Utilization |
|---|---|---|---|
| Number of Slices | 1871 | 3584 | 52% |
| Number of Slice Flip Flops | 743 | 7168 | 10% |
| Number of 4-input LUTs | 3798 | 7168 | 52% |
| Number of BRAMs | 4 | 16 | 25% |
| Number of GCLKs | 5 | 8 | 62% |
| Number of DCMs | 1 | 4 | 25% |

Table 7.1: The table shows the hardware utilization of the programmable hardware fabric when the camera tracking SoC from the case study is configured. The number of slices corresponds to logic-gates. BRAM is a on-chip memory resource. GCLK and DCMs are clock resources. The underlying device is a Xilinx XC3S400. Currently, there are enough resources to extend the design. If more resources are needed a bigger FPGA must be used. The XC3S400 is not particularly big. The top-notch chips which may be considered in future revisions can easily be configured with a dozen RISC cores e.g.

Figure 7.1 shows the utilization of the programmable hardware logic when the design is loaded. There is sufficient room for custom extensions left[10].

The SoC is specified in a common hardware specification language (VHDL) and can easily be changed. Except for area constrains the designer may extend and modify the baseline SoC as needed.

The case study uses the camera tracking system-on-chip (SoC) to demonstrate the cycle accurate power measurement capabilities. Various instruction patterns of the soft-core processor have proven to exhibit different power profiles. The designer can use the findings to optimize micro-architectural aspects or the compiler code generation.

The second case studies demonstrates an example in which the design flow is extended to compute a custom metric:

Custom metrics help to explore the sub-design space of individual components - for example the processor. The presented metric identifies code hot spots of a specified size that do not need to be contiguous. These hot spots in turn are candidates for custom hardware accelerators in a sensor node system on chip (SoC). An efficient algorithm to compute the metric has been introduced and two mathematical kernels were analyzed to illustrate the results of a non-trivial example.

## 7.3 Closing Remarks

The design space exploration platform already covers many aspects of wireless sensor networks. The platform and the design flow have been developed with extensibility in mind so that new

[10] The XC3S400 programmable hardware chip had the biggest logic capacity in a package that our electronics laboratory could handle. The state-of-the art chips have a manifold capacity.

| | Sensim | Senos | Senos - Core |
|---|---|---|---|
| Physical Source Lines of Code | 29,709 | 97,389 | 2,524 |
| Person-Months | 84 | 294 | 6 |
| Total Estimated Cost to Develop (USD) | $950,986 | $3,308,088 | $71,423 |

Table 7.2: The estimations in this table have been derived for the software components of the design space exploration platform. Sensim is the simulation component which includes a full-system simulator. Senos is the configurable system software library which partially includes 3rd-party source code. The Senos core contains the scheduling algorithm among other things.

metrics, power models and hardware components can be integrated in a systematic manner. The case studies have demonstrated this capability.

The low-level details of the platform are well-documented in various project files which have been created for this thesis. For the interested readers the schematics and board photographs of the hardware driven platform have been included in the appendix.

Figure 7.2 shows the development cost- and time estimations for the software components of the design space exploration platform alone. The estimations[11] are based on the basic COCOMO model and assume an average salary of roughly USD 56,000/year. The cost of the hardware development has not been assessed but the bill of materials for the prototype board is around USD 400 - not including the exploration boards with image sensor and radio transceiver.

I hope that the concepts laid out in this thesis are picked up in future platforms and that exploration and sharing of sensor network system-on-chip (SoC) designs becomes more wide spread within the community.

The basic sensor nodes with their *limited capabilities* have caused many researchers to be hesitant to experiment with real hardware components. Many retreated to software simulation only. Fortunately, the unreflected usage of (3rd-party) simulators has been criticized in the community. Some members emphasized that results are only credible when they are validated against the ground truth. My platform has the capabilities to explore and assess *advanced*- and *basic* sensor node designs within a reasonable amount of time. The measurements based on the real hardware components provide a ground truth.

Besides the conception of the exploration platform and its flows, it has been a special concern of mine to familiarize the reader with the challenges encountered in simulating radio wave propagation. The cellular community has - as mentioned before - already quite advanced models. It would be desirable to have simulators of the quality of commercial network planning software

[11] The estimations are based on SLOCCount version 2.26.

which is based on 3D ray-tracing and realistic terrain databases.

## 7.4 Outlook

The programmable hardware allows almost arbitrary sensor node system-on-chip (SoC) designs to be realized. For communication exploration a configurable radio transceiver is already available and has been successfully used. Although many parameters can be varied the current radio transceiver is still not as flexible as the programmable hardware fabric is for the system-on-chip (SoC)-design (see Figure 7.1).

Hence it would be straightforward to extend the hardware driven platform with a s*oftware defined radio* exploration module. A software defined radio - by definition - allows the designer to control all aspects of signal generation.

Besides signal generation the frequency band plays a major role which is often underestimated. The choice of frequency determines the range, susceptibility to interference (WLAN for example) and the physical properties of the antenna. Certain frequency ranges lend themselves to special applications due to their physical properties. If RF-localization - for example - is required then UWB (ultra-wide band) offers intrinsic advantages over other frequency ranges. Thus seemingly unimportant design choices like the frequency band can have a strong impact on the overall performance. The usage of UWB (ultra-wide band) and similar technologies is still very limited but has the potential to shape the future of wireless sensor networks.

The focus in this thesis has been on the exploration of the communication and computation domain. The exploration of the power supply domain has been left for future work. The design space exploration platform has a fixed power supply which has been optimized for highly accurate power measurements. Therefore, the power supply is neither very power efficient nor adjustable. Nevertheless, it would be an interesting challenge to explore the feasibility of a programmable and freely adjustable power supply exploration platform. If the current exploration platform would be complemented with a fully adjustable power supply and a software defined radio then all relevant design domains could be assessed together.

# 8 Appendix: Hardware Driven Platform

## 8.1 Hyperion

Hyperion - see Figure 8.1- is the part of the design space exploration platform. A programmable hardware fabric holds the sensor node's system-on-chip (SoC) design. For convenience two low power SRAMs are already on board. Hyperion can be extended with exploration modules which hold sensors or radio transceivers (see Figure 8.2 and 8.3).

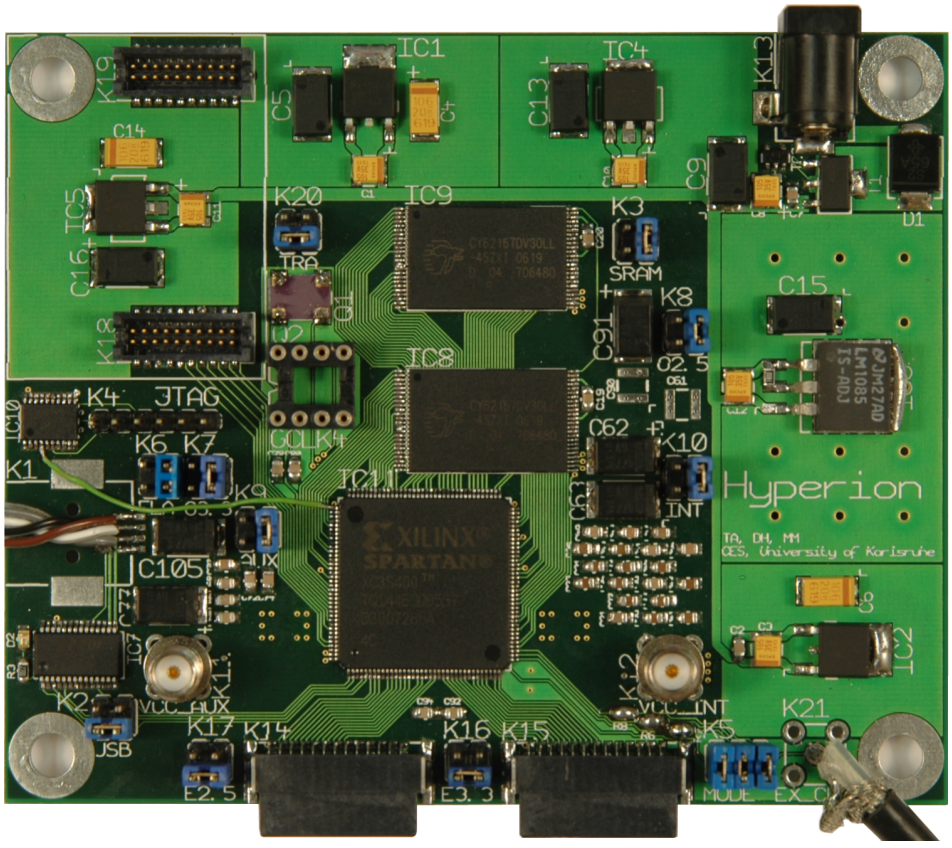



Figure 8.1: Hardware Design Space Exploration Platform Hyperion: - 1st revision (6-layers) - The Hyperion board has a Spartan-3 FPGA (programmable hardware fabric) at its lower center. Above it are two low power SRAMs. At the bottom two extension ports can be seen. The radio transceiver expansion port is in the upper left. The coax plugs are for clock input and cycle accurate power measurements.

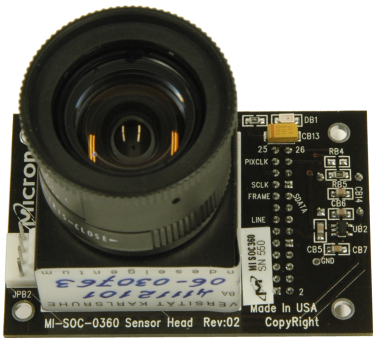



Figure 8.2: Sensor Exploration Module: a Micron MISOC-360 module. It has a CMOS image sensor (MT9V111) with a resolution of 640x480 pixels. The sensor is capable of taking 30 pictures/second.

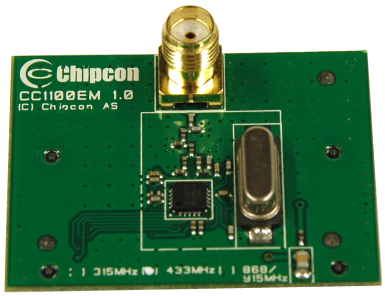



Figure 8.3: Radio Transceiver Exploration Module: a Texas Instruments CC1100 ultra-low power radio transceiver. The maximum data rate is 256 kbit/s. Depending on the board population 300-348 MHz, 400-464 MHz or 800-928 MHz are possible. The photo shows a 400-464 MHz module. Many parameters are configurable - for example various modulation schemes. Furthermore the CC1100 has various power modes with different wake-up times and savings. Due to the high integration only a few passive components are needed.

### 8.1.1 Schematics

The figures on the following pages show Hyperion's schematics (Page 1-7):

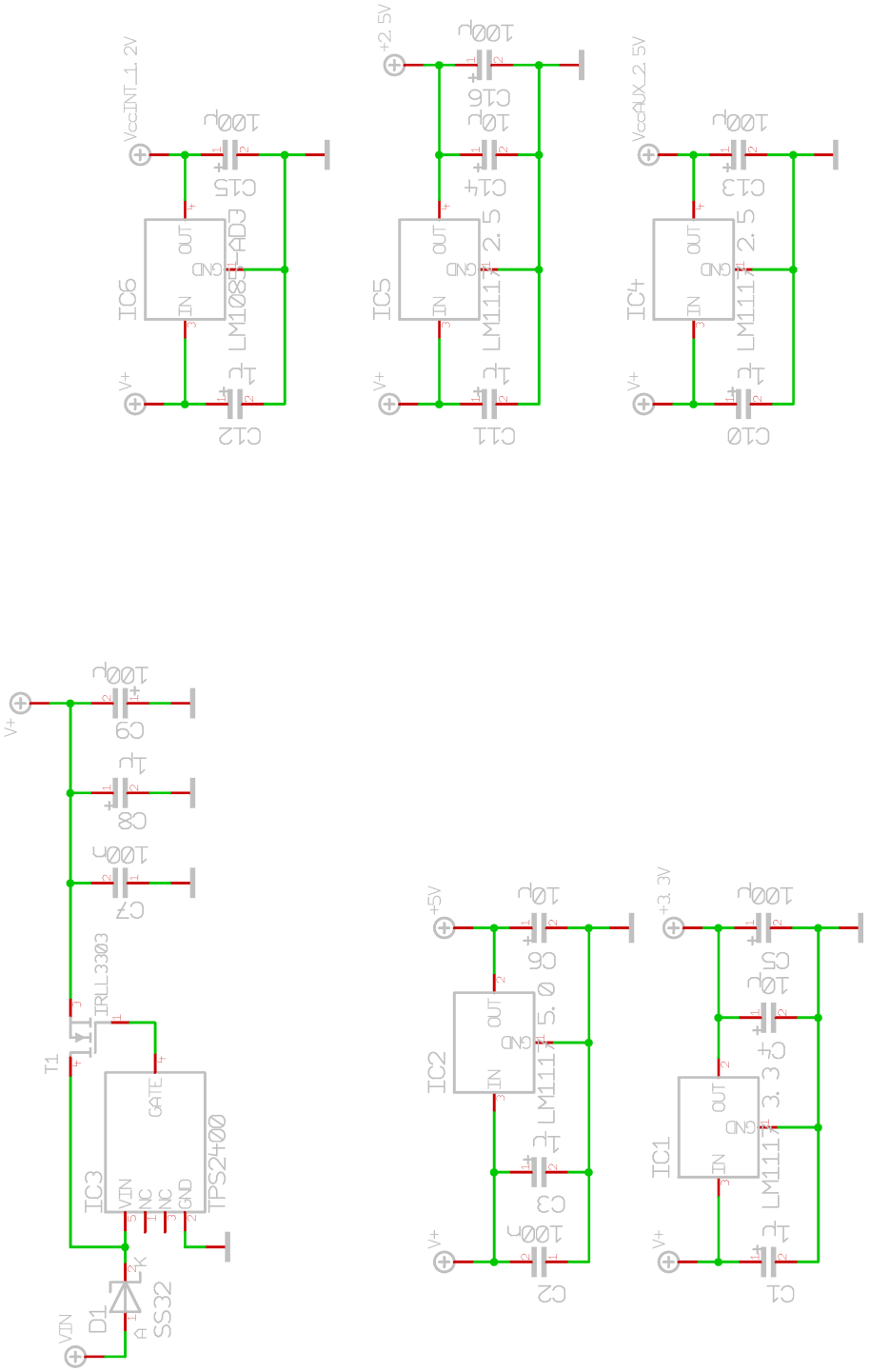



Figure 8.4: Hyperion Schematic - Page 1 - The different linear voltage regulators are shown as well as the over- and under-voltage protection circuit.

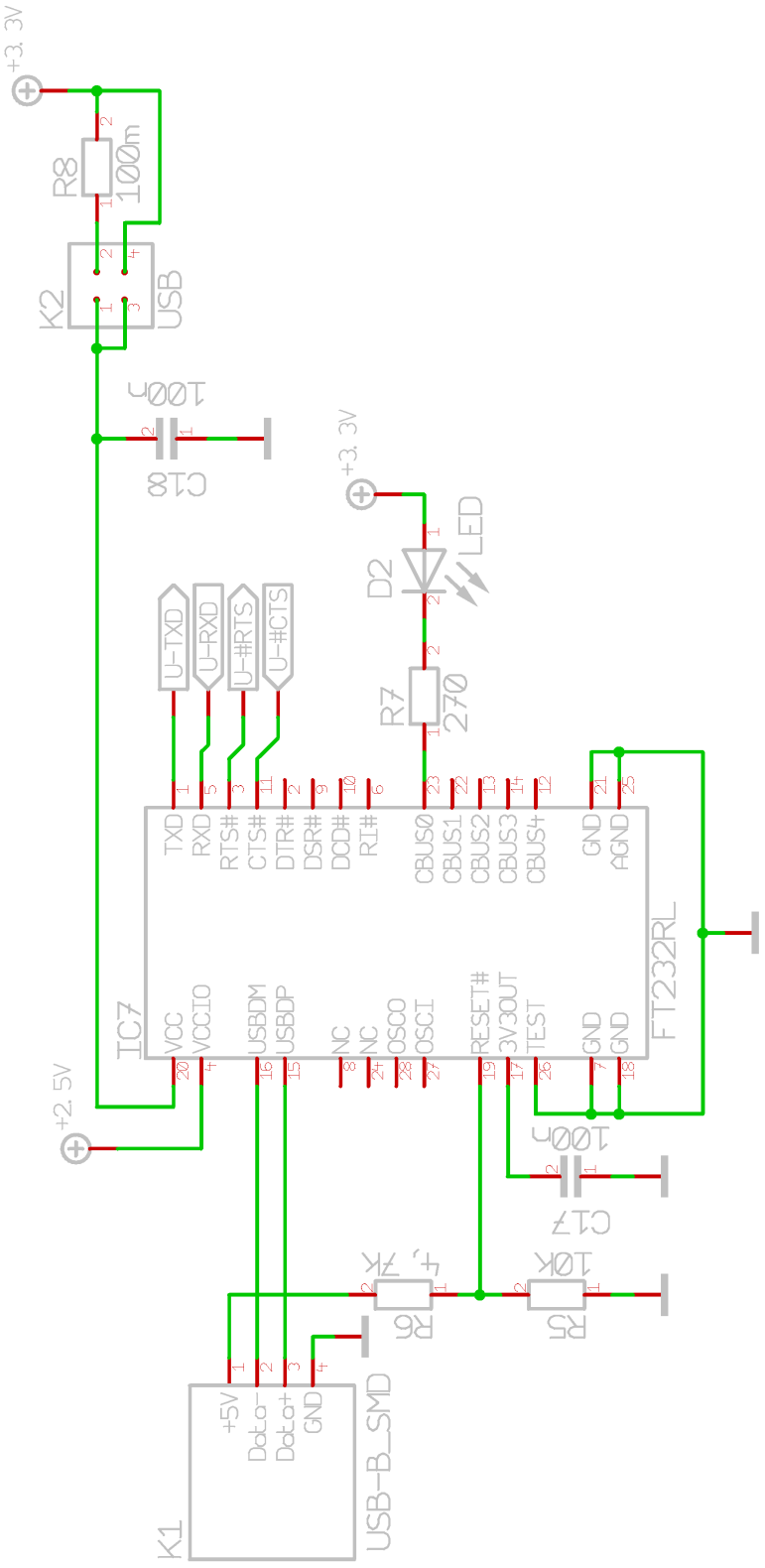



Figure 8.5: Hyperion Schematic - Page 2 - The USB/Serial interface is shown.

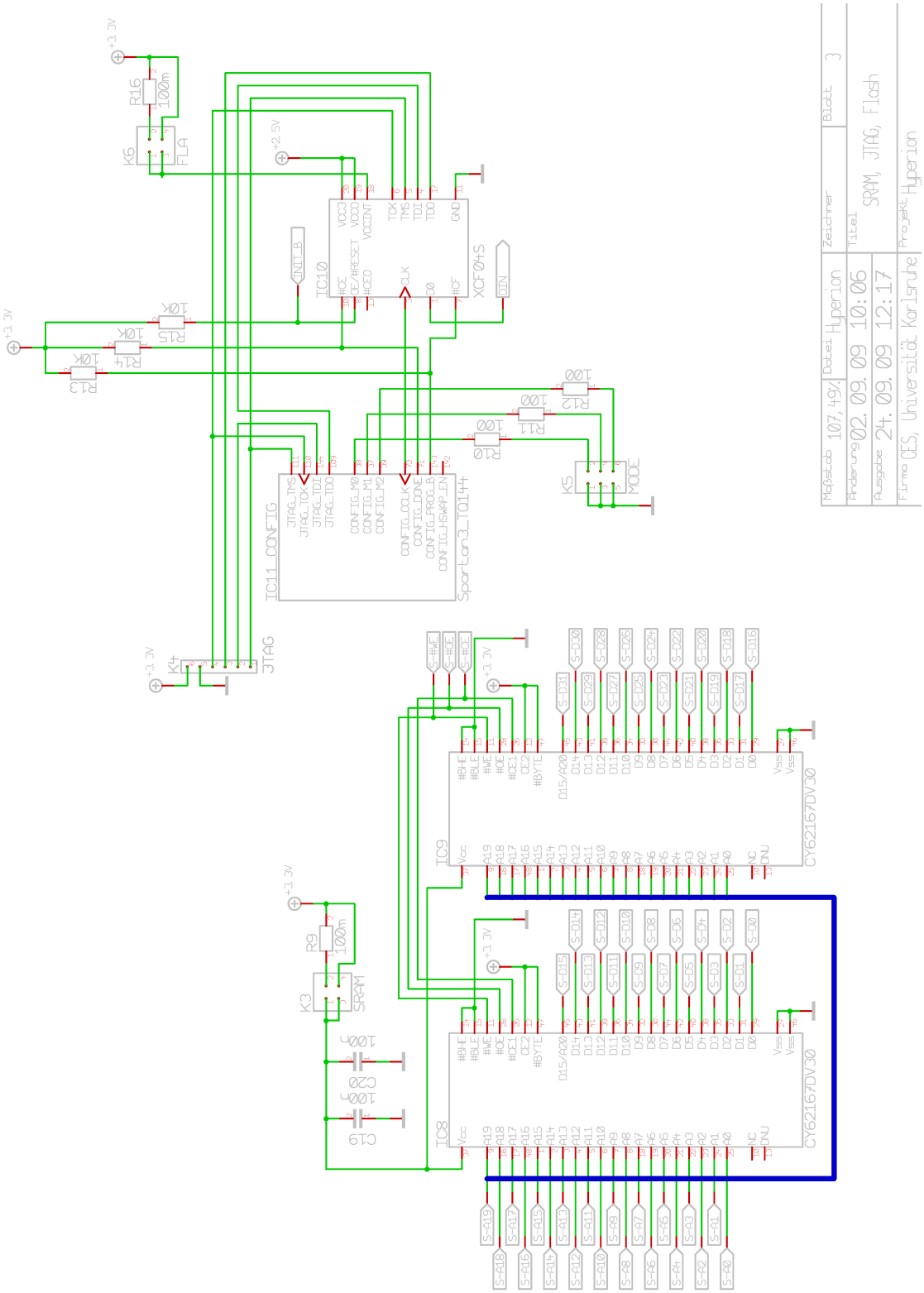



Figure 8.6: Hyperion Schematic - Page 3 - The two SRAMs are shown as well as the JTAG- and PROM-FPGA-interface. Two jumper blocks and measurement resistors can be seen too.

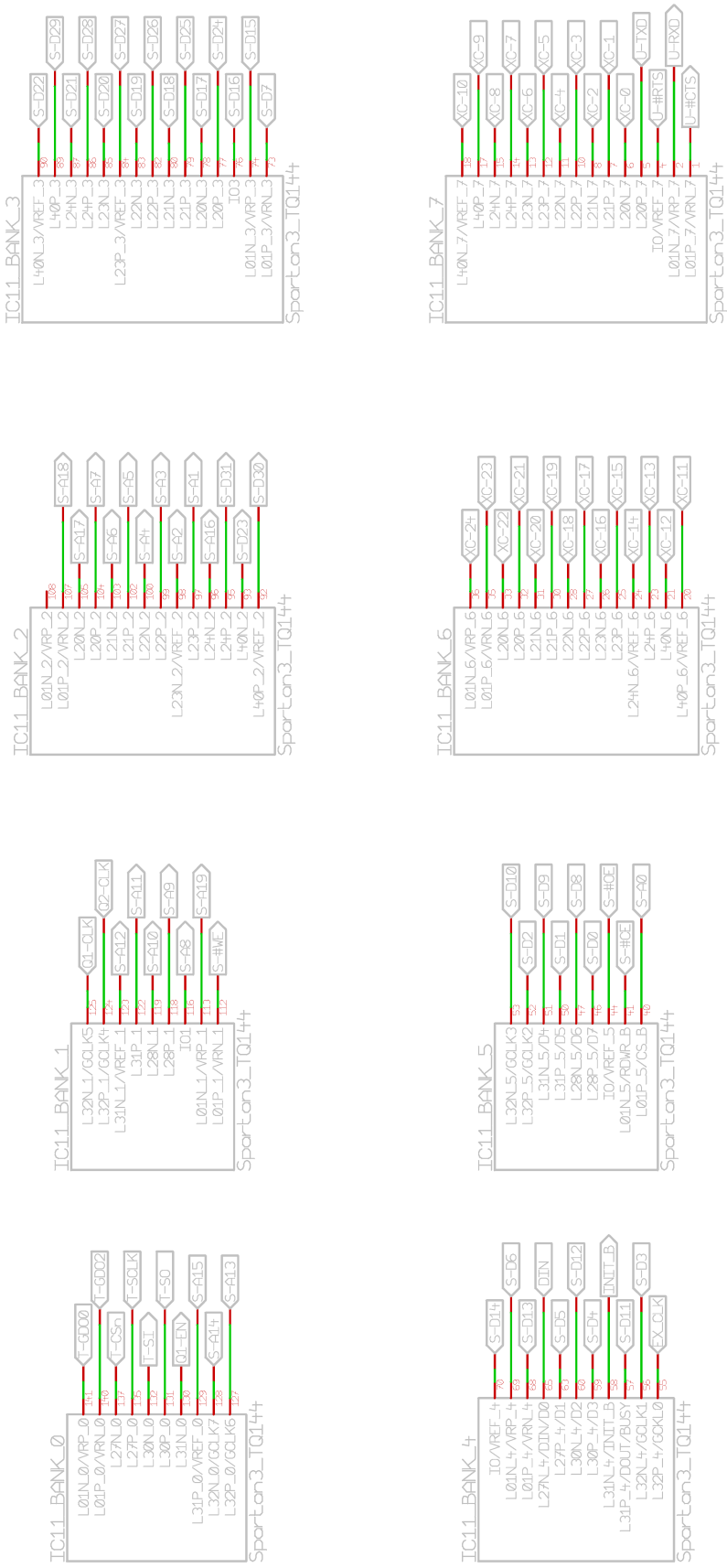



Figure 8.7: Hyperion Schematic - Page 4 - The I/O-banks of the Spartan-3 FPGA are shown.

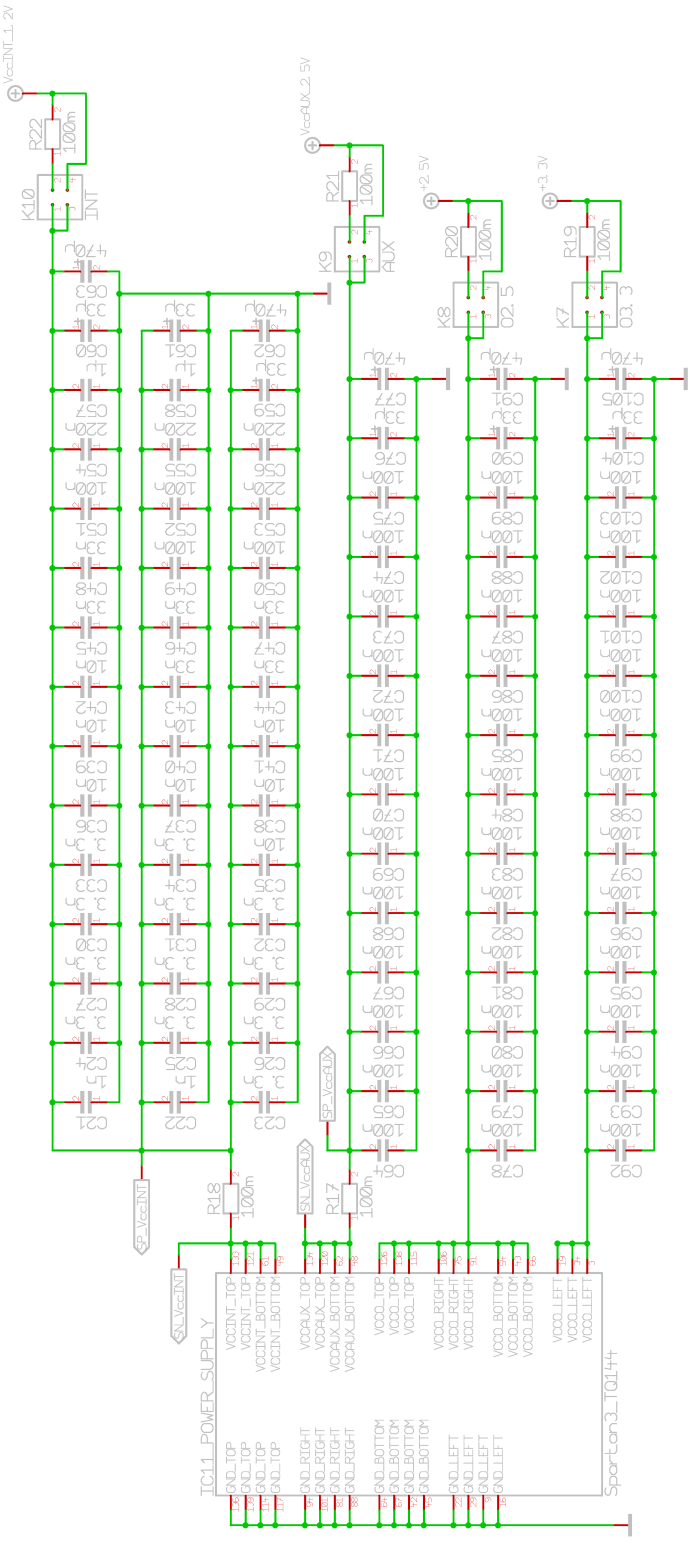



Figure 8.8: Hyperion Schematic - Page 5 - The FPGA's power supply is shown. Besides the ground lines the FPGA needs three different voltages: VCC_INT, VCC_AUX and VCCO. VCC_INT is for the the logic fabric, VCC_AUX for configuration and clock power. VCCO supplies the I/O ports. The decoupling capacitor networks provide a stable power supply and handle short spikes of power consumption. *R*17 and *R*18 are measurement resistors for cycle accurate power measurement. *R*19-*R*22 are for average power measurements.

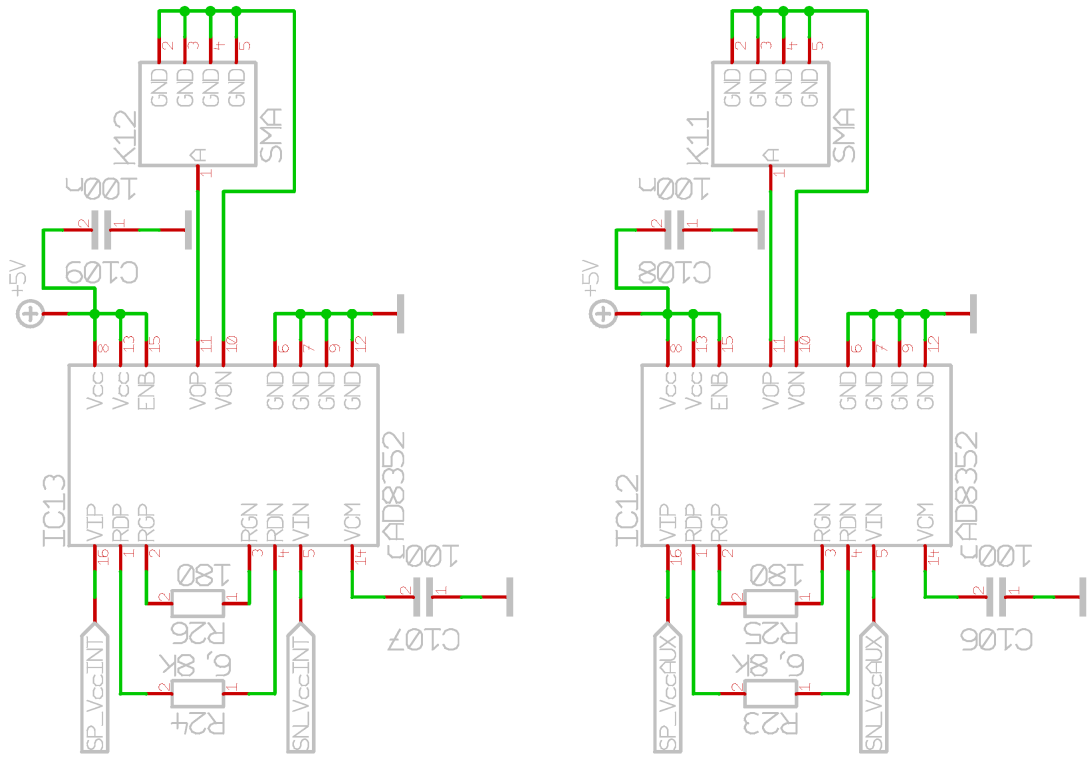



Figure 8.9: Hyperion Schematic - Page 6 - 2 GHz ultra low distortion differential amplifiers are shown. They have initially been used for cycle accurate power measurements. The second revision does not use them anymore and interfaces with the laboratory equipment directly.

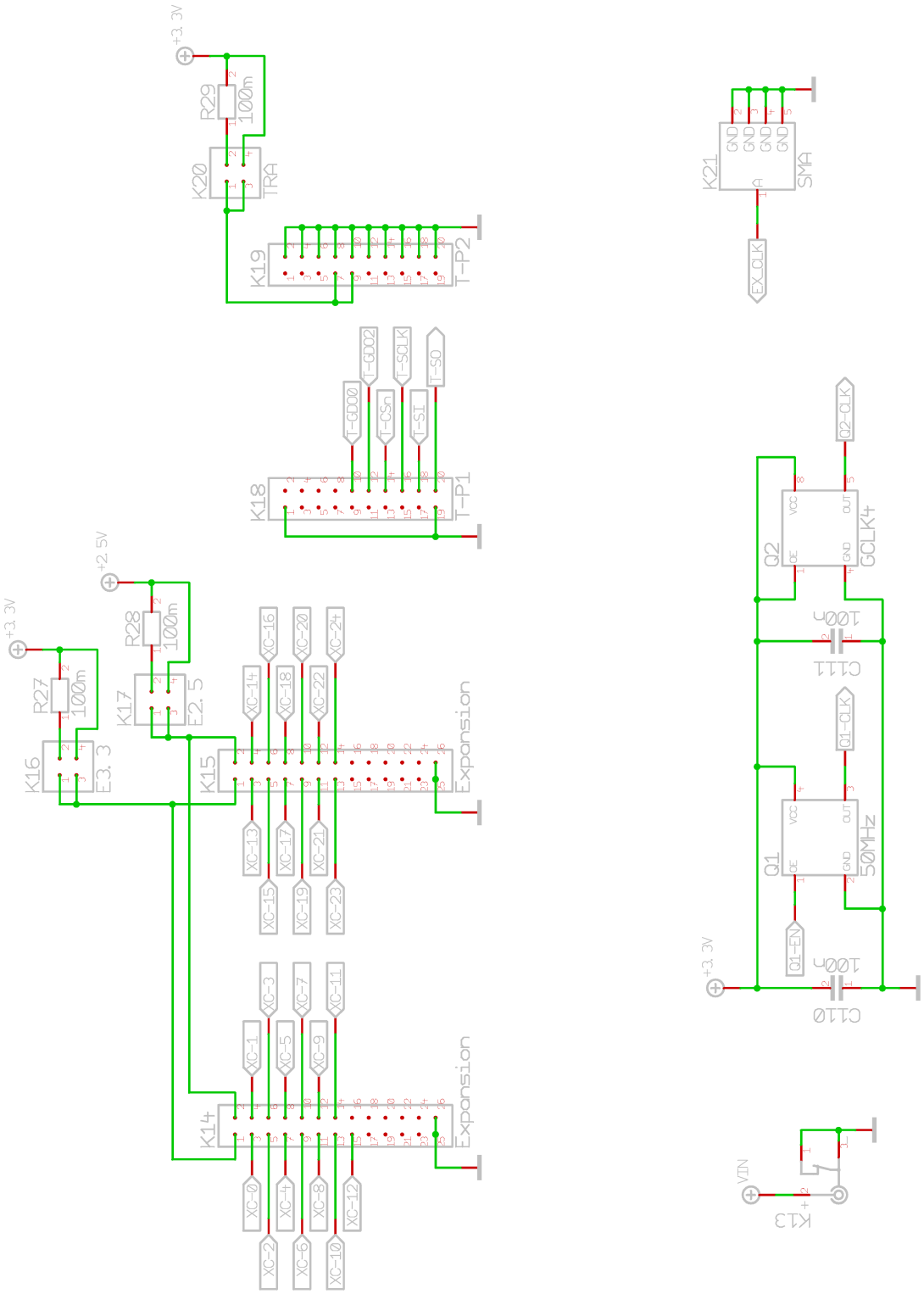



Figure 8.10: Hyperion Schematic - Page 7 - The expansion ports for exploration modules and the clock sources are shown. The clock can be sourced from an on-board oscillator, or a socket-ed oscillator or via a coax-plug.

### 8.1.2 Layout

The Hyperion board has six-layers. The two ground planes are not shown.

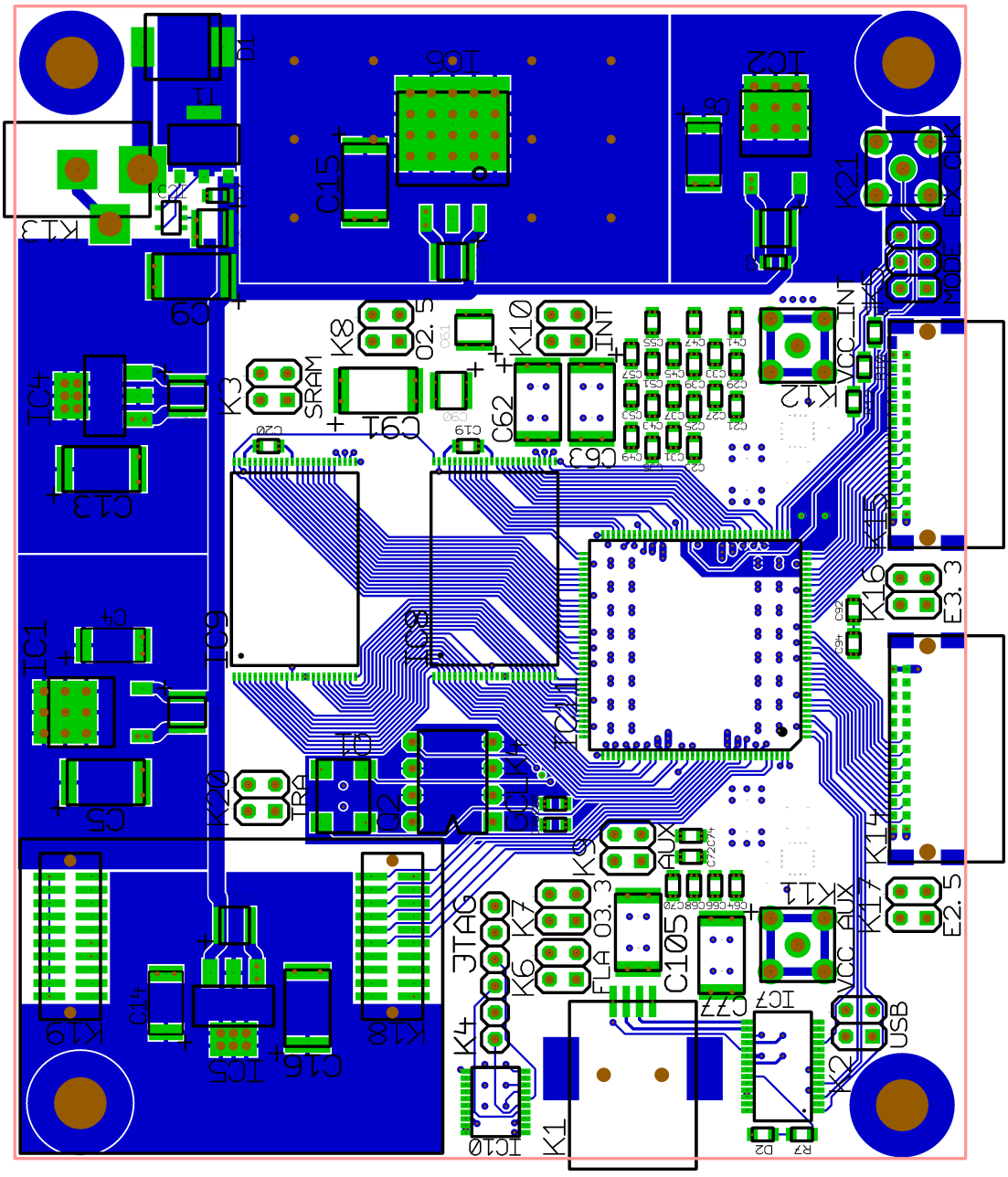



Figure 8.11: Hyperion - Printed Circuit Board - Top Layer

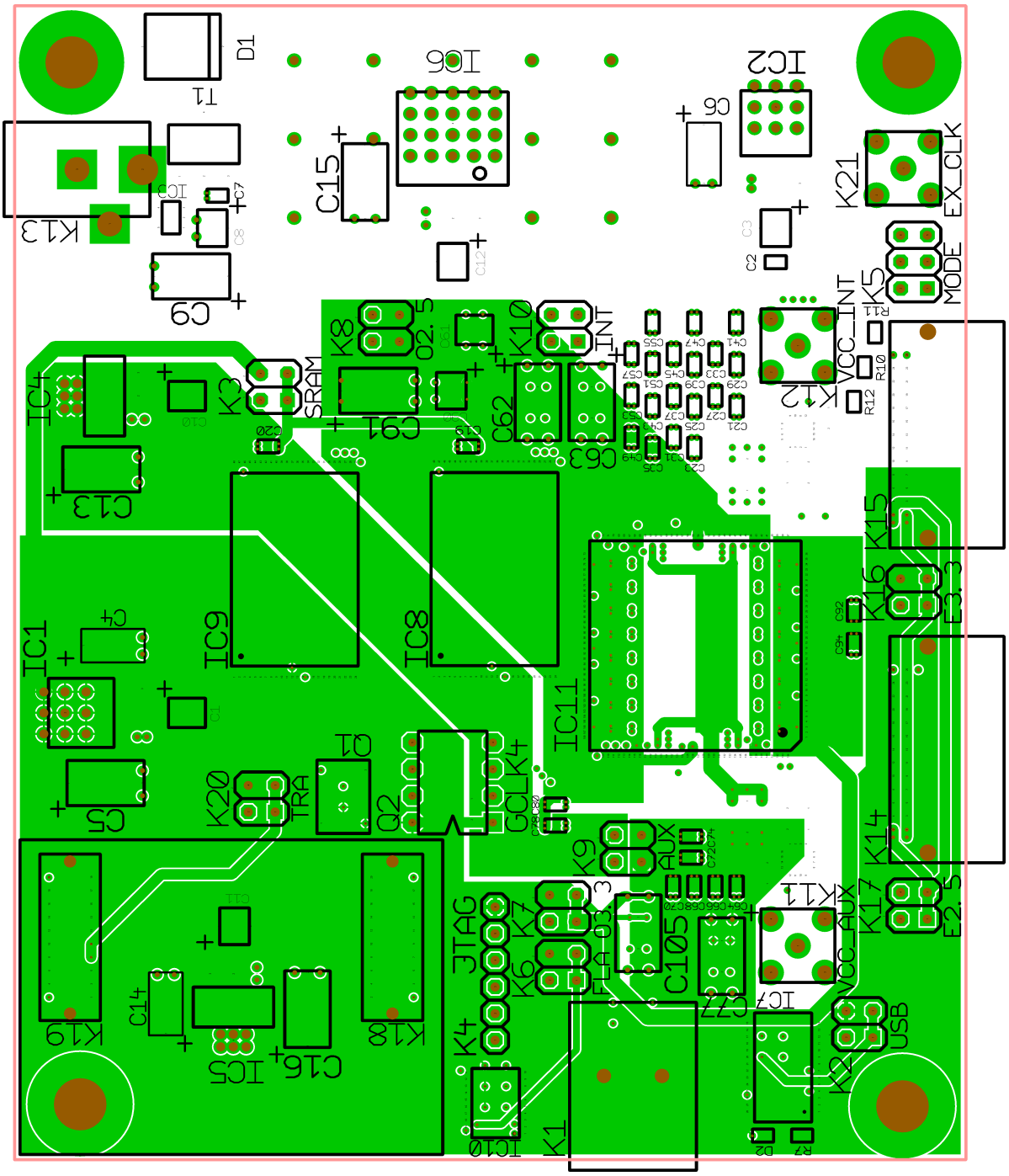



Figure 8.12: Hyperion - Printed Circuit Board - Inner Layer 1

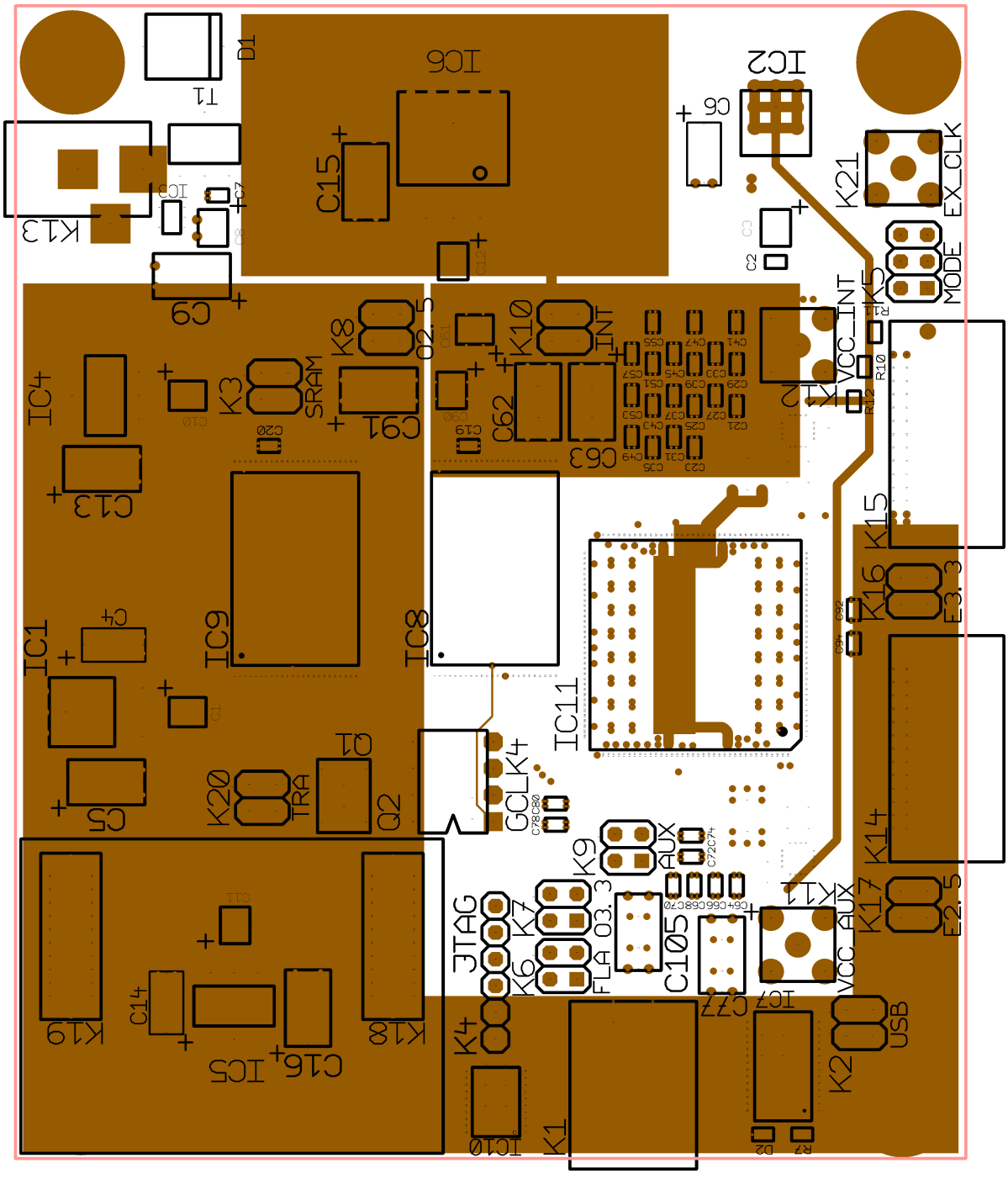



Figure 8.13: Hyperion - Printed Circuit Board - Inner Layer 2

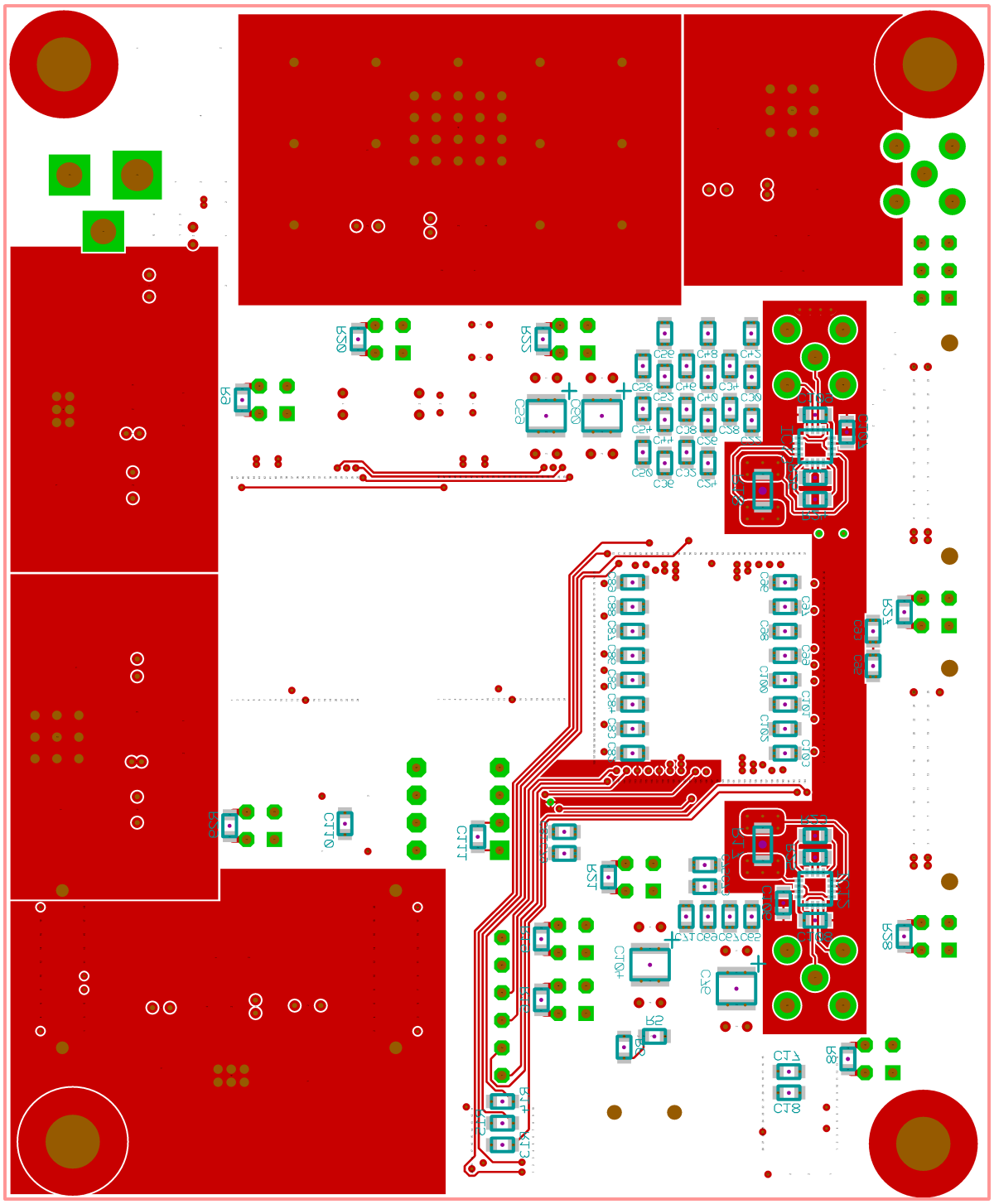


Figure 8.14: Hyperion - Printed Circuit Board - Bottom Layer

## 8.2 Theia

Theia (see Figure 8.15) is a memory exploration module for Hyperion. It holds up to three different memory types. The focus is on non-volatile memories like MRAM, FeRAM and FLASH. Theia has its own power supply and measurement circuit so that it can also be used with other boards besides Hyperion. The connection topology can be re-configured because all three memories are connected to a re-programmable wiring fabric (a CPLD).

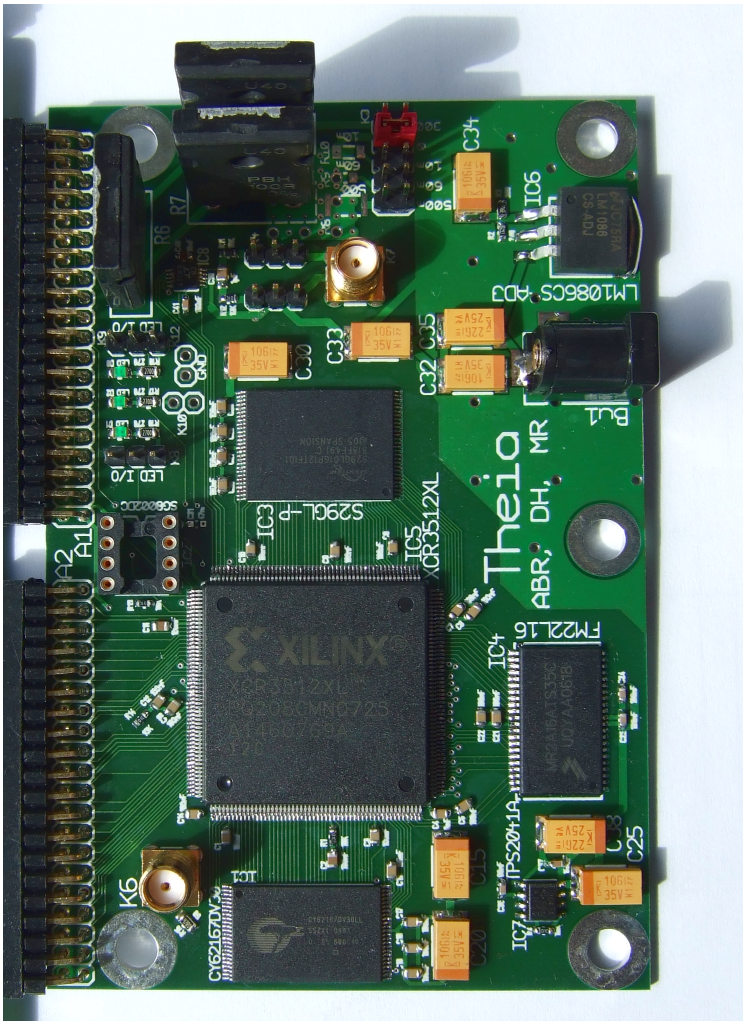



Figure 8.15: Theia - The picture shows the first revision of the memory exploration module Theia. The center chip is a programmable wiring fabric, a CPLD. It connects to the three memory chips on the top, bottom and right. The external connectors are shown on the left. The coax plug can be used to source an external clock signal. Theia has on-board power measurement circuitry. Differently dimensioned measurement resistors can be selected. Optionally, a on-board A/D-converter can assess the power consumption. Some memory chips can be powered-down if they cannot do it by themselves.

### 8.2.1 Schematics

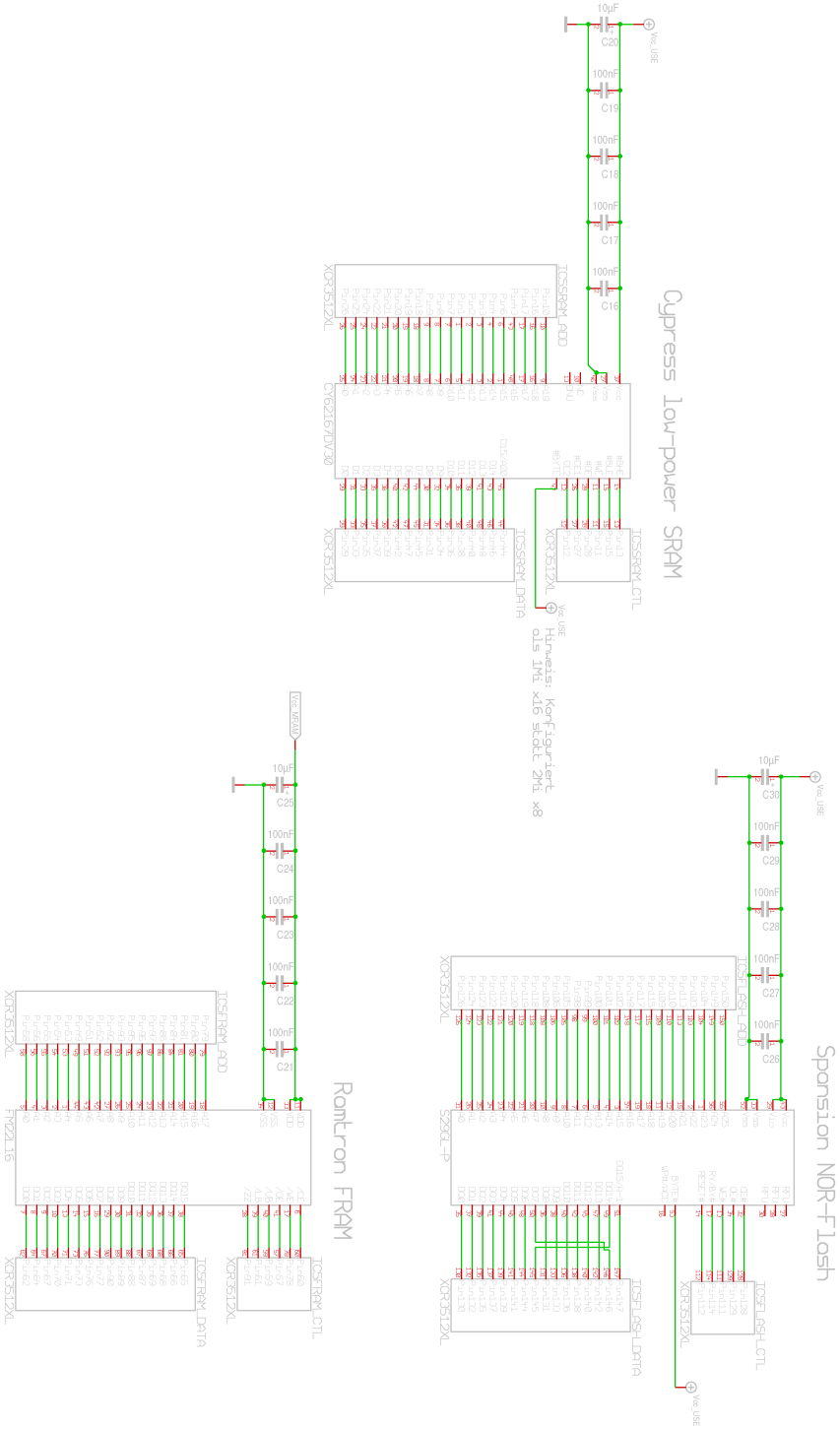



Figure 8.16: Theia Schematic - Page 1 - The memories are shown. This board version has a Cypress low-power SRAM, a Spansion NOR-FLASH and a RAMTRON FRAM.

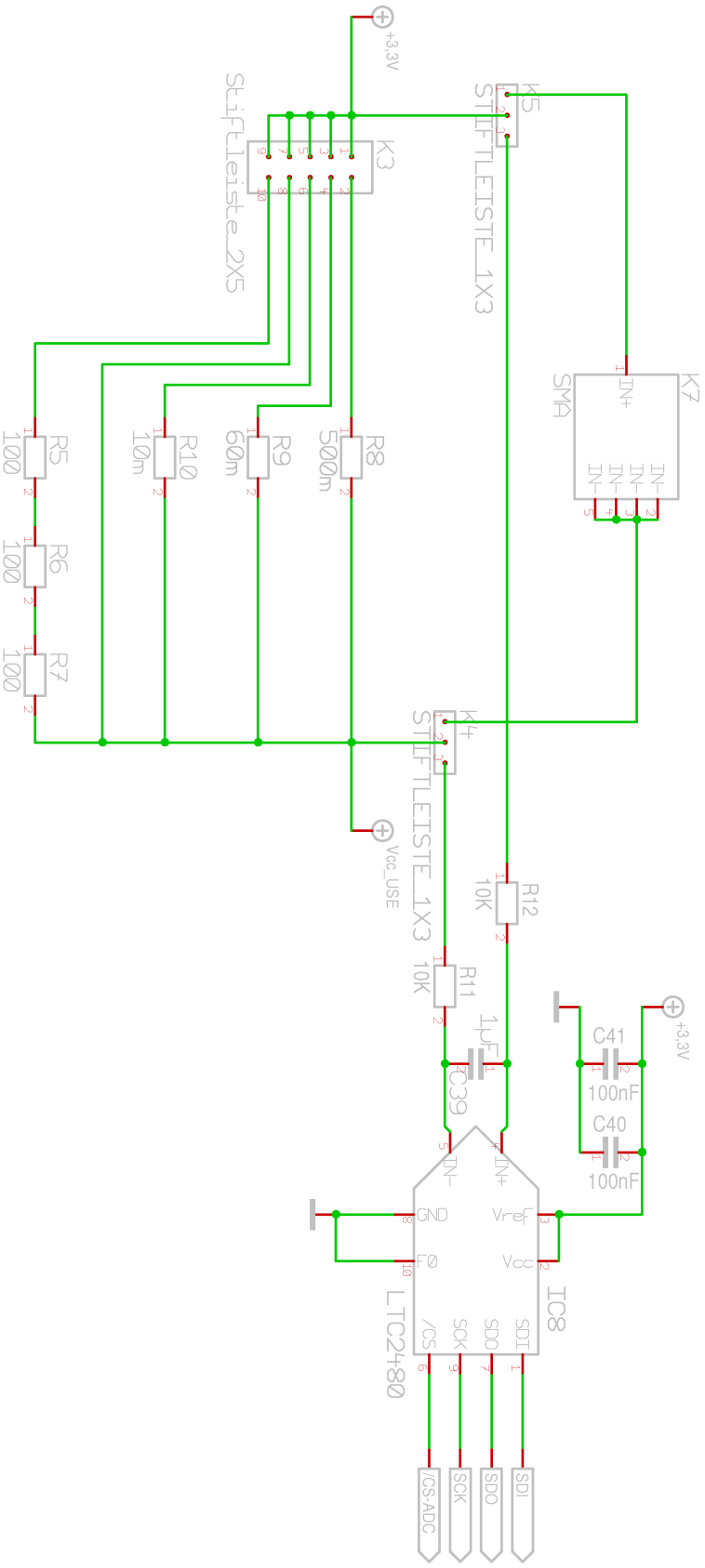



Figure 8.17: Theia Schematic - Page 2 - Power Measurement: The on-board selectable measurement resistors and the A/D-converter are shown.

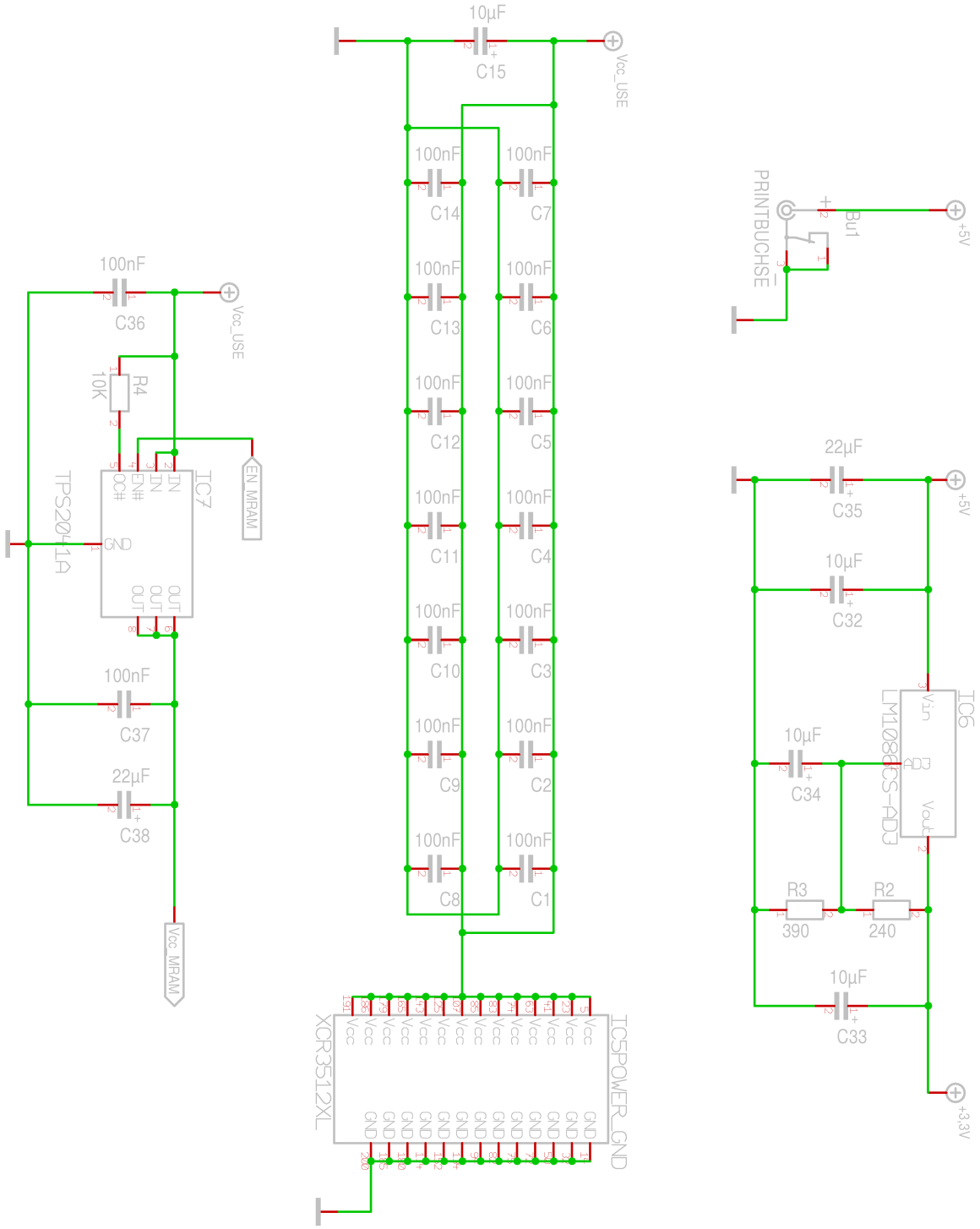



Figure 8.18: Theia Schematic - Page 3 - Power Supply - A voltage regulator, a capacitor decoupling network and an external power-switch for the MRAM are shown.

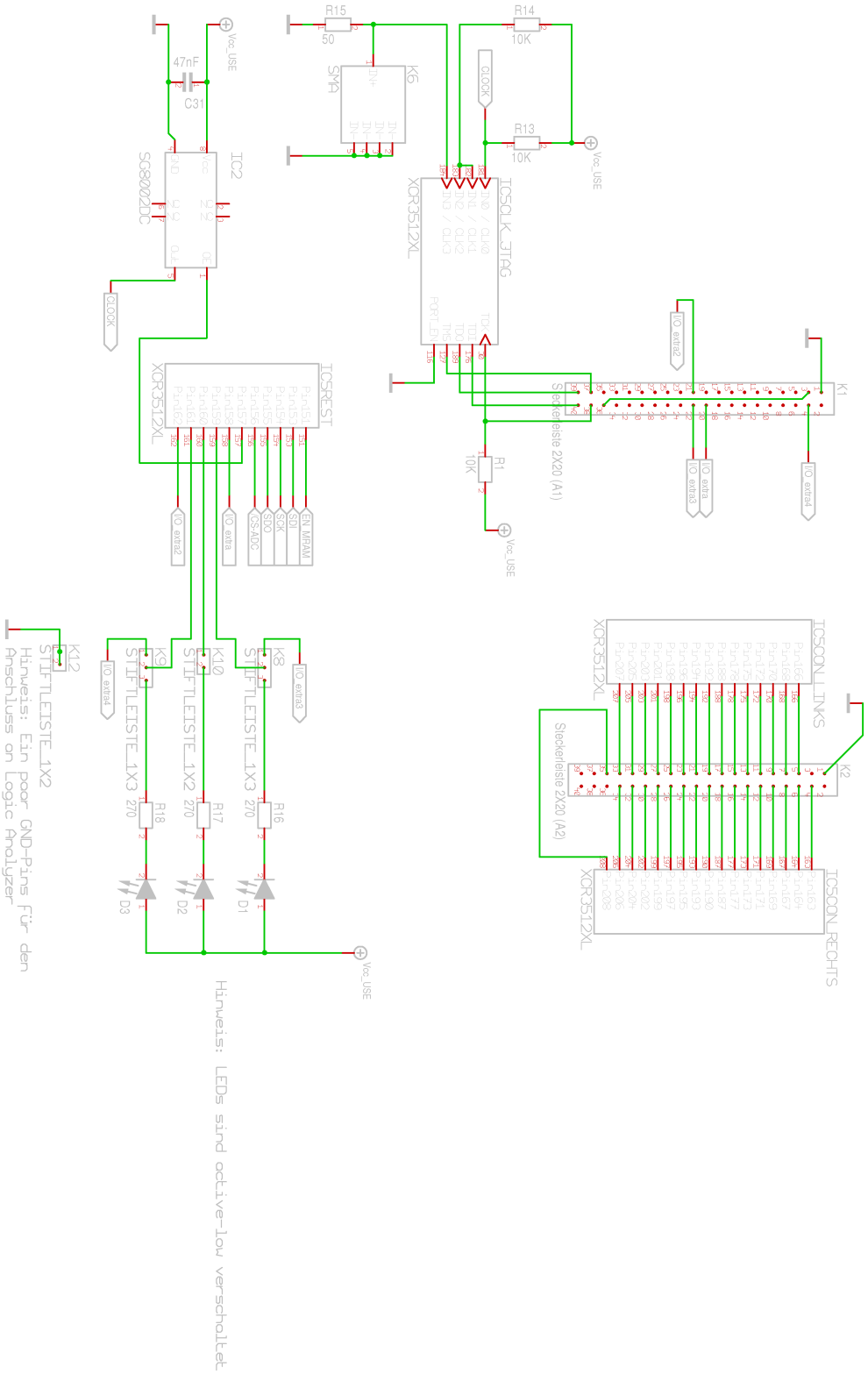



Figure 8.19: Theia Schematic - Page 4 - Miscellaneous: The configuration circuitry is shown (JTAG). Besides the configuration circuitry the clock sources (on-board chip and coax plug) and status LEDs can be seen.

### 8.2.2 Layout

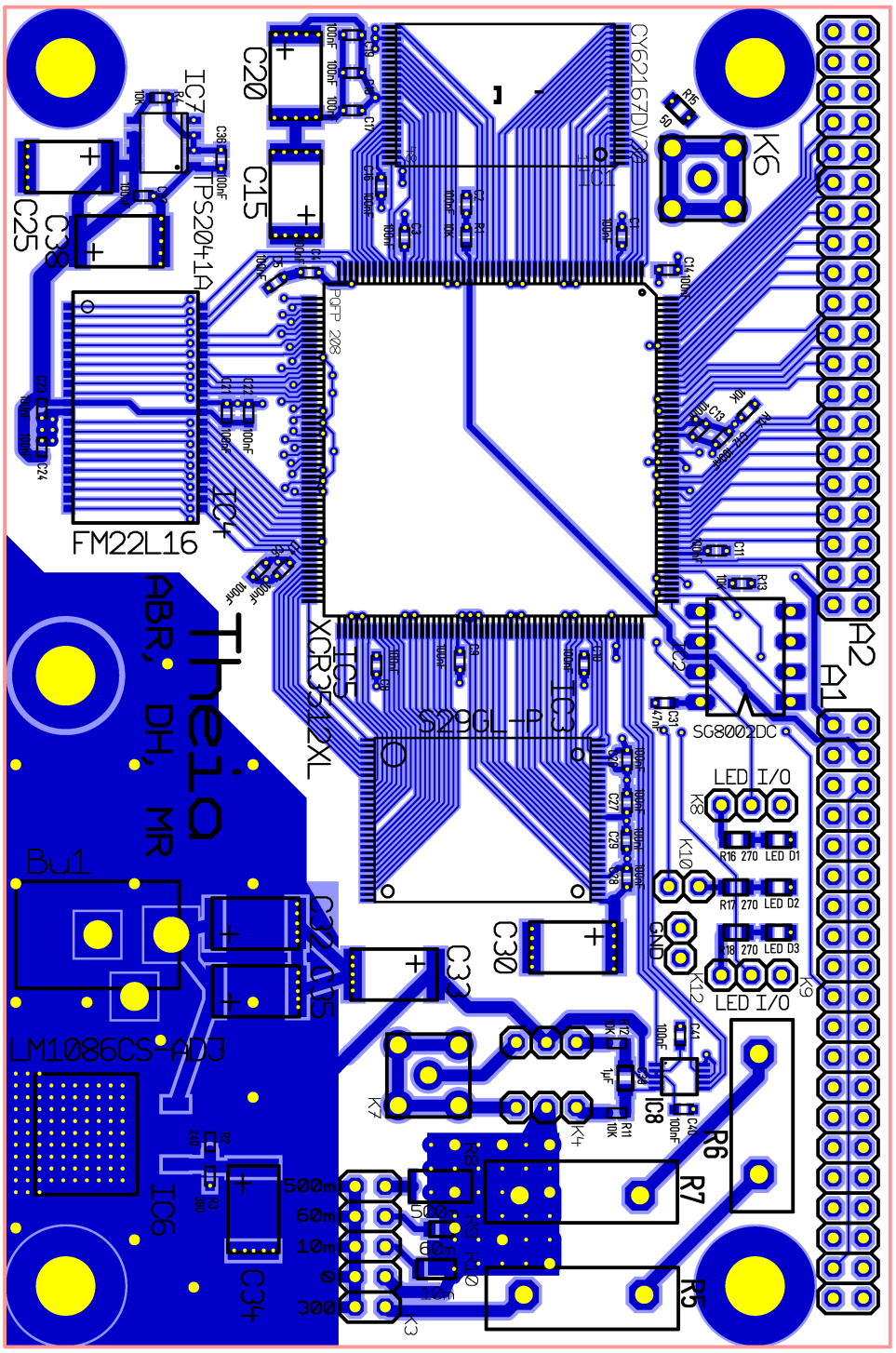



Figure 8.20: Theia Printed Circuit Board - Top Layer

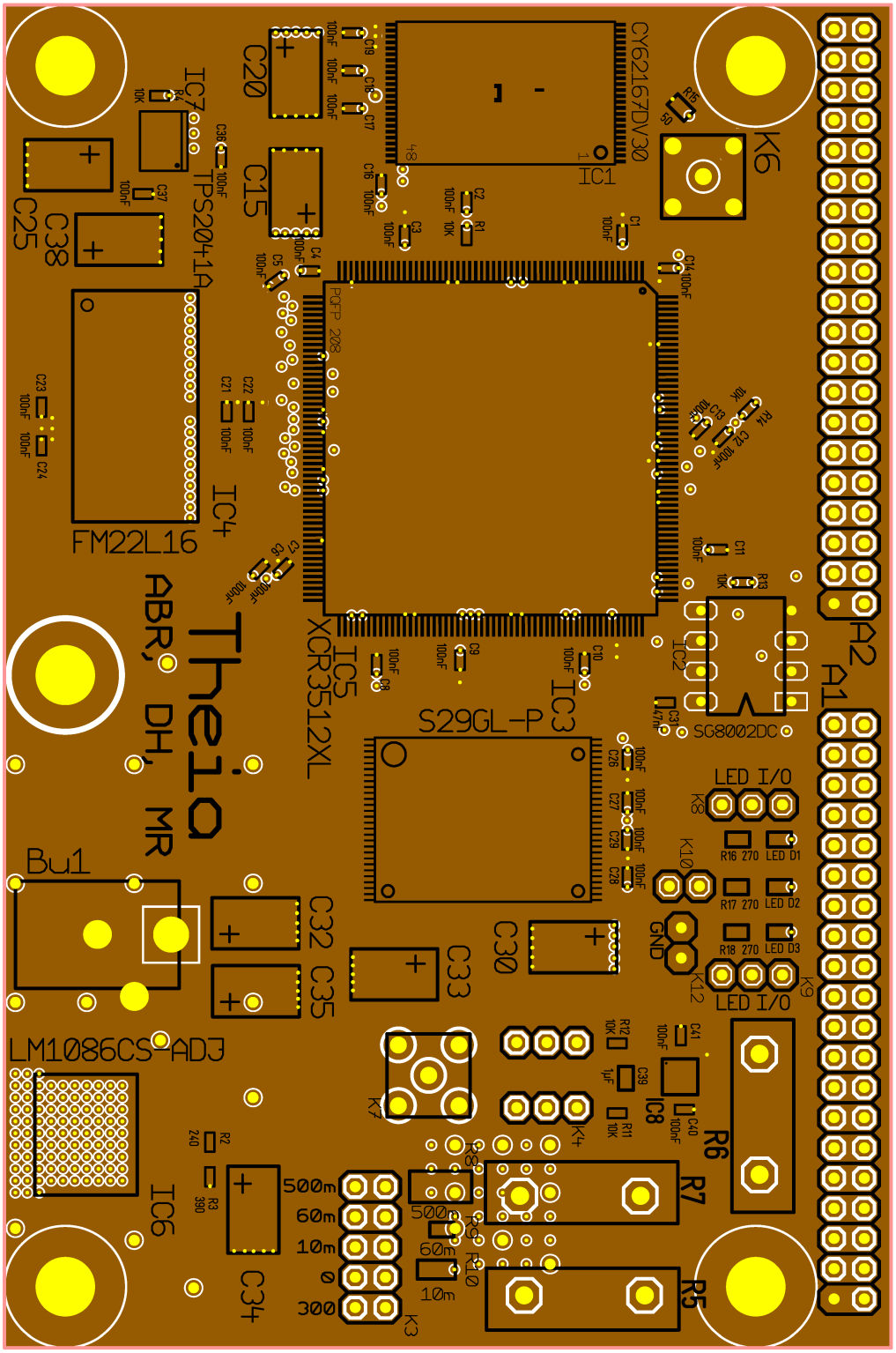



Figure 8.21: Theia Printed Circuit Board - Ground Layer

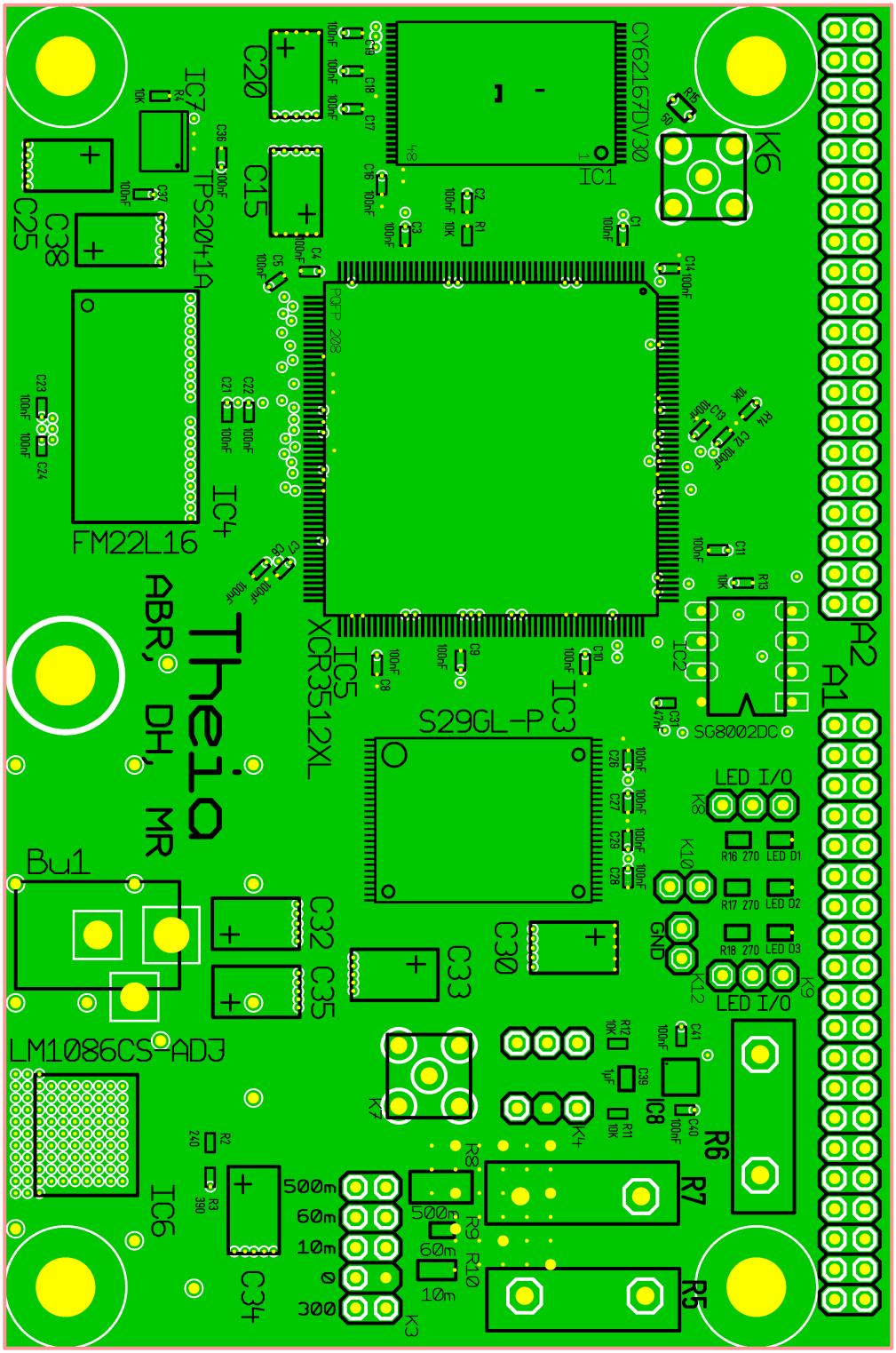



Figure 8.22: Theia Printed Circuit Board - VCC Layer

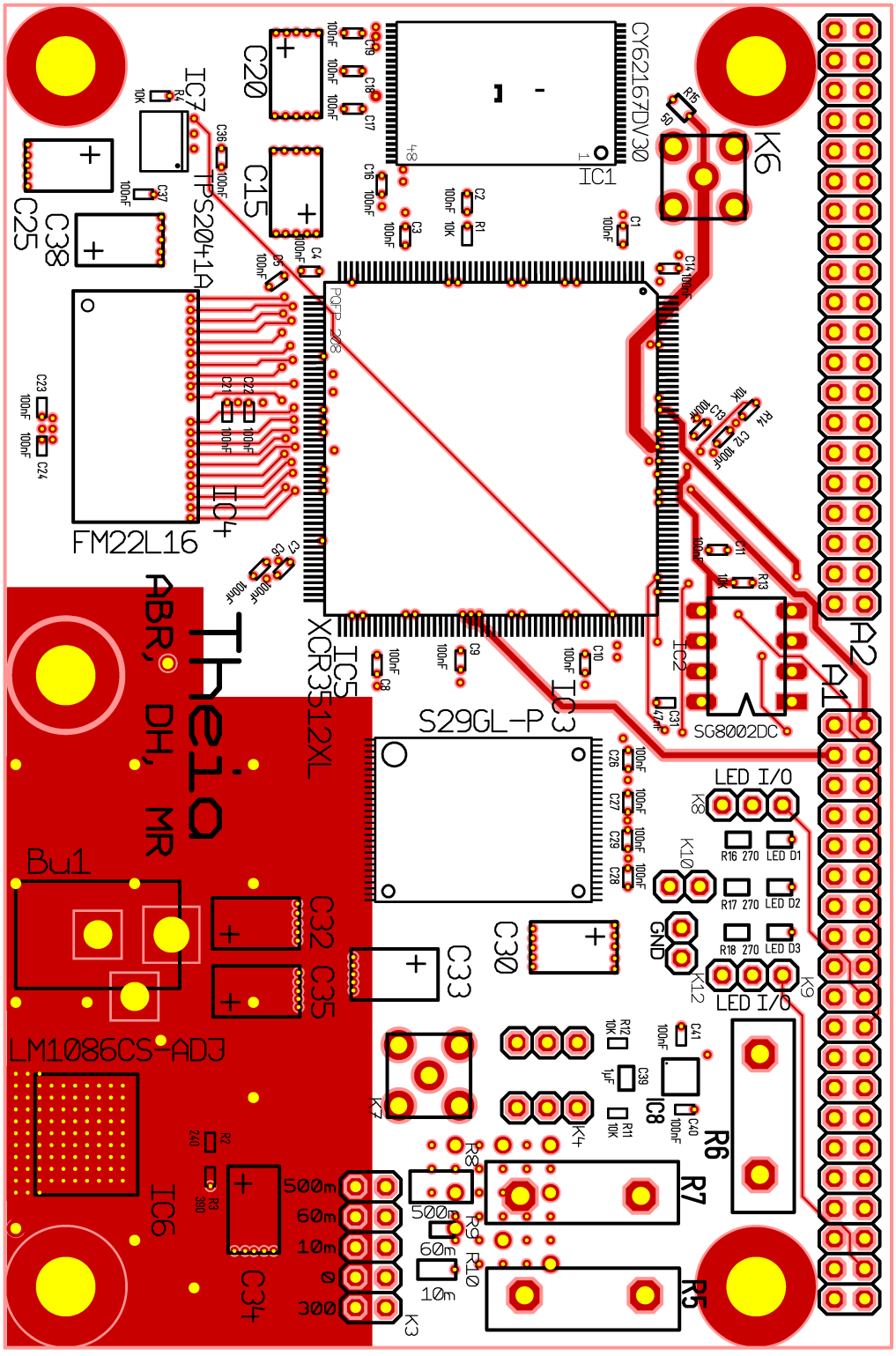



Figure 8.23: Theia Printed Circuit Board - Bottom Layer

| Sensor Node Type / Year | Mica 2001 | Telos 2004 | SunSPOT 2007-2009 | Hyperion 2007-2009 (this thesis) |
|---|---|---|---|---|
| Microcontroller | | | | |
| Type | ATmega128 | TI-MSP430 | ARM-920 | Configurable |
| Frequency [MHz] | 8 | 16 | 180 | Variable |
| Word Size [Bit] | 8 | 16 | 32 | Variable |
| Program Memory [kb] | 128 | 48 | 128 | - |
| RAM [kb] | 4 | 10 | 16 | 43 |
| Active Power [mW] | 8 | 3 | 216 (total node) | 58-144 quiescent + up to 300 dynamic |
| Sleep Power [µW] | 75 | 15 | 89 (total node) | Not applicable |
| Wake-Up Time [µs] | 180 | 6 | N/A | Not applicable |
| (On-board) Storage | | | | |
| Chip | AT45DB041B | ST M25P80 | S71PL032J40 | 2xCY62167DV30 (see also Theia) |
| Size [kb] | 512 Flash | 1024 Flash | 4096 Flash / 512 PSRAM | 3906 SRAM |
| Active Power [mW] | 45 | 45 | N/A | 120 |
| Sleep Power [µW] | 30 | 150 | N/A | 16 |
| Radio | | | | |
| Chip | TR1000 | CC2420 | | CC1100 (plugable) |
| Data rate [kbps] | 40 | 250 | | 1.2-250 |
| Modulation Type | ASK | O-QPSK | | OOK, ASK, 2-FSK, GFSK, MSK |
| Receive Power [mW] | 12 | 38 | | 54 |
| Transmit Power @ 0 dBm [mW] | 36 | 35 | | 53 |
| max. Range [m] | 60 | 100 | | 200 |
| Expansion and Sensors | | | | |
| Expansion [#pins] | 51 | 16 | 30 | 51=16+15+8+12 |
| Integrated Sensors | no | yes | no | no |
| Software | | | | |
| Typ. Operating System | TinyOS | TinyOS | Squawk Java Virtual Machine | SenOS |

Figure 8.24: Sensor node platforms ([61, 106], Sun Spot Developer's Guide/Owner's Manual, eSPOT Rev 6.2 Schematics, Datasheets: AT91RM9200, S71PL-J MCPs, CY62167DV30, CC110, XC3S400, M25P80, Crossbow: TelosB and Mica2)

# Bibliography


[1] Crossbow - commercial sensor nodes. `http://www.xbow.com`.

[2] The network simulator ns-2. `http://nsnam.isi.edu/nsnam/`.

[3] Tinyos - hard- and software specifications. `http://tinyos.net/`.

[4] Anastassia Ailamaki, Christos Faloutos, Paul S. Fischbeck, Mitchell J. Small, and Jeanne VanBriesen. An environmental sensor network to determine drinking water quality and security. *SIGMOD Rec.*, 32(4):47–52, 2003.

[5] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, Aug 2002.

[6] Deborah Estrin Alberto Cerpa, Naim Busek. Scale: A tool for simple connectivity assessment in lossy environments. *Center for Embedded Networked Sensing*, 2003.

[7] D.H. Albonesi, R. Balasubramonian, S.G. Dropsbo, S. Dwarkadas, F.G. Friedman, M.C. Huang, V. Kursun, G. Magklis, M.L. Scott, G. Semeraro, P. Bose, A. Buyuktosunoglu, P.W. Cook, and S.E. Schuster. Dynamically tuning processor resources with adaptive processing. *Computer*, 36(12):49–58, Dec. 2003.

[8] N. Amitay. Modeling and computer simulation of wave propagation in lineal line-of-sight microcells. *Vehicular Technology, IEEE Transactions on*, 41(4):337–342, Nov 1992.

[9] Robert Moffatt J. D. Joannopoulos Peter Fisher AndrÃ© Kurs, Aristeidis Karalis and Marin Soljacic. Wireless power transfer via strongly coupled magnetic resonances. *Science 317 (5834) - Massachusetts Institute of Technology, Cambridge*, 2007.

[10] ATMEL. *8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash - ATmega128,ATmega128L*. Rev. 2467-AVR-07/09.

[11] ATMEL. *8-bit Instruction Set*. Rev. 0856H-AVR-07/09.

[12] A. Barton-Sweeney, D. Lymberopoulos, and A. Sawides. Sensor localization and camera calibration in distributed camera sensor networks. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–10, Oct. 2006.

[13] Athanassios Boulis. Castalia: revealing pitfalls in designing distributed algorithms in wsn. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 407–408, New York, NY, USA, 2007. ACM.

[14] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 22–31, New York, NY, USA, 2002. ACM.

[15] S. Calderara, R. Cucchiara, and A. Prati. A distributed outdoor video surveillance system for detection of abnormal people trajectories. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 364–371, Sept. 2007.

[16] University Of California, Shad Roundy, Paul K. Wright, and Jan Rabaey. A study of low level vibrations as a power source for wireless sensor nodes shad roundy * , paul k. wright, jan rabaey. *Computer Communications*, 26:1131–1144, 2003.

[17] Taieb Znati Cauligi S. Raghavendra, Krishna M. Sivalingam, editor. *Wireless Sensor Networks*. Kluwer Academic, 2004.

[18] Liu Gao McIlwrath Kevin Zhang Xiao Feng Huggins Robert A. Cui Yi Chan Candace K., Peng Hailin. High-performance lithium battery anodes using silicon nanowires. *Nature Nanotechnology*, Nature Publishing Group:31 – 35, 2008.

[19] Chandra-Sekaran, A., Nwokafor, A., Shammas, L., Kunze, C., Mueller-Glaser, and K.D. A disaster aid sensor network using zigbee for patient localization and air temperature monitoring. *International Journal on Advances in Networks and services*, January 2009.

[20] A.-K. Chandra-Sekaran, A. Nwokafor, P. Johansson, K.D. Mueller-Glaser, and I. Krueger. Zigbee sensor network for patient localization and air temperature monitoring during emergency response to crisis. In *Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on*, pages 233–238, Aug. 2008.

[21] Ashok-Kumar Chandra-Sekaran, Prabu Dheenathayalan, Pascal Weisser, Christophe Kunze, and Wilhelm Stork. Empirical analysis and ranging using environment and mobility adaptive rssi filter for patient localization during disaster management. In *ICNS '09: Proceedings of the 2009 Fifth International Conference on Networking and Services*, pages 276–281, Washington, DC, USA, 2009. IEEE Computer Society.

[22] Ashok-Kumar Chandra-Sekaran, Gerd Flaig, Christophe Kunze, Wilhelm Stork, and Klaus D. Müller-Glaser. Efficient resource estimation during mass casualty emergency response based on a location aware disaster aid network. In *EWSN*, pages 205–220, 2008.

[23] Ashok-Kumar Chandra-Sekaran, Gunnar Stefansson, Christophe Kunze, Klaus D. Muller-Glaser, and Pascal Weisser. A range-based monte carlo patient localization during emergency response to crisis. In *AICT '09: Proceedings of the 2009 Fifth Advanced International Conference on Telecommunications*, pages 21–26, Washington, DC, USA, 2009. IEEE Computer Society.

[24] Ashok-Kumar Chandra-Sekaran, Pascal Weisser, Klaus D. Muller-Glaser, and Christophe Kunze. A comparison of bayesian filter based approaches for patient localization during emergency response to crisis. In *SENSORCOMM '09: Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications*, pages 636–642, Washington, DC, USA, 2009. IEEE Computer Society.

[25] Anantha Chandrakasan. *Leakage in Nanometer CMOS Technologies (Integrated Circuits and Systems)*. Springer, 2005.

[26] Naehyuck Chang and Kwanho Kim. Real-time per-cycle energy consumption measurement of digital systems. *Electronics Letters*, 36(13):1169–1171, Jun 2000.

[27] Naehyuck Chang, Kwanho Kim, and Hyung Gyu Lee. Cycle-accurate energy consumption measurement and analysis: Case study of arm7tdmi. In *Low Power Electronics and Design, 2000. ISLPED '00. Proceedings of the 2000 International Symposium on*, pages 185–190, 2000.

[28] Naehyuck Chang, Kwanho Kim, and Hyung Gyu Lee. Cycle-accurate energy measurement and characterization with a case study of the arm7tdmi. *IEEE Trans. Very Large Scale Integr. Syst.*, 10(2):146–154, 2002.

[29] Chih chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. Sos: A dynamic operating system for sensor networks. In *Proceedings of the Third International Conference on Mobile Systems, Applications, And Services (Mobisys*. ACM Press, 2005.

[30] Cassandras Christos and Lafortune Stephane. *Introduction to Discrete Event Systems*. Springer, 1999.

[31] Linear Technology Corporation. *Dual DC/DC Converter with USB Power Manager and Li-Ion Battery Charger*. LTC3455/LTC3455-1.

[32] David E. Culler. Wireless embedded systems and networking foundations of ip-based ubiquitous sensor networks - wsn technology and hardware architectures. *University of California, Berkeley*, 2007.

[33] M. Daniels, K. Muldawer, J. Schlessman, B. Ozer, and W. Wolf. Real-time human motion detection with distributed smart cameras. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 187–194, Sept. 2007.

[34] Chip Elliott David Kotz, Calvin Newport. The mistaken axioms of wireless-network research. Darthmouth College Computer Science Technical Report TR2003-467, 2003.

[35] Alec Woo David Culler Bhaskar Krishnamachari Stephen Wicker Deepak Ganesan, Deborah Estrin. Complex behavior at scale: An experimental study of low-power wireless sensor networks. *Center for Embedded Networked Sensing*, 2002.

[36] F. Dias, F. Berry, J. Serot, and F. Marmoiton. Hardware, design and implementation issues on a fpga-based smart camera. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 20–26, Sept. 2007.

[37] I. Diaz, M. Heijligers, R. Kleihorst, and A. Danilin. An embedded low power high efficient object tracker for surveillance systems. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 372–378, Sept. 2007.

[38] Timothy Armstrong Joerg Henkel Dominic Hillenbrand, Michael Mende. Hyperion: A sensor node test bed for (high-speed) power measurements. In *Design, Automation and Test in Europe -DATE*. University Booth, 2007.

[39] Adam Dunkels, Niclas Finne, Joakim Eriksson, and Thiemo Voigt. Run-time dynamic linking for reprogramming wireless sensor networks. In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, November 2006.

[40] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *LCN '04: Proceedings of the 29th Annual*

*IEEE International Conference on Local Computer Networks*, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[41] Bernie Yip Amarjeet Singh Winston Wu Dustin McIntire, Kei Ho and William J. Kaiser. The low power energy aware processing (leap) embedded networked sensor system. Technical report, November 18 2005.

[42] Prabal Dutta, Mark Feldmeier, Joseph Paradiso, and David Culler. Energy metering for free: Augmenting switching regulators for real-time monitoring. In *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, pages 283–294, Washington, DC, USA, 2008. IEEE Computer Society.

[43] A. El-Hoiydi, C. Arm, R. Caseiro, S. Cserveny, J.-D. Decotignie, C. Enz, F. Giroud, S. Gyger, E. Leroux, T. Melly, V. Peiris, F. Pengg, P.-D. Pfister, N. Raemy, A. Ribordy, D. Ruffieux, and P. Volet. The ultra low-power wisenet system. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 971–976, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[44] A. El-Hoiydi, C. Arm, R. Caseiro, S. Cserveny, J.-D. Decotignie, C. Enz, F. Giroud, S. Gyger, E. Leroux, T. Melly, V. Peiris, F. Pengg, P.-D. Pfister, N. Raemy, A. Ribordy, D. Ruffieux, and P. Volet. The ultra low-power wisenet system. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 971–976, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[45] A. El-Hoiydi and J.-D. Decotignie. Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In *ISCC '04: Proceedings of the Ninth International Symposium on Computers and Communications 2004 Volume 2 (ISCC''04)*, pages 244–251, Washington, DC, USA, 2004. IEEE Computer Society.

[46] Environmentally Scavenged Energy, Shad Roundy, Brian P. Otis, Yuen hui Chee, Jan M. Rabaey, and Paul Wright. A 1.9ghz rf transmit beacon using. In *Dig. IEEE Int. Symposium on Low Power Elec. and Devices, Seoul, Korea*, 2003.

[47] Leonidas Guibas Feng Zhao. *Wireless Sensor Networks An Information Processing Approach*. Morgan Kaufmann, 2004.

[48] Rodrigo Fonseca, Prabal Dutta, Philip Levis, and Ion Stoica. Quanto: Tracking energy in networked embedded systems. In *OSDI*, pages 323–338, 2008.

[49] T. Fugen, J. Maurer, T. Kayser, and W. Wiesbeck. Verification of 3d ray-tracing with non-directional and directional measurements in urban macrocellular environments. In *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd*, volume 6, pages 2661–2665, May 2006.

[50] T. Fugen, J. Maurer, and W. Wiesbeck. Cluster characterization in urban macrocellular environments with ray-tracing. In *Vehicular Technology Conference, 2005. VTC-2005-Fall. 2005 IEEE 62nd*, volume 3, pages 1723–1727, Sept., 2005.

[51] David Gay, Matt Welsh, Philip Levis, Eric Brewer, Robert von Behren, and David Culler. The nesc language: A holistic approach to networked embedded systems. In *In Proceedings of Programming Language Design and Implementation (PLDI*, pages 1–11, 2003.

[52] Andrea Goldsmith. *Wireless Communications*. Cambridge University Pres, 2005.

[53] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. A dynamic operating system for sensor nodes. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 163–176, New York, NY, USA, 2005. ACM.

[54] V. Handziski, J. Polastre, J.-H. Hauer, C. Sharp, A. Wolisz, and D. Culler. Flexible hardware abstraction for wireless sensor networks. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 145–157, Jan.-2 Feb. 2005.

[55] M. Hata. Empirical formula for propagation loss in land mobile radio services. In *IEEE Transactions Vehic. Technol.*, volume 20, pages 318–325, August 1980.

[56] Tian He, Sudha Krishnamurthy, Liqian Luo, Ting Yan, Lin Gu, Radu Stoleru, Gang Zhou, Qing Cao, Pascal Vicaire, John A. Stankovic, Tarek F. Abdelzaher, Jonathan Hui, and Bruce Krogh. Vigilnet: An integrated sensor network system for energy-efficient surveillance. *ACM Trans. Sen. Netw.*, 2(1):1–38, 2006.

[57] Tian He, Sudha Krishnamurthy, John A. Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh. Energy-efficient surveillance system using wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 270–283, New York, NY, USA, 2004. ACM.

[58] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM.

[59] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10 pp. vol.2–, Jan. 2000.

[60] S. Hengstler, H. Aghajan, and A. Goldsmith. Application-oriented design of smart camera networks. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 12–19, Sept. 2007.

[61] Jason L. Hill and David E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[62] Jason Lester Hill. *System Architecture for Wireless Sensor Networks*. 2003.

[63] Dominic Hillenbrand. Exploiting heterogeneous (die-stacked) memories in future cmps to reduce power consumption. In *ACACES - Advanced Computer Architecture and Compilation for Embedded Systems, Terrassa, Spain*, pages 155–158, 2009.

[64] Dominic Hillenbrand and Jörg Henkel. Block cache for embedded systems. In *ASP-DAC '08: Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, pages 322–327, Los Alamitos, CA, USA, 2008. IEEE Computer Society Press.

[65] Andreas Willig Holger Karl. *Protocols and Architectures for Wireless Sensor Networks*. Wiley, 2005.

[66] Texas Instruments. *MSP430x1xx User's Guide Rev. F.* 2006.

[67] Texas Instruments. Cc1100 low-power sub-1 ghz rf transceiver. *Chipon Datasheet*, 2009.

[68] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[69] S. Madden A. Woo J. Polastre C. Whitehouse R. Szewczyk C. Sharp D. Gay M. Welsh D. Culler J. Hill, P. Levis and E. Brewer. Tinyos: An operating system for sensor networks. 2003.

[70] Qiangfeng Jiang and D. Manivannan. Routing protocols for sensor networks. In *Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE*, pages 93–98, Jan. 2004.

[71] David A. Patterson John L. Hennessy. *Computer Architecture A Quantitative Approach*. Morgan Kaufmann, 2006.

[72] Cory Sharp David Culler Joseph Polastre, Robert Szewczyk. The mote revolution: Low power wireless sensor network devices. In *The Mote Revolution: Low Power Wireless Sensor Network Devices*. Hot Chips 16: A Symposium on High Performance Chips, 2004.

[73] Edgar H. Callaway Jr. *Wireless Sensor Networks: Architectures and Protocols*. Auerbach Publications, 2003.

[74] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for "smart dust". In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278, New York, NY, USA, 1999. ACM.

[75] Sung-Mo Steve Kang and Yusuf Leblebici. *CMOS Digital Integrated Circuits Analysis and Design*. McGraw-Hill, 2002.

[76] Aristeidis Karalis, J.D. Joannopoulos, and Marin Soljacic. Efficient wireless non-radiative mid-range energy transfer. *Annals of Physics*, 323(1):34 – 48, 2008. January Special Issue 2008.

[77] Brad Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM.

[78] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Wirel. Netw.*, 6(4):307–321, 2000.

[79] David Kotz, Calvin Newport, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliott. Experimental evaluation of wireless simulation assumptions. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 78–82, New York, NY, USA, 2004. ACM.

[80] Bhaskar Krishnamachari. *Networking Wireless Sensors*. 2006.

[81] O. Landsiedel, K. Wehrle, and S. Gotz. Accurate prediction of power consumption in sensor networks. In *EmNets '05: Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, pages 37–44, Washington, DC, USA, 2005. IEEE Computer Society.

[82] H. Q. Le, W. J. Starke, J. S. Fields, F. P. O'Connell, D. Q. Nguyen, B. J. Ronchetti, W. M. Sauer, E. M. Schwarz, and M. T. Vaden. Ibm power6 microarchitecture. *IBM J. Res. Dev.*, 51(6):639–662, 2007.

[83] Hyung Gyu Lee, Kyungsoo Lee, Yongseok Choi, and Naehyuck Chang. Cycle-accurate energy measurement and characterization of fpgas. *Analog Integr. Circuits Signal Process.*, 42(3):239–251, 2005.

[84] Martin Leopold. Hogthrobvo users manual. 2007.

[85] Martin Leopold. *Sensor Network Motes: Portability and Performance*. 2007.

[86] Martin Leopold, Marcus Chang, and Philippe Bonnet. Characterizing mote performance: A vector-based methodology. In *EWSN*, pages 321–336, 2008.

[87] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. pages 115–148. 2005.

[88] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM.

[89] Yingmin Li, K. Skadron, D. Brooks, and Zhigang Hu. Performance, energy, and thermal considerations for smt and cmp architectures. In *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pages 71–82, Feb. 2005.

[90] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM.

[91] D. Maniezzo, K. Yao, and G. Mazzini. Energetic trade-off between computing and communication resource in multimedia surveillance sensor network. In *Mobile and Wireless Communications Network, 2002. 4th International Workshop on*, pages 373–376, 2002.

[92] A. Manjeshwar and D.P. Agrawal. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pages 2009–2015, Apr 2001.

[93] Krishnamachari Bhaskar Marco Zuniga Zamalloa. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Trans. Sen. Netw.*, 3(2):7, 2007.

[94] Klaus S. Madsen Philippe Bonnet Marcus Chang, Cecile Cornou. Lessons from the hogthrob deployments. In *Second International Workshop on Wireless Sensor Network Deployments*, 2008.

[95] Dustin McIntire, Kei Ho, Bernie Yip, Amarjeet Singh, Winston Wu, and William J. Kaiser. The low power energy aware processing (leap)embedded networked sensor system. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pages 449–457, New York, NY, USA, 2006. ACM.

[96] Imad Mahgoub Mohammad Ilyas, editor. *Handbook of Sensor Networks Compact Wireless and Wired Sensing Systems*. CRC, 2004.

[97] D. Estrin M. Hansen T. Harmon E. Kohler M. Srivastava N. Ramanathan, T. Schoellhammer. The final frontier: Embedding networked sensors in the soil. Technical report, November 2006.

[98] H. Nakagawa, K. Ishida, T. Ohta, and Y. Kakuda. Goli: Greedy on-demand routing scheme using location information for mobile ad hoc networks. In *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on*, pages 1–1, July 2006.

[99] George C. Necula, Scott McPeak, Shree Prakash Rahul, and Westley Weimer. Cil: Intermediate language and tools for analysis and transformation of c programs. In *CC '02: Proceedings of the 11th International Conference on Compiler Construction*, pages 213–228, London, UK, 2002. Springer-Verlag.

[100] Sanjay Jha Nirupama Bulusu. *Wireless Sensor Networks A Systems Perspective*. Artech House Publishers, 2005.

[101] University of Cincinnati E. Belding-Royer C. Perkins Nokia Research Center, University of California Santa Barbara. Ad hoc on-demand distance vector (aodv) routing. *(Experimental) Request for Comments: 3561*, 2003.

[102] Debashis Panigrahi, Sujit Dey, Ramesh Rao, Kanishka Lahiri, Carla Chiasserini, and Anand Raghunathan. Battery life estimation of mobile embedded systems. In *VLSID '01: Proceedings of the The 14th International Conference on VLSI Design (VLSID '01)*, page 57, Washington, DC, USA, 2001. IEEE Computer Society.

[103] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. *Mobile Computing Systems and Applications, IEEE Workshop on*, 0:90, 1999.

[104] Hai N. Pham, Dimosthenis Pediaditakis, and Athanassios Boulis. From simulation to real deployments in wsn and back. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–6, June 2007.

[105] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM.

[106] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[107] J.M. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, and T. Tuan. Picoradics for wireless sensor networks: the next challenge in ultra-low-power design. In *Solid-State*

*Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International*, volume 2, pages 156–445, 2002.

[108] J.M. Rabaey, M.J. Ammer, Jr. da Silva, J.L., D. Patel, and S. Roundy. Picoradio supports ad hoc ultra-low power wireless networking. *Computer*, 33(7):42–48, Jul 2000.

[109] Vijay Raghunathan and Pai H. Chou. Design and power management of energy harvesting embedded systems. In *ISLPED '06: Proceedings of the 2006 international symposium on Low power electronics and design*, pages 369–374, New York, NY, USA, 2006. ACM.

[110] R. Rao, S. Vrudhula, and D.N. Rakhmatov. Battery modeling for energy aware system design. *Computer*, 36(12):77–87, Dec. 2003.

[111] Theodore S. Rappaport. *Wireless Communications: Principles and Practice (2nd Edition)*. Prentice Hall, 2002.

[112] Qingchun Ren and Qilian Liang. An energy-efficient mac protocol for wireless sensor networks. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 1, pages 5 pp.–, Nov.-2 Dec. 2005.

[113] Josep Rius, Alejandro Peidro, Salvador Manich, and Rosa Rodriguez-Sánchez. Power and energy consumption of cmos circuits: Measurement methods and experimental results. In *PATMOS*, pages 80–89, 2003.

[114] Shadrach Joseph Roundy. *PhD Thesis - Energy Scavenging for Wireless Sensor Nodes with a Focus on Vibration to Electricity Conversion*. 2003.

[115] Paolo Santi. *Topology Control in Wireless Ad Hoc and Sensor Networks*. Wiley, 2005.

[116] M. Seltzer, D. Krinsky, K. Smith, and Xiaolan Zhang. The case for application-specific benchmarking. In *Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on*, pages 102–107, 1999.

[117] Bosch Sensortec. *BMA150 Digital, triaxial acceleration sensor*. 2008.

[118] Bosch Sensortec. *BMP085 Digital Pressure Sensor*. 2008.

[119] Paul Kenneth Wright Shad Roundy, Jan M. Rabaey. *Energy Scavenging for Wireless Sensor Networks: with Special Focus on Vibrations*. 2003.

[120] Li Shang, Alireza S. Kaviani, and Kusuma Bathala. Dynamic power consumption in virtex-ii fpga family. In *FPGA '02: Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pages 157–164, New York, NY, USA, 2002. ACM.

[121] Xiaolei Shi and Guido Stromberg. Syncwuf: An ultra low-power mac protocol for wireless sensor networks. *IEEE Transactions on Mobile Computing*, 6(1):115–125, 2007.

[122] Dongkun Shin, Hojun Shim, Yongsoo Joo, Han-Saem Yun, Jihong Kim, and Naehyuck Chang. Energy-monitoring tool for low-power embedded programs. *Design and Test of Computers, IEEE*, 19(4):7–17, Jul/Aug 2002.

[123] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications.

In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 188–200, New York, NY, USA, 2004. ACM.

[124] IEEE Computer Society. 802.15.4a part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans). *Std 802.15.4aTM-2007*, 2007.

[125] Eunseok Song, Young-Kil Park, Soon Kwon, and Soo-Ik Chae. A cycle-accurate energy estimator for cmos digital circuits. In *PATMOS*, pages 159–168, 2004.

[126] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. Understanding the causes of packet delivery success and failure in dense wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 419–420, New York, NY, USA, 2006. ACM.

[127] Margaret Martonosi Stefanos Kaxiras. *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool Publishers, 2008.

[128] M. Stordeur and I. Stark. Low power thermoelectric generator-self-sufficient energy supply for micro systems. In *Thermoelectrics, 1997. Proceedings ICT '97. XVI International Conference on*, pages 575–577, Aug 1997.

[129] Inc. Sun Microsystems. Small programmable object technology (sun spot) developer's guide. In *Sun Labs*, 2008.

[130] Inc. Sun Microsystems. Suntm spot owner's manual blue release 4.0. In *Sun Labs*, 2008.

[131] E. Ohmori T. Okumura and K. Fukuda. Field strength and its variability in vhf and uhf land mobile service. In *Review Electrical Communication Laboratory*, volume 16, pages 825–873, Sept.-Oct. 1968.

[132] Andrew Tannenbaum. *Modern Operating Systems Second Edition*. Prentice Hall, 2001.

[133] Jie Tao, Dominic Hillenbrand, and Holger Marten. Instruction hints for super efficient data caches. In *ICCS (2)*, pages 677–685, 2009.

[134] Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: scalable sensor network simulation with precise timing. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 67, Piscataway, NJ, USA, 2005. IEEE Press.

[135] Ben L. Titzer and Jens Palsberg. Nonintrusive precision instrumentation of microcontroller software. In *LCTES '05: Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, pages 59–68, New York, NY, USA, 2005. ACM.

[136] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 121–132, Jan.-2 Feb. 2005.

[137] W.H.W. Tuttlebee. Software-defined radio: facets of a developing technology. *Personal Communications, IEEE*, 6(2):38–44, Apr 1999.

[138] John P. Uyemura. *CMOS Logic Circuit Design*. Springer, 1999.

[139] Andras Varga. The omnet++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference*, pages 319–324, Prague, Czech Republic, June 2001. SCS – European Publishing House.

[140] Kashif Virk, Jan Madsen, Andreas Vad Lorentzen, Martin Leopold, and Phillipe Bonnet. Design of a development platform for hw/sw codesign of wireless integrated sensor nodes. In *DSD '05: Proceedings of the 8th Euromicro Conference on Digital System Design*, pages 254–260, Washington, DC, USA, 2005. IEEE Computer Society.

[141] A. Wang and A. Chandrakasan. Energy-efficient dsps for wireless sensor networks. *Signal Processing Magazine, IEEE*, 19(4):68–78, Jul 2002.

[142] Tim Wark, Peter Corke, Pavan Sikka, Lasse Klingbeil, Ying Guo, Chris Crossman, Phil Valencia, Dave Swain, and Greg Bishop-Hurley. Transforming agriculture through pervasive wireless sensor networks. *IEEE Pervasive Computing*, 6(2):50–57, 2007.

[143] B. Warneke, M. Last, B. Liebowitz, and K.S.J. Pister. Smart dust: communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, Jan 2001.

[144] John G. Webster. *Electrical Measurement, Signal Processing, and Displays*. CRC, 2003.

[145] Matt Welsh. Experiences with sensor networks for volcano monitoring, 2006.

[146] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 108–120, Jan.-2 Feb. 2005.

[147] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.

[148] A. Wheeler. Commercial applications of wireless sensor networks using zigbee. *Communications Magazine, IEEE*, 45(4):70–77, April 2007.

[149] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, New York, NY, USA, 2004. ACM.

[150] Yingqi Xu, Wang-Chien Lee, Jianliang Xu, and G. Mitchell. Psgr: priority-based stateless geo-routing in wireless sensor networks. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 8 pp.–680, Nov. 2005.

[151] Zongkai Yang, Qifei Zhang, Xu Du, and Linfeng Yuan. Location-based adaptive ad hoc routing (laar). In *Communications and Information Technology, 2005. ISCIT 2005. IEEE International Symposium on*, volume 2, pages 1013–1017, Oct. 2005.

[152] N.H. Zamora and R. Marculescu. Coordinated distributed power management with video sensor networks: Analysis, simulation, and prototyping. In *Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on*, pages 4–11, Sept. 2007.

[153] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13, New York, NY, USA, 2003. ACM.

[154] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138, New York, NY, USA, 2004. ACM.

[155] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 517–526, Oct. 2004.

# Curriculum Vitae

Dominic Hillenbrand
Born 2.2.1979 in Heidelberg, Germany

## CONTACT INFORMATION

Home Address
An der Bleiche 32
67319 Wattenheim, Germany
Tel. (+49) 6356-98 95 11

Departmental Address
Haid-und-Neu-Str. 7 (Building 07.21)
76131 Karlsruhe, Germany

dominic.hillenbrand@gmail.com
http://ces.univ-karlsruhe.de/~hillenbrand/

## UNIVERSITY EDUCATION

| | |
|---|---|
| University of Cambridge, UK | October 2008 – August 2009<br>Researcher at the Computer Laboratory<br>Member of Trinity Hall<br>C3D – Project / EPSRC Grant (UK) |
| University of California,<br>Los Angeles (UCLA) | August – October 2007<br>Research Visit - Professor Srivastava, NESL-Laboratory<br>DFG-sponsored (German Research Council) |
| Technical University of Kaiserslautern | 1998 – August 2004<br>German Diplom (Master's equivalent)<br>Computer science and electrical engineering<br>Grade: very good |
| Auckland University | 2001<br>Visit to Auckland University, New Zealand |

## EMPLOYMENT

| | |
|---|---|
| Technical University of Karlsruhe | Since July 2005<br>Research assistant and PhD candidate – defense January 2010<br>Computer Science Department<br>Chair for Embedded Systems, Prof. Jörg Henkel |
| University of Hannover | February 2005 – July 2005<br>Research assistant at Hannover University<br>Embedded software engineering |

**SCHOOL EDUCATION**

| | |
|---|---|
| Gymnasium Weierhof<br>am Donnersberg | 1989 – June 1998<br>Abitur (=German secondary school diploma qualifying for university admission or matriculation) |
| German School Cape Town<br>(Republic of South Africa) | 1997 |

**PRIZES/AWARDS**

| | |
|---|---|
| 2005 – 2008 | Graduiertenkolleg - DFG-sponsored elite program for forthcoming scientists |
| 2008 | EPSRC grant (Engineering and Physical Sciences Research Council) - UK Government's leading funding agency for research and training in engineering and the physical sciences) |
| 2009 | EU Grant (European Network of Excellence on High Performance and Embedded Architecture and Compilation - HiPEAC) for the ACACES Summer School, Barcelona |

**CONFERENCES/WORKSHOPS/PUBLICATIONS**

| | |
|---|---|
| 2009 | "*Exploiting heterogeneous (die-stacked) memories in future CMPs to reduce power consumption*" – ACACES, Terrassa/Barcelona, Spain |
| 2009 | "*Instruction Hints for Super Efficient Data Caches*", International Conference on Computational Science - Louisiana - USA |
| 2008 | "*Block Cache for Embedded Systems*" - ASP-DAC 2008, Asia and South Pacific Design Automation Conference - Seoul, South Korea |
| 2008 | "*Hyperion – A flexible Sensor Node Testbed*" - GRK-Workshop - Workshop der Graduiertenkollegs, Schloss Dagstuhl, Germany |
| 2007 | "*Hyperion: A Sensor Node Test Bed for (High-Speed) Power Measurements*" – DATE, Design, Automation and Test in Europe, University Booth - Nice, France |
| 2006 | "*Design and evaluation of a hardware architecture for future sensor nodes*" - GRK Workshop - Schloss Dagstuhl, Germany |
| 2005 | "*Analyse von Performanceverbesserungen für ein Baukastensystem zur Konstruktion von Regelkreisen*" - Linux Automation Conference, Hannover, Germany |
| 2004 | "*Spezifikation und Simulation eingebetteter Komponenten*" – Diploma thesis, University of Kaiserslautern, Germany |
| 2001 | "*Real Time Performance Improvements*" - Semester Thesis, University of Kaiserslautern, Germany, 2001 |

**REVIEWS - International Conferences**

I took part in the following reviews:

| | |
|---|---|
| 2008 | SIPS - IEEE Workshop on Signal Processing Systems, Washington, D.C. - U.S.A. |
| 2008 | SASP - IEEE Symposium on Application Specific Processors, San Francisco, California |
| 2008 | SCOPES - International Workshop on Software and Compilers for Embedded Systems, Munich, Germany |
| 2008 | CODES+ISSS, International Conference on Hardware/Software Codesign and System Synthesis - Grenoble, France |
| 2007 | ECRTS, IEEE European Micro Conference on Real-Time Systems, Pisa, Italy |
| 2007 | SAMOS, International Symposium on Systems, Architectures, Modeling and Simulation - Samos, Greece |
| 2006 | RTTS - IEEE Real-Time Systems Symposium, Rio de Janeiro, Brazil |

**TEACHING – Technical University of Karlsruhe**

| | |
|---|---|
| 2006 – 2008 | 7 Master Thesis Supervisions |
| 2005 – 2008 | 9 Semester Thesis Supervisions |
| 2005 – 2008 | Seminar “Embedded Systems in Sensor Networks” |
| 2005 – 2008 | Lecture “Wireless Sensor Networks” as part of the “Low Power Design” lecture series |
| 2005 – 2008 | Lecture “UML – Unified Modeling Language” as part of the “Software-Engineering for Embedded Systems” lecture series |

**Supervised Master Theses / Diplomarbeiten**

| | |
|---|---|
| Sebastian Kobbe - 2008 | *"Konzeption und Evaluierung eines energieeffizienten Sensornetzwerks anhand realer Sensorknoten"* - *"Conception and Evaluation of a Energy Efficient Sensor Network by Using Real Sensor Nodes"* |
| Michael Sätzler - 2007 | *"Konzeption und Entwurf eines Sensorknoten SoC - FPGA Implementierung"* - *"Conception and Design of a Sensor Node SoC - A FPGA - Implementation"* |
| Martin Fischer - 2007 | *"Konzeption eines energieeffizienten Protokolls für ein Kamerasensornetzwerk"* - *"Conception of an Energy Efficient Protocol for a Camera Tracking Sensor Network"* |
| Xu Yongchuny - 2007 | *"Sensornetzwerksimulation und Debugging"* - *"Sensor Network Simulation and Debugging"* |
| Michael Stoll - 2007 | *"Konzeption und Entwurf eines Sensorknoten SoC - Simulation"* - *"Conception and Design of a Sensor Node SoC - Simulation"* |
| Armstrong Timothy and Mende, Michael - 2007 | *"Konzeption und Entwurf einer FPGA basierten Sensorknoten-Entwicklungsplattform"* - *"Conception and Design of a FPGA based Sensor-Node Development Platform"* |

**Supervised Semester Theses / Studienarbeiten**

| | |
|---|---|
| Matthias, Rosenfelder - 2008 | *"Theia: A versatile, heterogeneous and configurable memory tile for low-power sensor-nodes"* |
| David Dueck - 2007 | *"Anbindung eines CMOS Bildsensor an den Hyperion-Sensorknoten"*<br>-<br>*"Integrating a CMOS-image sensor into the Hyperion-Sensor Node"* |
| Sebastian Kobbe - 2007 | *"Entwurf und Implementierung der Hyperion Infrastruktur - Scheduling"*<br>-<br>*"Design and Implementation of the Hyperion Infrastructure - Scheduling"* |
| Xu Yongchuny - 2007 | *"Leistungs- und Energieanalyse in Sensornetzwerken"*<br>-<br>*"Performance and Energy-Analysis in Sensor Networks"* |
| Sebastian Reiter - 2007 | *"Energieeffiziente Codecompression für Sensorknoten"*<br>-<br>*"Energy Efficient Code Compression for Sensor Nodes"* |
| Cheng Weiwei - 2007 | *"Spezifikation und Energieverbrauchsanalyse eines Sensornetzwerkes"*<br>-<br>*"Specification and Energy-Analysis of a Sensor Network"* |
| Sätzler Michael - 2006 | *"Systemsimulator und -profiler für Sensornetzwerkknoten"*<br>-<br>*"System-Simulator and -profiler for Sensor Nodes"* |
| Demel Michael - 2006 | *"Leistungsmessungen an Sensorknoten"*<br>-<br>*"Power measurements of Sensor Nodes"* |