DIPLOMARBEIT

Evaluierung der Reduktion der Leistungsaufnahme durch eine rekonfigurierbare Architektur

ausgeführt zur Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

Univ.Prof. Dr.phil.nat. Christoph Grimm
Mag. Dipl.-Ing. Dr. Johann Glaser

am

Institut für Computertechnik (E384)
der Technischen Universität Wien

durch

Georg Blemenschitz, BSc
Matr.Nr. 0425435
Anton Baumgartner Str. 44/C3/1602
1230 Wien

Wien, am 2. März 2016 _____

Kurzfassung

Durch die weite Verbreitung von Systemen mit Mikroprozessor in den letzten Jahren ist der Bedarf der Reduktion ihrer Leistungsaufnahme gestiegen. Insbesondere Funksensorknoten stellen eine Herausforderung in Bezug auf begrenzte Energieressourcen und lange Lebensdauer dar. Neben der klassischen Realisierung des Funksensorknotens mit einem Mikrocontroller gibt es einen alternativen Ansatz mit autonomen, rekonfigurierbaren Modulen in einem System-on-Chip. Dieser Ansatz wurde jedoch noch nicht hinreichend in Bezug auf die Reduktion des Energieverbrauches evaluiert. In dieser Arbeit wird die Entwicklung einer automatisierten Messumgebung und die Durchführung einer vereinheitlichten Evaluierung gezeigt. Dabei wird der Energieverbrauch der bisher üblichen Implementierung mit Mikrocontroller als Referenz verwendet. Dem Gegenübergestellt wird die Umsetzung mit den autonomen, rekonfigurierbaren Modulen in Form eines Testchips. Die Evaluierung ergab bei dem betrachteten Szenario eine Reduktion des Energieverbrauches des alternativen Ansatzes von drei Zehnerpotenzen. Das Ergebnis zeigt das Potential des Ansatzes und der Ansatz wird weitere Verbreitung auch in allgemeinen Sensorknoten finden.

Abstract

Due to the spread of systems with microprocessor in recent years, the need of reducing their power consumption has increased. In particular, wireless sensor nodes represent a challenge in terms of limited energy resources and long service life. In addition to the classical implementation of the wireless sensor node with a microcontroller there is an alternative approach with autonomous reconfigurable modules in a system-on-chip. This approach has not yet been sufficiently evaluated in terms of reduction of energy consumption. In this work the development of an automated measurement environment and the implementation of a unified evaluation is shown. Therefore the power consumption of the classic implementation with a microcontroller is used as a reference. This is compared to an implementation with the autonomous reconfigurable modules in the form of a test chip. The evaluation found in the considered scenario a reduction of the energy consumption of the alternative approach by three orders of magnitude. The results show the potential of the approach and the approach is likely to be more widely used also in general sensor nodes.

Danksagung

An dieser Stelle möchte ich meinem Betreuer, Herrn Dr. Johann Glaser, für die interessante Aufgabenstellung, die wertvollen, technischen Diskussionen und seine Geduld bei der Fertigstellung dieser Arbeit danken.
Recht herzlich möchte ich mich bei meinen Eltern, Brigitte und Franz Blemenschitz, dafür bedanken, dass sie mir das Studium ermöglicht und mich auf meinem Weg unterstützt haben. Der größte Dank gebührt meiner Frau Christiane, die stets an meiner Seite ist und mich immer wieder ermuntert hat, wenn es mal nicht nach Plan lief.

Inhaltsverzeichnis

Abkürzungen

ADC	Analog-Digital-Converter
AHB	Advanced High-Performance Bus
ARM	Advanced RISC Machines
ASIC	Application Specific Integrated Circuit
CAD	Computer Aided Design
CLB	Configurable Logic Block
CMOS	Complementary Metal-Oxide-Semiconductor
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
DuT	Device under Test
eFPGA	embedded Field Programmable Gate Array
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FSMD	Finite State Machine with Datapath
GPIB	General Purpose Interface Bus
GPIO	General-Purpose-Input-Output
HAL	Hardware Abstraction Layer
IC	Integrated Circuit
ICT	Institut für Computertechnik
IPG	Input Pattern Gate
ISM	Input Switch Matrix
JTAG	Joint Test Action Group
JVM	Java Virtual Machine
LED	Light-Emitting Diode
LPM	Low-Power Mode
LUT	Look-Up Table
MCU	Microcontroller Unit
MISO	Master In Slave Out
MOSI	Master Out Slave In
MSB	Most Significant Bit
NPLC	Number of Power Line Cycles

NSR	Next State Register
PLL	Phase Locked Loop
PTL	Pass-Transistor Logic
RAM	Random Access Memory
RFID	Radio-Frequency Identification
RTC	Real Time Clock
RTL	Register Transfer Level
SCLK	Serial Clock
SCPI	Standard Commands for Programmable Instruments
SNOPS	Sensor Node Optimization through Power Simulation
SoC	System-on-Chip
SPI	Serial Peripheral Interface
SR	State Register
SS	Slave Select
SSG	State Selection Gate
TL	Transaction Level
TR-FSM	Transition-based Reconfigurable FSM
UART	Universal Asynchron Receive Transmit
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WSN	Wireless Sensor Network

1 Einleitung

Embedded Systems sind heute weiter verbreitet denn je. Aufgrund der zunehmenden Integration fällt den Endbenutzern meist nicht mehr auf, wie allgegenwärtig Embedded Systems sind. Das liegt an den vielfältigen Möglichkeiten und der unterschiedlichsten Leistungsfähigkeit der Systeme. Nicht vergessen darf dabei aber werden, dass Embedded Systems auch gewissen Einschränkungen unterworfen sind. Ein Beispiel dafür ist die Batterielaufzeit bei mobilen Consumer-Geräten, die ein hohes Maß an Energieeffizienz für möglichst lange Laufzeit voraussetzt.

Zu Beginn diesen Kapitels wird auf die Motivation dieser Arbeit in Abschnitt 1.1 eingegangen. Dabei werden unter anderem Anwendungsgebiete betrachtet, für welche die Ergebnisse dieser Arbeit relevant sind. In Abschnitt 1.2 werden die Ziele und die Aufgabenstellung mit den daraus folgenden Herausforderungen erklärt. Den Abschluss des Kapitels bildet Abschnitt 1.3 mit dem Ausblick auf die weitere Arbeit.

1.1 Motivation

Consumer-Geräte stellen jedoch nur einen Teil der Embedded Systems dar. Elektronische Schließsysteme, berührungsloses Bezahlen oder die Radio-Frequency Identification (RFID)-Technik sind ebenfalls Vertreter der Embedded Systems. Gerade in den letzten Jahren ist ein starker Trend hin zu kabel- oder berührungslosen Techniken zu bemerken. Allen gemein sind strengere Anforderungen an den Energieverbrauch als bei kabelgebundener Technik.

In diesem Abschnitt werden grundlegende Überlegungen dazu angestellt, warum es wünschenswert ist, den Energieverbrauch zu senken, welche Anwendungsgebiete im Fokus dieser Arbeit liegt und was eine rekonfigurierbare Architektur zur Reduktion der Leistungsaufnahme beitragen kann.

Den Energieverbrauch zu senken hat unterschiedliche Motive. Die folgende Aufzählung ist nicht ausschließend und erhebt keinen Anspruch auf Vollständigkeit.

- Eines der Motive kann auf das verstärkte Umweltbewusstsein in der Gesellschaft zurückgeführt werden. Unternehmen werben gezielt mit sparsamen Technologien um sich mit einem besseren Image einen Wettbewerbsvorteil gegenüber den Mitbewerbern zu sichern.
- Ein weiterer Beweggrund ist wirtschaftlicher Natur. Bei batteriebetriebenen Anwendungen resultiert ein geringerer Energieverbrauch in einer längeren Lebensdauer und verursacht damit geringere Wartungskosten.

- Ein anderes Motiv ist technisch begründet. Beim Energy Harvesting ist im Gegensatz zur Batterie nicht die Gesamtmenge der Energie, sondern die in einer Zeitspanne verfügbare Energie ausschlaggebend [vgl. KHZS07].

Für das untersuchte Anwendungsgebiet, Sensorknoten, sind besonders die letzten beiden Motive relevant. Unter Sensorknoten werden sowohl kabelgebundene Sensorknoten wie auch Funksensorknoten in Wireless Sensor Networks (WSNs) verstanden. Aufgrund dessen, dass Funksensorknoten energieautark sind, werden wesentlich strengere Anforderungen an den Energieverbrauch gestellt als bei den kabelgebundenen Pendants.

Funksensornetze bestehen aus mehreren Funksensorknoten die zumeist ein ad-hoc Netzwerk aufbauen. Abhängig von der eingesetzten Architektur wird die Kommunikation durch das Netz von Hop zu Hop bis zum Ziel übertragen oder einzelne Knoten übernehmen die Funktion eines Routers. Die Funksensorknoten bestehen aus einem Mikroprozessor mit Speicher und einem Transceiver-Modul. Außerdem sind noch eine Real Time Clock (RTC) und eine Sensorschnittstelle vorhanden, die heute meist zusammen mit dem Mikroprozessor ein System-on-Chip (SoC) bilden.

Funksensorknoten werden in vielen Bereichen eingesetzt, von denen nur einige kurz vorgestellt werden sollen. Im Bereich der Gebäudeautomatisierung gibt es Ansätze durch WSNs bereits den Installationsaufwand zu reduzieren [ÖPR+07]. Im System werden dann sowohl die Raumklimasteuerung mit Temperatursensoren, Heizung und Klimaanlage als auch Sicherheitssysteme wie Feuermelder eingebunden. In der Automobilindustrie werden nicht nur Reifendrucksensoren als Funksensorknoten ausgeführt sondern zusehends weitere Funktionen wie die Beleuchtung kabellos überwacht [TVF08]. Ein weiteres Anwendungsgebiet für WSNs ist die Medizintechnik mit Erfassung von Blutzucker, Blutdruck und Pulsoxymetrie [CLCC09]. Außerdem gibt es zahlreiche Anwendungsfälle für WSNs im Bereich der Umweltüberwachung wie Überwachung der Luftverschmutzung [KPM10], Waldbrand-Früherkennung [DRTAMA12], Überwachung der Wasserqualität [OMCH+07] oder Frühwarnung bei anstehenden Umweltkatastrophen wie Hochwasser [BRR08].

Allen Anwendungsfällen gemein ist die Tatsache, dass die eingesetzten Sensoren möglichst effizient abgefragt werden müssen. Dabei ist die klassische Realisierungen mit Mikrocontrollern nicht besonders effizient, da sie weit mehr Ressourcen in Form von Schnittstellen, Speicher und Rechenleistung mitbringen als nötig wäre. Als alternativen Ansatz dazu wurde die rekonfigurierbare Architektur am Institut für Computertechnik (ICT) entwickelt [Gla15, S. 3]. Die Architektur basiert auf autonomen, rekonfigurierbaren Modulen ergänzt um mehrere Busschnittstellen zur Abfrage der Sensoren. Dadurch, dass die autonomen, rekonfigurierbaren Module einige Aufgaben der Mikrocontroller wesentlich effizienter durchführen können, kann Energie gespart werden. Durch die Busschnittstelle wird allgemein eine größere Flexibilität durch ein breites Spektrum an verfügbaren Sensoren erreicht. Durch die Rekonfigurierbarkeit lässt sich eine größere Anzahl von Anwendungsfällen abdecken, was wiederum in größeren Stückzahlen des Chips resultiert und somit wirtschaftlicher ist. Durch die rekonfigurierbare Architektur wird jedoch der Trade-Off einer Energieeinsparung gegen erhöhten Entwicklungsaufwand eingegangen. Eine maßgeschneiderte Lösung für einen Anwendungsfall wäre noch effizienter jedoch mit Einschränkungen der Einsatzmöglichkeiten.

1.2 Ziel und Aufgabenstellung

Das Ziel dieser Arbeit ist die Evaluierung der rekonfigurierbaren Architektur in Hinblick auf den Energieverbrauch. Dazu ist eine Metrik notwendig, welche die Grundlage der Bewertung bildet. Außerdem wird eine Referenz mit Vergleichswerten und ein praxisnaher Anwendungsfall benötigt. Ein weiteres Ziel ist die Entwicklung einer universellen und erweiterbaren Messumgebung. Diese Messumgebung soll die Basis für die der Evaluierung zugrundeliegenden Messung sein. Ein zusätzlich Ziel ist die standardisierte Auswertung der Messergebnisse und deren grafische Aufbereitung. Die wesentlichen Herausforderungen birgt die Erstellung der flexiblen, automatisierten Messumgebung.

Aus den Zielen wird die konkrete Aufgabenstellung abgeleitet. Die Evaluierung der rekonfigurierbaren Architektur soll anhand eines Testchips, welcher mit autonomen, rekonfigurierbaren Modulen inklusive Sensor-Schnittstellen am ICT entwickelt wurde, durchgeführt werden. Als Referenz werden dem Testchip Ultra-Low-Power Microcontroller Units (MCUs) gegenübergestellt. Für diese MCUs müssen Evaluationsplatinen entwickelt und gefertigt werden, bevor sie nach der Inbetriebnahme gemessen werden können. Für die Messung ist ein bestehendes Eval-Plattform-Board zu einer automatisierten Messumgebung zu erweitern. Diese Messumgebung soll für zukünftige Projekte erweiterbar gestaltet werden und flexibel für unterschiedliche Prüflinge und Anwendungsfälle ausgelegt werden. Zur rascheren Anpassung an neue Anforderungen sollen grundlegende Parameter einfach parametrierbar sein. Die Auswertung soll wie die Messung automatisiert erfolgen und die ermittelten Daten grafisch darstellen.

Die Flexibilität der Messumgebung muss in mehrerer Hinsicht berücksichtigt werden. Einerseits sollen unterschiedliche Prüflinge gemessen und bewertet werden. Andererseits müssen Freiheiten bezüglich der Messgeräteauswahl für zukünftige Anforderungen gewahrt bleiben. Um eine Erweiterbarkeit zu gewährleisten, ist es notwendig Schnittstellen zu definieren und standardisierte Protokolle proprietären vorzuziehen. Die Forderung einer automatisierten Messung bedingt ein gewisses Maß an Fehlertoleranz um Benutzerinterventionen insbesondere bei längerer Laufzeit einer Messung vermeiden zu können.

1.3 Gliederung der Arbeit

Im diesem Abschnitt wird die weitere Gliederung der Arbeit erläutert. Kapitel 2 stellt den aktuellen Stand der Technik in Bezug auf die Reduktion des Energieverbrauches bei Integrierten Schaltungen auf unterschiedlichen Abstraktionslevel vor. Ergänzt wird das Kapitel mit dem Ansatz mit autonomen, rekonfigurierbaren Modulen. Kapitel 3 befasst sich mit den theoretischen Grundlagen der Messung und Evaluierung. Dabei wird die wissenschaftliche Untersuchung mit den notwendigen Ergebnissen, den Einflussgrößen und der Berechnung und Bewertung des Energieverbrauches eingehend erläutert. Kapitel 4 stellt die Entwicklung der Messumgebung mit allen benötigten Komponenten, wie der Steuerung des Messsystems oder der Mikrocontroller-Module, vor. Das Kapitel 5 beinhaltet die Messergebnisse und Besonderheiten der einzelnen untersuchten Chips. In Kapitel 6 werden die Ergebnisse für jeden untersuchten Chip einzeln betrachtet und anschließend gegenübergestellt und diskutiert. Kapitel 7 bildet mit der Zusammenfassung der Arbeit den Abschluss.

2 Stand der Technik

Das Ziel der Leistung- bzw. Energiereduktion lässt sich auf verschiedenen Ebenen des Designs bzw. Abstraktionsniveaus verfolgen und jeweils optimieren. Dabei werden folgende große Teilaspekte abgegrenzt.

- Zum einen die für die Fertigung des jeweiligen Chips verwendete Herstellungstechnologie,
- das jeweils angewendete Schaltungsdesign auf Transistor-Level und
- auf höherer Ebene die gewählte Architektur des Halbleiterchips oder des angedachten SoC.

In dieser Art wird in dieser Arbeit auch der aktuelle Stand der Technik auf diesen verschiedenen Abstraktionsebenen dargestellt. Im Besonderen wird an späterer Stelle auf einen am ICT verfolgten Ansatz eingegangen.

Allen betrachteten Ansätzen gemein ist, dass eine oder mehrere Komponenten der Gleichung 2.1 [vgl. RCN03, S. 225] optimiert werden. Die Gesamtverlustleistung setzt sich aus folgenden Teilen zusammen:

- P_{switch} Umladeverlustleistung
- P_{sc} Kurzschlussverlustleistung
- P_{stat} statische Verlustleistung

$$P_{\text{tot}} = P_{\text{switch}} + P_{\text{sc}} + P_{\text{stat}} \tag{2.1}$$

Die Summe aus Umlade- und Kurzschlussverlustleistung wird auch als dynamische Verlustleistung (P_{dyn}) bezeichnet.

2.1 Halbleitertechnologie

Die Wahl eines passenden Herstellungsprozesses ist für das Ergebnis der Schaltung von entscheidender Bedeutung. Aus diesem Grund muss die Auswahl nach mehreren gewichteten Faktoren passieren. Aufgrund verschiedener Randbedingungen lässt sich selten eine reine Optimierung auf

den Energieverbrauch durchführen. Es ist jedoch möglich, falls mehrere alternative Herstellungsprozesse verfügbar sind, den optimalen Prozess auszuwählen.

Für die Auswahl eines Prozesses müssen folgende Punkte, ohne Gewichtung und ohne Anspruch auf Vollständigkeit, berücksichtigt werden:

- Geschwindigkeit
- Verfügbarkeit
- Größe der Schaltung / Komplexität
- Anforderungen an Umgebungsbedingungen / Versorgungsspannungen
- Kosten

Im folgenden wird auf die aktuellen Trends bei den Halbleiterprozessen eingegangen. Generell kann man beobachten das der Trend zu immer kleineren Strukturgrößen geht [ORT14]. Dabei ist man aktuell bei 14 nm angelangt. Diese aktuellen, kleinen Strukturgrößen sind jedoch nicht unmittelbar für die große Anzahl an Schaltungen repräsentativ. Zum einen kosten solche Prozesse sehr viel in der Fertigung und Entwicklung, zum anderen wird nicht immer eine derart hohe Integrationsdichte benötigt. Das Gros der Schaltungen wird in älteren und damit gröberen Strukturen hergestellt. Durch die kleinen Strukturbreiten fallen die einzelnen integrierten Bauelemente kleiner aus, wodurch sich schnellere und komplexere Schaltungen auf kleinerer Chipfläche realisieren lassen.

Kleinere Strukturbreiten bieten jedoch nicht nur Vorteile, es werden nun Effekte dominanter, die bei früheren CMOS Prozessen nicht ins Gewicht gefallen sind. Insbesondere steigen die statischen Verluste durch Leckströme stark an [Hit07, S. 38]. Dies widerspricht jedoch dem Gedanken, den Energieverbrauch zu optimieren. Es gibt allerdings auch eigens optimierte Halbleiterprozesse, die auf geringe Leckströme optimiert sind. Diese sind in der Regel langsamer, als die entsprechend auf Performance ausgelegten Prozesse.

Eine Verbesserung der statischen Verluste wird in [CLC04] am Beispiel eines Field Programmable Gate Array (FPGA) vorgestellt. Durch eine Verknüpfung von Low-Threshold-Transistoren mit geringer Verzögerung und High-Threshold-Transistoren, welche geringere Leck-Ströme aufweisen, wird das Design optimiert. Hierbei werden nicht zeitkritische Signale oder Blöcke mit High-Threshold-Transistoren realisiert, unter anderem statisches Random Access Memory (RAM). Zeitkritische Schaltblöcke werden hingegen, insbesondere für Routing und Interconnections, mit geringerer Verzögerung umgesetzt. Diese Maßnahmen erfordern ein neues Schaltungsdesign, versprechen jedoch eine Reduktion der Leck-Ströme um 2 Größenordnungen bei gleicher Performance. Wesentlicher Nachteil ist ein erhöhter Bedarf an Chipfläche von ca. 17 %.

Eine andere Überlegung zum Thema Energieverbrauch kann von folgender Seite angegangen werden. Mit moderneren Technologien ist es möglich, kleine Threshold-Spannungen einzelner Transistoren zu realisieren [BMM01]. Damit lassen sich auch die Versorgungsspannungen verkleinern, jedoch nicht im beliebigen Maße. Man könnte fordern, dass die Versorgungsspannung möglichst klein wird, da sie nach Gleichung 2.2 [vgl. BMM01] quadratisch in den Energieverbrauch eingeht. Problematisch ist jedoch dabei, dass der Unterschied zwischen 0 und 1 im Strom immer geringer wird, da die Transistoren auch im Sub-Threshold Bereich Strom leiten und bei maximaler Spannung (Versorgungsspannung) nicht genug leiten wenn $V_{\mathrm{DD}} \leq V_{\mathrm{th}}$ ist. Damit ist das sogenannte

Voltage-Scaling gewissen Grenzen unterworfen. Es darf außerdem nicht vergessen werden, dass die Schaltungen auch mit Verringerung der Versorgungsspannung, langsamer werden.

$$
\begin{aligned}
P_{\text{switch}} &= \frac{1}{2} C_{\text{L}} V_{\text{DD}}^2 f_{\text{Clock}} E_{\text{SW}} \\
\text{mit} \quad C_{\text{L}} &= \text{Lastkapazität} \\
V_{\text{DD}} &= \text{Versorgungsspannung} \\
f_{\text{Clock}} &= \text{Taktfrequenz} \\
E_{\text{SW}} &= \text{Schaltwahrscheinlichkeit}
\end{aligned}
\tag{2.2}
$$

Wie beschrieben bieten sich auf der Ebene der Halbleitertechnologie Optimierungsmöglichkeiten sowohl bei statischen Verlusten, als auch bei dynamischen Verlusten. Aufgrund der stetig kleiner werdenden Strukturbreiten ist es notwendig insbesondere die statischen Verluste in Grenzen zu halten. Die Reduktion der dynamischen Verluste lässt sich gut durch Verringerung der Versorgungsspannung umsetzen. Dabei darf jedoch nicht vergessen werden, dass jede Optimierung auch Nachteile bei der Performance, des Bedarfs an Chipfläche oder der Kosten durch aufwendigere Herstellungsprozesse bringen kann.

2.2 Schaltungsdesign auf Transistor-Ebene

Parallel zur Energieoptimierung auf Halbleiterprozess-Ebene, bietet es sich an, auf der nächsthöheren Ebene weiter zu optimieren. Dabei muss das Ziel sein, die Leistungsaufnahme durch Schaltungsdesign zu reduzieren aber dabei trotzdem die für die Applikation notwendige Geschwindigkeit der Schaltung aufrecht zu erhalten. Dabei profitiert man davon, dass aktuelle Halbleiterprozesse für viele Schaltungen große Geschwindigkeitsreserven bieten. D.h. man versucht die Geschwindigkeitsreserven gegen eine Leistungseinsparung einzutauschen.

Die in diesem Abschnitt betrachteten Arbeiten zielen darauf ab, den dynamischen Verbrauch zu reduzieren. Nach Gleichung 2.2 kann jeweils ein optimierter Parameter identifiziert werden:

- Durch mehrere Spannungslevels lässt sich V_{DD} für einzelne Schaltungsteile senken.
- Durch Pass-Transistor Logic (PTL) lässt sich die Anzahl an Transistoren verringern, was zu einer geringen Lastkapazität C_{L} führt.
- Mit Clock-Gating lässt sich die Schaltwahrscheinlichkeit E_{SW} für den betroffenen Schaltungsteil auf Null setzen.
- Die Control-Generated-Clocks und ein Ansatz mit Dual-Edge-Triggered Flip-Flops reduzieren die effektive Schaltfrequenz f_{Clock}.

Durch Einführen mehrerer Spannungslevels oder variabler Versorgungsspannungen auf einem Chip, wird es möglich, zeitkritische Schaltungsteile mit höherer Spannung zu betreiben. Des Weiteren wird es möglich, einfache Interfaces zu Standard-Spannungslevels unter anderem für Input-Output (z.B. 3,3 V) zu bieten, während der Chip-Kern mit geringerer Spannung optimal arbeiten kann. Die wesentliche Herausforderung dabei ist, energieeffiziente Level-Shifter zu verwenden und für eine effiziente Spannungsverteilung am Chip zu sorgen [BMM01].

[IK11] untersuchten auf dem Sub-Threshold Regime basierende Logik-Zellen. Dabei werden Schaltungen mit normaler CMOS und mit PTL Implementierung verglichen. Neu an dem Ansatz ist die Kombination der zwei Technologien, PTL und Sub-Threshold Bereich. Die Gegenüberstellung wird anhand von D-Flip-Flops und einem Volladdierer in der jeweiligen Technologie exemplarisch ausgeführt. Dabei zeigt sich, dass der dynamische Energieverbrauch durch PTL, aufgrund der geringeren Anzahl an Invertern, geringer ausfällt. Beim statischen Verbrauch konnte im Mittel keine Verbesserung nachgewiesen werden. Auch gibt es keinen wesentlichen Geschwindigkeitsnachteil für PTL. Einschränkend muss jedoch erwähnt werden, dass es sich hierbei um Simulationsergebnisse handelt.

Eine weitere Methode zur Reduktion der dynamischen Schaltungsverluste, welche sehr häufig angewandt wird, ist Clock-Gating [BMM01]. Dabei wird durch selektives Aussetzen des Taktes an Flip-Flops oder auch ganzen Logik-Blöcken, die Umladung der Gate-Kapazitäten unterbunden und somit die dynamischen Verluste reduziert. Günstig ist, dass der Zustand in den Flip-Flops erhalten bleibt und somit nicht erst bei Aktivierung wiederhergestellt werden muss. Jedoch verursacht das Clock-Gating einen Schaltungsoverhead und ebenfalls eine Latenz bei Reaktivierung. Verständlicherweise können auch nur Ressourcen, welche gerade nicht benötigt werden, von diesem Mechanismus profitieren.

Zwei wesentlichen Problemen des Clock-Gatings widmet sich das Konzept der Control-Generated-Clocks in Finite State Machine with Datapaths (FSMDs) [RN00]. Das erste Problem betrifft mögliche Glitches (Störimpulse) auf der Taktleitung, sollte das Steuersignal erst nach dem Takt eintreffen. Das zweite Problem hängt mit der komplexen statischen Timing-Analyse beim Clock-Gating zusammen, wenn das Steuersignal von Daten abhängig ist, was laut [RN00] üblicherweise der Fall ist. Der Verbesserungsansatz ist folgender: Es wird ein neuer Takt für den Datenpfad eingeführt. Dazu ist es notwendig, die flankengesteuerten Flip-Flops im Datenpfad auf die jeweils andere Flanke umzustellen. Des weiteren wird der neue Datentakt durch Puls-Generatoren erzeugt und der Zustandsautomat muss entsprechend umkodiert werden. Laut Angaben der Autoren zeigen Analysen mit kommerziellen Synthese-Tools eine weitere Reduktion der Leistungsaufnahme um 16 % gegenüber Clock-Gating.

FPGAs werden jeweils als erste Integrated Circuits (ICs) in kleineren Strukturgrößen gefertigt und sind auf Performance und Durchsatz optimiert. [GZR99] stellt im Kontrast dazu ein energieeffizientes FPGA-Design vor. Die Arbeit identifiziert das Routing als Haupt-Verlustquelle mit etwa 65 % der dynamischen Verlustleistung. Aufgrund dieser Erkenntnis wird eine Optimierung des Routings mit 3 Klassen an Verbindungen angestrebt:

- Verbindungen zu den nächsten Nachbar-Zellen
- ein symmetrisches Verbindungsnetz
- hierarchische Verbindungen

Ergänzt wird das Konzept mit Dual-Edge-Triggered Flip-Flops in den Logik-Blöcken. Damit sollen beide Taktflanken ausgenutzt werden können und bei halbem Takt der gleiche Durchsatz erzielt werden. Im Vergleich mit kommerziellen FPGAs geben die Autoren ein Verbesserungspotenzial von mehr als einer Zehnerpotenz an.

Während statische Verluste in den erwähnten Arbeiten keine Rolle spielen, können die dynamischen Verluste beim Schaltungsdesign auf Transistor-Ebene reduziert werden. Konzepte wie

Clock-Gating versprechen dabei großes Einsparungspotenzial. Der Ansatz der Control-Generated-Clocks und der Ansatz mit Dual-Edge-Triggered Flip-Flops muss jeweils zu den asynchronen Schaltwerken eingeordnet werden, welche nach wie vor Forschungsgebiet sind.

2.3 Schaltungsdesign auf Architektur-Ebene

Eine weitere Verbesserung hinsichtlich des Energieverbrauches wird auf der Architektur-Ebene erreicht. Eine Einteilung kann in allgemeine Optimierungen auf Basis bereits vorgestellter Techniken und anwendungsspezifische Verbesserungen vorgenommen werden.

Bei allgemeinen Optimierungen wird auf bereits vorgestellte Techniken zurück gegriffen. Ein Beispiel für die Kombination verschiedener Techniken ist das Dynamic Power Management. Abhängig von der Anwendung oder Auslastung eines Prozessors werden Clock Gating oder eine Verringerung des Taktes durchgeführt. Durch Anpassen der Versorgungsspannung (nach entsprechender Anpassung des Taktes) oder Abschalten der Versorgungsspannung nicht benötigter Schaltungsteile kann der Energieverbrauch weiter gesenkt werden [BMM01]. Diese Technik findet sich aufgrund der immer restriktiveren Anforderungen auch in Mikrocontrollern wieder. Exemplarisch sei hier auf die Serie MSP430 von Texas Instruments verwiesen, welche sich durch ein komplexes System zur Optimierung der Taktverteilung am Chip auszeichnet (vgl. Basic-Clock-Module+ [Tex13, S. 274]).

Ebenfalls eine Optimierung auf unterschiedlichen Abstraktionsebenen für Low-Power untersucht [DM95]. Einige der beschriebenen Optimierungen sind heutzutage in kommerziellen Synthese-Tools bereits Standard. Dazu zählen die Vereinfachung von logischen Ausdrücken (don't-care Optimierung) oder die Faktorisierung komplexer Logik-Ausdrücke. Des weiteren wird die Zustandskodierung von Finite State Machines (FSMs) oftmals an die zugrundeliegende Architektur angepasst.

Die Anwendungsdomäne hat wesentlichen Einfluss auf die Effizienz der gewählten Architektur. Die Anwendungsdomänen lassen sich grob in zwei Gruppen einteilen. Die erste Gruppe ist mit Fokus auf Daten- oder Signalverarbeitung und die zweite mit Fokus auf Steuerungsaufgaben. Beide Gruppen haben unterschiedliche Ansprüche an die Funktionalität. Ein Beispiel für Signalverarbeitung sind digitale Filter, die von effizienten Multiplizierer-Addierer-Blöcken und breiten, parallelen Datenpfaden profitieren. Im Gegensatz dazu stehen steuerungszentrierte Aufgaben, die einzelne Signale verarbeiten und hohe Flexibilität fordern. Aufgrund der benötigten hohen Flexibilität werden Mikrocontroller für Steuerungsaufgaben eingesetzt.

Einen Ansatz, welcher auf dynamisch rekonfigurierbarer Hardware basiert beschreibt [CCMM04]. Unter „dynamisch“ wird hierbei eine Rekonfiguration zur Laufzeit verstanden. Das Ziel ist, bei der Flexibilität der Hardware ein vergleichbares Level wie in Software zu erreichen und damit verbunden bessere Performance auf geringerer Chipfläche. Das betrachtete Hauptproblem ist der fehlende Support für den Workflow. Umgesetzt wird das Konzept mit einem Konfigurations-Controller in Hardware und einem dreistufigen Framework mit folgenden Teilen:

- Die erste Stufe, Design und Validierung, validiert das dynamisch rekonfigurierbare System auf dem Transaction Level (TL) und übersetzt das Design auf Register Transfer Level (RTL).
- Die zweite Stufe, Partitionierung und Scheduling, extrahiert die benötigten Hardwareblöcke und definiert ihre zeitliche Abhängigkeit.

- Die letzte Stufe, Synthese und Rekonfigurierbarkeit, generiert die benötigten Konfigurations-Bitstreams und fügt den Laufzeit-Konfigurations-Controller ein.

Der wesentliche Nachteil des Ansatzes ist, dass die Rekonfigurationszeit in das Ergebnis mit einfließt. Darum ist eine Performanceverbesserung erst ab mehreren Hundert Operation möglich. Als Beispiel nennen die Autoren hierzu digitale Filter.

Eine andere Arbeit stellt FPGAs und anwendungsspezifische embedded Field Programmable Gate Arrays (eFPGAs) gegenüber, welche speziell auf arithmetische Anwendungen optimiert sind. Durch Reduktion der Routing-Möglichkeiten, wo die meiste Energie verbraucht wird, und Orientierung an einem Datenpfad-Modell, soll der Energieverbrauch gesenkt werden. Die Logikblöcke enthalten neben den üblichen Look-Up Tables (LUTs) auch 2 Volladdierer in Hardware. Des Weiteren sind die Routing Möglichkeiten unter Berücksichtigung der Lokalität stark reduziert. Das Verbesserungspotential wurde anhand einer Umsetzung in 130 nm Complementary Metal-Oxide-Semiconductor (CMOS) bei drei Anwendungen (8x8 Multiplizierer, 4-Bit Multiplizierer-Addierer, 4-Bit Butterfly (Bit Permutation)) untersucht und brachte eine Verringerung des Energieverbrauches um Faktor 5-10 [SNBN06].

Ein weiterer Ansatz mit Fokus auf Datenverarbeitung beschreibt die Partitionierung einer FSMD in Sub-Prozesse mit jeweils Controller und Datenpfad. Voraussetzung zur Verringerung des Energieverbrauches ist, dass die einzelnen Sub-Prozesse unabhängig und somit abschaltbar sind. Das zu lösende Problem besteht darin, durch die Partitionierung mehr Energie einzusparen, als durch Kommunikation zwischen den Sub-Prozessen verbraucht wird. Dazu wird eine Analyse der Daten zwischen den Blöcken und eine anschließende Optimierung mittels Heuristik durchgeführt. Die erzielte durchschnittliche Leistungsreduktion ist mit 42 % angegeben [HVH99].

Mit grob-granularen rekonfigurierbaren Architekturen lässt sich, im Vergleich zu konventionellen FPGAs, das Routing effizienter gestalten. Dadurch entsteht weniger Konfigurationsoverhead und der Chip ist effizienter. Eine Kategorisierung dieser Architekturen kann in 3 Gruppen erfolgen [Har01]:

- Anordnung in 2D-Matrix
- Anordnung in mehreren 1D-Matrizen
- Layout mit zentralem Kreuzschienenverteiler

Die Funktionsblöcke der untersuchten Architekturen sind mit teils unterschiedlicher Granularität ausgeführt. Die Anwendungsdomänen sind jeweils sehr speziell wie Multimediaanwendungen oder digitale Signalverarbeitung.

Während bei Mikrocontrollern die Verringerung des Energieverbrauches bereits länger Thema ist, gibt es auf dem Gebiet der FPGAs kaum Energiesparfeatures in selbem Umfang. Dem begegnet „Pika“ [TRD+07], ein Low-Power FPGA Hard-IP-Core speziell für batteriebetriebene Anwendungen. Der Ansatz von Pika ist der Transfer von etablierten Designtechniken für Application Specific Integrated Circuits (ASICs) zu FPGAs. Basis der Optimierung ist ein Xilinx Spartan-3 FPGA Kern, welcher entsprechend angepasst wird. Im Fokus stehen dabei das Routing, welches zum großen Teil für die dynamischen Verluste verantwortlich ist und die Verringerung der statischen Verluste, ebenfalls durch Routing aber auch durch den Konfigurationsspeicher verursacht. Um das Design mit vorhandener Synthese-Software programmieren zu können und mit etablierter Prozesstechnologie herstellen zu können, wurde auf Kompatibilität zu bestehenden Tools und Prozessen geachtet. Erreicht wird die Verbesserung wiederum durch die Kombination mehrerer Techniken:

- Voltage Scaling bietet bei FPGAs großes Einsparungspotential, da FPGAs typischerweise auf gute Leistung ausgelegt sind und über Leistungsreserven verfügen.

- Da der Konfigurations-Speicher keine zeitkritische Komponente ist, können langsame aber energieeffiziente Transistoren verwendet werden. Das untersuchte Design verwendet High-Threshold Transistoren mit einer Gate-Oxid-Schicht mittlerer Dicke.

- Durch Power-Gating jeweils ganzer Configurable Logic Blocks (CLBs) lassen sich nicht benötige Ressourcen, bei moderatem Overhead, abschalten.

- Eine Erweiterung zu Power-Gating stellen die Standby Modi dar, die eine Untermenge oder alle Power-Gates abschalten können.

Um die Standby-Modi zu realisieren wird der Zustand der Flip-Flops abgespeichert und eine einfache Steuerungseinheit für Standby und Wake-Up eingesetzt. Wesentliche Einschränkung des Designs ist, dass keine Spezialblöcke optimiert wurden wie Multiplizierer, Taktaufbereitung oder Block-RAM. Berechnungen nach Post-Layout Simulationen zeigen Energieeinsparungen von 46 % im aktiven Zustand und 99 % im Standby [TRD+07].

Auf dem Gebiet der Mikrocontroller gibt es neben Application Notes der Chip-Hersteller [Ive11] auch wissenschaftliche Arbeiten, welche sich mit Design für Low-Power befassen. Eine Arbeit davon beschreibt Low-Power Modes (LPMs) in MCUs [Das09]. Das erklärte Ziel ist, die MCU möglichst lange in Sleep Modes verweilen zu lassen und alle nicht benötigten Chip-Komponenten abzuschalten, um dadurch den Durchschnittsverbrauch zu senken. Die wesentliche Anforderung an die MCU ist dabei die Rückkehr zur normalen Programmausführung mit geringster Verzögerung. Zur Latenz tragen vor allem die Dauer bis zur Stabilisierung des Taktes und die Interrupt-Routine bei. Je nach MCU wird die Optimierung der Aufwach-Zyklen unterschiedlich realisiert und oftmals mehrere LPMs in einem Chip umgesetzt, um das Programmdesign optimal an die Anwendung anpassen zu können.

Während MCUs aufgrund geringer Programmkomplexität und besserem Optimierungspotential in Assembler oder hardwarenahen Programmiersprachen programmiert werden, wählt die Arbeit [MKC02] einen anderen Ansatz. Der FemtoJava genannte Mikrocontroller ist Stack-basiert und kann direkt Java Bytecode ausführen. Der Ansatz zur Reduktion des Energieverbrauchs zielt auf eine Reduktion der Speicherzugriffe. Dazu wird eine Harvard-Architektur verwendet, welche kompatibel zur Java Virtual Machine (JVM) Spezifikation gehalten ist. Der Programmcode wird durch ein eigenes Computer Aided Design (CAD) Framework in Very High Speed Integrated Circuit Hardware Description Language (VHDL) übersetzt und direkt als Logik abgebildet. Zusammen mit dem FemtoJava Mikrocontroller in VHDL kann das Design für ein FPGA mit einem konventionellen Synthese-Tool synthetisiert werden. Durch Optimierungen bei der Synthese werden nur benötigte Befehle des Mikrocontrollers umgesetzt. Die durchschnittliche Ersparnis in den untersuchten Applikationen wird mit 31,46 % angegeben.

Auf Architektur-Ebene gibt sehr großes Energieeinsparungspotential, im Besonderen im Bereich der Signal- und Datenverarbeitung. Für Steuerungsaufgaben werden weiterhin Mikrocontroller eingesetzt, welche durch Overhead bedingt durch ihre Flexibilität als nicht besonders energieeffizient betrachtet werden müssen.

2.4 Ansatz mit autonomen, rekonfigurierbaren Modulen

Im Bereich der WSNs ist ein besonders geringer Energieverbrauch erforderlich, um mit einer Batterieladung eine möglichst hohe Lebensdauer des Knotens zu erreichen. Um die Energieeffizienz zu erhöhen, können verschiedene Aufgaben, welche sonst von der Central Processing Unit (CPU) übernommen werden, einem autonomen, rekonfigurierbaren Logikmodul übertragen werden. Dadurch kann die CPU länger in einem LPM verweilen und benötigt weniger Energie [Gla15, S. 1f]. Abbildung 2.1 zeigt einen WSN Knoten mit zwei zusätzlichen autonomen, rekonfigurierbaren Modulen für ein Sensor-Interface und ein Kommunikations-Interface.

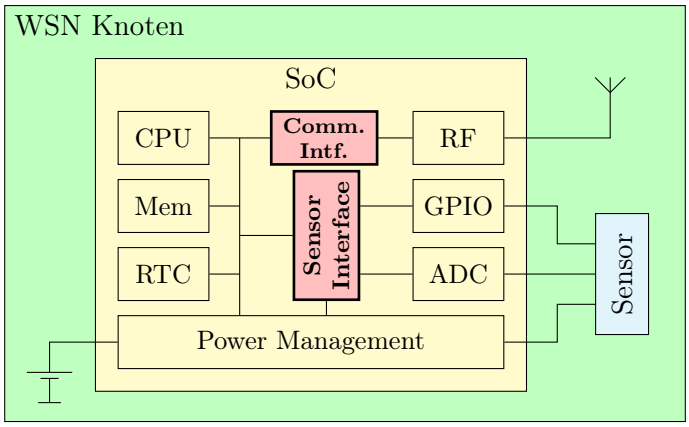


Abbildung 2.1: Architektur des SoC [Gla15, Figure 1.2, S. 3, mit Änderungen]

Das Design eines autonomen, rekonfigurierbaren Moduls wird in [GGHG11] als zweiphasiger Prozess beschrieben und soll hier kurz zusammengefasst werden. Die erste Design-Phase oder Pre-Silicon Design Phase widmet sich der Spezifikation und dem Design der logischen Blöcke und Funktionen inklusive der Verbindungen. Die erste Phase endet mit dem letzten Schritt der Produktion des ICs. In der zweiten Design-Phase oder Post-Silicon Design Phase wird die eigentliche Applikation definiert. Nach Synthese und Mapping auf die in der ersten Phase erstellten Ressourcen, kann nach Platzierung und Routing die Konfigurationsinformation erstellt werden. Im so entwickelten System bleibt der CPU die Aufgabe der initialen Konfiguration der autonomen, rekonfigurierbaren Module beim Einschalten.

Ein wichtiges Bauteil des Ansatzes mit autonomen, rekonfigurierbaren Modulen ist die Transition-based Reconfigurable FSM (TR-FSM). Abbildung 2.2 zeigt die Architektur einer TR-FSM. Der Grundgedanke ist, dass direkt die Zustandsübergänge statt der logischen Ein- und Ausgangsfunktionen der FSM implementiert werden. Die TR-FSM ist als Reihe von Transition Rows organisiert. Im Weiteren soll auf die einzelnen Blöcke genauer eingegangen werden.

Das State Selection Gate (SSG) vergleicht Zustand des State Register (SR) mit dem gespeicherten Konfigurationswert und schaltet den betreffenden Zustandsübergang frei. Die konfigurierbare Input Switch Matrix (ISM) wählt eine Untermenge der Eingangssignale aus und verbindet sie mit dem Input Pattern Gate (IPG). Das IPG ist ein konfigurierbarer Logik-Block. Wenn die Bedingungen für den Zustandsübergang erfüllt sind, wird der Ausgang des entsprechenden IPG auf logisch „1“ gesetzt. Die Architektur verfügt über mehrere Transition Rows mit unterschiedlicher Eingangsbreite. Der nächste Zustand wird durch das Next State Register (NSR) für den aktiven Übergang ermittelt. Die Anzahl der Zustandsübergänge ist limitiert, jedoch flexibel genug

um wenige Zustände mit vielen Übergängen oder aber viele Zustände mit weniger Übergängen abzubilden [GDHG11].

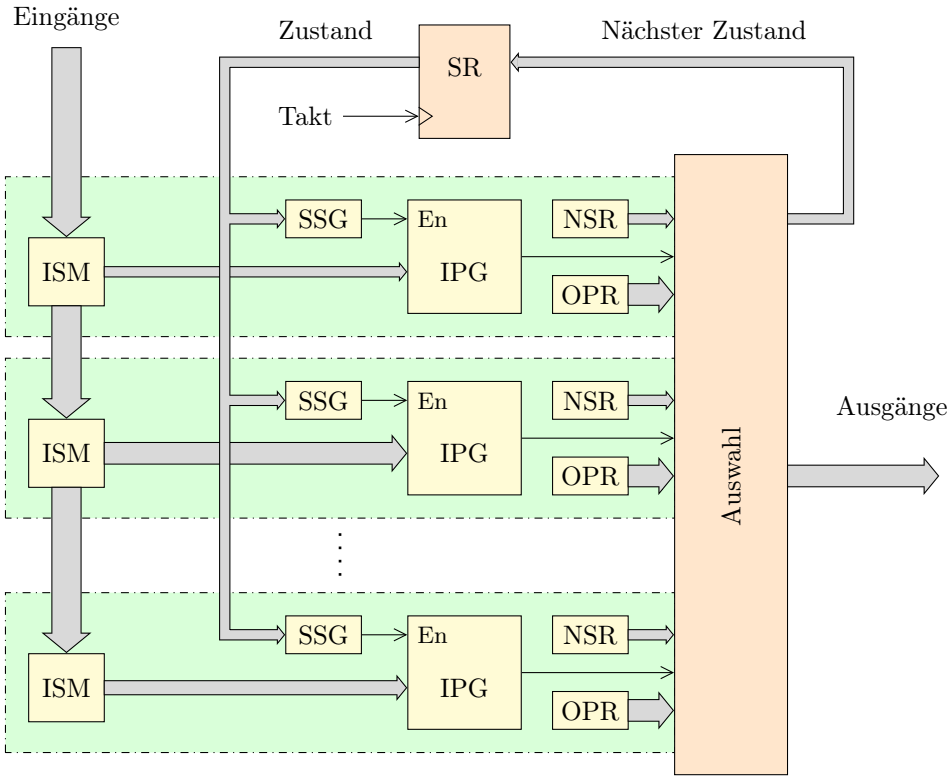


Abbildung 2.2: Blockschaltbild der TR-FSM [Gla15, Figure 3.14, S. 81, mit Änderungen]

Um das Konzept zu evaluieren wurde ein Testchip mit Standard-Zellen in 130 nm CMOS und mit 6 Metall-Layern produziert. Dieser Testchip ist auf eine typische Anwendungs-Klasse ausgelegt, ein rekonfigurierbares Sensor-Interface. Dabei wird ein Sensor in periodischem Zeitintervall aktiviert. Während einer Wartezeit führt der Sensor eine Analog-Digital-Wandlung durch, die abgewartet werden muss. Anschließend wird der Wert vom Testchip gelesen und ein Vergleich mit einem zuvor gemessenem Wert durchgeführt. Wenn die Differenz der Werte einen Schwellwert überschreitet, wird der neue Wert gespeichert und die MCU für die Weiterverarbeitung aktiviert. In der weiteren Verarbeitung können sowohl komplexe Berechnungen durchgeführt werden als auch der Messwert über ein Netzwerk übertragen werden.

Der Testchip stellt ein Beispiel für ein autonomes, rekonfigurierbares Modul dar. Abbildung 2.3 zeigt das Blockschaltbild des Testchips. Der Kern besteht aus folgenden drei Komponenten: der Steuerlogik, dem Byte Speicher und einer arithmetischen 16-Bit Einheit. Ergänzt wird der Kern durch periphere Schnittstellen für Analog-Digital-Converter (ADC) und serielle Busmaster (I^2C, SPI, 1-Wire, PWM, SENT und SPC). Die Steuerlogik besteht aus 4 unabhängigen TR-FSMs, welche zwischen 5 und 15 Eingangssignale verarbeiten und bis zu 61 Zustandsübergänge abbilden können.

Die Konfiguration des Testchip erfolgt durch Schieberegister, welche vergleichbar dem Programmiervorgang bei FPGAs mit Joint Test Action Group (JTAG) Schnittstelle geladen werden können. In einem SoC könnte die Konfiguration auch in einem entsprechenden Speicherbereich abgelegt und automatisch in die Schieberegister geladen werden. Nach der Konfiguration kann der

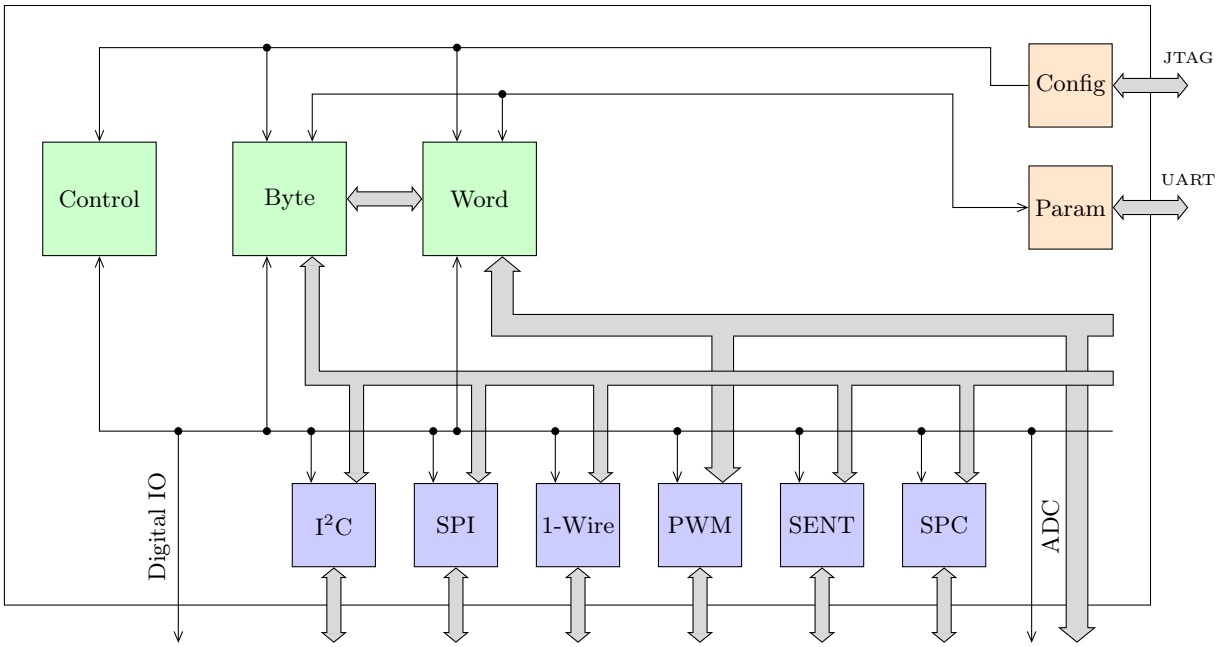


Abbildung 2.3: Blockschaltbild des Testchips [Gla15, Figure 5.7, S. 135]

Testchip bereits seine Funktion aufnehmen. Im Gegensatz zur Konfiguration steht die Parametrisierung für Anpassungen während der Laufzeit zur Verfügung. Mit der Parametrisierung sind einzelne Byte Werte des Speichers über eine Universal Asynchron Receive Transmit (UART)-Schnittstelle nach RS232 adressierbar. Damit lässt sich der Schwellwert der Sensormessung anpassen oder es kann der Sensorwert abgefragt werden [GDHG11].

Der Testchip wurde vollständig manuell als Design in VHDL entwickelt. Um die Konfiguration zu generieren, wurden in der Programmiersprache Pascal Klassen erstellt, welche die Hardware-Funktionen abbilden. Des Weiteren wird in einem Programm die Applikation definiert, welches die entsprechenden Klassen instanziiert und Methoden ausführt. Abschließend sind weitere Methoden nötig, um das Design auf die Hardware zu mappen und den Konfigurations-Bitstream zu erstellen. Eine Verifikation ist bei diesem Prozess erst am Ende möglich. Dazu wird der VHDL Code des Testchips in einer Testbench betrieben und die Konfiguration mit JTAG geladen [Gla15, S. 133f].

Eine Weiterentwicklung mit automatisierter Design Methodik wird in [Gla15, Kapitel 4, S. 85ff] beschrieben. Zu Beginn müssen dabei Anforderungen und Anwendungsszenario des geplanten SoC definiert werden. In einem weiteren Schritt nimmt der Designer eine Partitionierung der Aufgaben zwischen CPU und rekonfigurierbarer Hardware vor. Anschließend definiert der Designer die Interfaces zwischen CPU und rekonfigurierbarem Modul und fügt dem SoC nach Bedarf weitere Module, wie Speicher und serielle Busmaster hinzu.

Die Funktionalität des rekonfigurierbaren Moduls wird mittels einer Sammlung von Beispiel-Applikationen präzise spezifiziert. Dabei handelt es sich um Logik-Designs in VHDL oder Verilog. In einem semi-automatischen Ablauf wird aus den Beispiel-Applikationen das rekonfigurierbare Modul als RTL Design generiert. Bei diesem Prozess wird ebenfalls die Konfiguration für jede Beispiel-Applikation erstellt. Darüber hinaus bietet das so generierte rekonfigurierbare Modul ausreichend Flexibilität um neue Applikation zu implementieren [Gla15, Kapitel 4, S. 85ff].

3 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Überlegungen zur Messung und Evaluierung beschrieben. Abschnitt 3.1 beschreibt die grundlegenden Überlegungen zur Evaluierung und definiert das Ziel. In Abschnitt 3.2 werden die Anforderungen an die Messung sowie die sich daraus ergebenden Herausforderungen erklärt. Abschnitt 3.3 befasst sich mit den für die Evaluierung notwendigen Ergebnissen. Der Abschnitt 3.4 listet die beeinflussbaren Variablen und die zu erwartenden Auswirkung auf. Die Anforderungen an die Messumgebung sind in Abschnitt 3.5 erläutert. In Abschnitt 3.6 wird auf die Berechnung des Stromverbrauches und im Weiteren auf die Berechnung des Energieverbrauches eingegangen. Der Abschnitt 3.7 beschreibt, wie die erhaltenen Ergebnisse fair bewertet werden können.

3.1 Ausgangspunkt

Der Trend bei elektrischen Geräten und elektronischen Schaltungen geht in Richtung geringerer Leistungsaufnahme. Wie bereits in der Einleitung erwähnt, stellen WSNs besonders strenge Anforderungen an den Energieverbrauch eines Knotens um die Lebensdauer zu verlängern. Dazu werden bei derzeitigem Stand der Technik Implementierungen mit MCUs eingesetzt. Um die Leistungsaufnahme weiter zu senken, sind weitergehende Ansätze notwendig. Abschnitt 2.4 stellt einen solchen Ansatz vor. Um zu klären welchen Vorteil der Ansatz in der Praxis bringt, ist eine eingehende Evaluierung notwendig.

Der Titel dieser Arbeit, „Evaluierung der Reduktion der Leistungsaufnahme durch eine rekonfigurierbare Architektur“, ist gleichzeitig auch der Ausgangspunkt für die theoretischen Überlegungen. Dabei wird die Idee aufgegriffen, ein SoC durch ein autonomes, rekonfigurierbares Modul, wie bereits in Abschnitt 2.4 vorgestellt, zu erweitern. Der Vorteil dieses Ansatzes ist, dass die CPU längere Schlafphasen durchlaufen kann. Außerdem wird durch das Verlagern der Sensormessung hin zu einem autonomen Modul, der bereits angesprochene Overhead des Mikrocontrollers vermieden. Zusammen soll das in einer Reduktion der Leistungsaufnahme des Gesamtsystems resultieren. Das erklärte Ziel der Arbeit ist die Evaluierung der Reduktion der Leistungsaufnahme dieses Ansatzes.

Evaluierung im allgemeinen Sinn bedeutet eine Untersuchung und Bewertung. Insbesondere für die Bewertung ist eine Referenz notwendig, um die richtigen Schlüsse aus Vergleichswerten ziehen zu können. Des Weiteren ist das Kriterium, anhand dessen die Bewertung vorgenommen wird ausschlaggebend. Im vorliegenden Fall wird die Leistungsaufnahme oder, daraus abgeleitet, der Energieverbrauch als Bewertungskriterium herangezogen. Der Energieverbrauch ist nach

Gleichung (3.1) durch die Leistungsaufnahme definiert.

$$E = \int P(t)\mathrm{d}t \tag{3.1}$$

Wie in Abschnitt 2.3 erwähnt, ist die Anwendungsdomäne bestimmend für den Erfolg architekturabhängiger Optimierungen. Die Untersuchung wird deswegen bei einer domänenspezifischen Aufgabe durchgeführt. Die Anwendungsdomäne des autonomen, rekonfigurierbaren Moduls ist ein Sensorinterface. Darum wird in dieser Arbeit ein praxisnaher Anwendungsfall für ein Sensorinterface gesucht.

Der Erfolg des Ansatzes mit einer rekonfigurierbaren Architektur wird durch den geringeren Energieverbrauch gegenüber dem konventionellen Ansatz mit MCU definiert. Nach dem aktuellen Stand der Technik wird in WSN Knoten eine MCU eingesetzt. Daher wird eine MCU als Referenz verwendet. Wesentlich für die Aussagekraft der Referenz ist, dass die Untersuchung bei gleichen Randbedingungen und gleichen Aufgaben erfolgt. Um den Vorteil der rekonfigurierbaren Architektur zu quantifizieren wird der Energieverbrauch sowohl mit rekonfigurierbarer Architektur, als auch mit konventioneller Umsetzung bestimmt.

Neben der Umsetzung mit MCU und autonomen, rekonfigurierbaren Modulen ist auch eine Realisierung mit FPGA möglich. In dieser Arbeit wird diese Umsetzung nicht weiter betrachtet, da Analysen aufzeigen, dass der Ansatz mit autonomen, rekonfigurierbaren Modulen noch effizienter ist [GHDG09]. Begründet wird das mit der feinen Granularität der Logikfunktionen und dem Routing am FPGA. In dieser Arbeit wird der in Abschnitt 2.4 vorgestellte Testchip als Umsetzung eines autonomen, rekonfigurierbaren Moduls verwendet.

In einer Application Note eines Mikrocontroller-Herstellers zum Thema Low-Power Design [Ive11, S. 5] wird der Durchschnittsverbrauch des Stromes als Kenngröße angegeben. Damit lässt sich bei gegebener Versorgungsspannung der durchschnittliche Energieverbrauch berechnen. In MCU Datenblättern ist der Stromverbrauch für Betriebsmodi wie volle Aktivität und LPMs jeweils einzeln ausgewiesen. Bei bekanntem Verhältnis der Aktivitäten lässt sich daraus der Durchschnittsverbrauch berechnen. Um das Verhältnis der Aktivitäten festlegen zu können, muss auch die Anzahl der Sensormessungen im Betrachtungszeitraum bekannt sein.

Eine größere Flexibilität beim Vergleich bietet dagegen der Energieverbrauch bezogen auf eine Sensormessung. Dadurch, dass das Ergebnis unabhängig von der Häufigkeit der Sensormessung ist, können die Ansätze und Implementierungen unabhängig vom Betrachtungsintervall verglichen werden. Der statische Verbrauch muss dabei aus dem Ergebnis herausgerechnet werden, siehe Abschnitt 3.3. Dadurch werden prozess-technische Einflüsse reduziert und ein besserer Schluss auf die Veränderung auf architektureller Ebene möglich.

Zusammengefasst wird in dieser Arbeit der Einfluss der Architektur anhand des Vergleichs von konventionellem Ansatz mit MCU und dem Ansatz mit autonomen, rekonfigurierbaren Modulen bestimmt. Ausgedrückt soll der Vorteil als Energieverbrauch pro Sensormessung, für den jeweiligen Ansatz, werden.

3.2 Vorgehensweise bei der Evaluierung

In diesem Abschnitt wird der Vergleiches zwischen konventionellem Ansatz mit MCU und Testchip als Vertreter eines autonomen, rekonfigurierbaren Moduls näher erläutert. Außerdem wird das

Sensorinterface mit Serial Peripheral Interface (SPI) und einem Temperatursensor als Testfall abgeleitet.

Der durchschnittliche Stromverbrauch kann mit einem Amperemeter gemessen werden. Daraus wird der Energieverbrauch bestimmt. Die Gleichung (3.2) zeigt die Abhängigkeit des Energieverbrauchs von Spannung und Strom im betrachteten Zeitintervall. Durch bestimmen der Komponenten und anschließende Berechnung lässt sich der Energieverbrauch bestimmen.

$$
\begin{aligned}
E &= \int_0^T u(t) \cdot i(t) \mathrm{d}t \\
\text{mit} \quad u(t) &= \text{zeitabhängige Spannung} \\
i(t) &= \text{zeitabhängiger Strom} \\
[0, T] &= \text{betrachtetes Zeitintervall}
\end{aligned}
\tag{3.2}
$$

Aufgrund der Ausrichtung der Anwendung auf Low-Power sind kleine Ströme bei der Messung zu erwarten. Diese werden üblicherweise durch eine Messung mit Messshunt und anschließender Integration ermittelt. Dabei kann die Integrationszeit nicht beliebig verlängert werden, da sich auch Fehler durch die Integration aufsummieren. Des weiteren ist die Messung aufgrund der kleinen Ströme wesentlich anfälliger für Störungen und Einstreuungen aus der Messumgebung. Die Herausforderung ist es, Störungen wie die 50 Hz Netzeinkopplung zu eliminieren.

Im betrachteten Zeitintervall $[0, T]$ wird ebenfalls die inaktive Zeit mit gemessen. Das bedeutet, die Anteile der Leckströme und des Taktnetzes müssen über einen Zyklus bestimmt werden. Anschließend lassen sich beide Anteile aus dem Ergebnis herausrechnen, siehe Abschnitt 3.6.

Aufgrund begrenzter Ressourcen, stand während der Erstellung der Arbeit kein SoC zur Evaluierung zur Verfügung. Darum wird das SoC auf eine MCU und ein autonomes, rekonfigurierbares Modul, den Testchip (siehe Abschnitt 2.4), abgebildet. Vorteilhaft ist, dass durch die zwei unabhängigen Implementierungen ein direkter Vergleich ermöglicht wird. Des Weiteren bietet diese Umsetzung eine höhere Flexibilität, da ebenfalls verschiedene MCUs miteinander verglichen werden können. Einen Nachteil stellt der Automotive-zertifizierte Testchip-Herstellungsprozess dar. Dieser ist vorrangig auf Störfestigkeit optimiert und nicht auf Energieeffizienz. Dadurch wird das zu erwartende Ergebnis schlechter ausfallen, gleichzeitig jedoch die Aussagekraft des Konzeptes gestärkt.

Die Messung wird jeweils mit MCU oder Testchip durchgeführt. Eine Messung des kombinierten Systems ist nicht notwendig. Dies ist dadurch begründet, dass die MCU auf dem SoC während der Aktivität des Sensorinterface des Testchips im LPM verweilt und nur den geringen statischen Stromverbrauch aufweist. Dieser statische Anteil würde wiederum durch die Definition des Energieverbrauches pro Sensormessung aus dem Ergebnis herausgerechnet werden. Wenn sowohl MCU als auch Testchip gemeint sind, wird in dieser Arbeit der Begriff Device under Test (DuT) verwendet. Der Energieverbrauch pro Sensormessung wird abhängig vom jeweiligen DuT mit dem Formelzeichen E_{MCU} für MCUs und E_{TC} für den Testchip abgekürzt.

Ein Sensorinterface hat die Aufgabe den Messwert eines Sensors zu erfassen und weiter zu verarbeiten oder zur Weiterverarbeitung an andere Instanzen zu übertragen. Die Integration des Sensors in ein SoC hätte Vorteile beim Energieverbrauch, weil die Steuerung und Kommunikation direkt über den Datenbus oder Memory-Mapped Register erfolgen kann. Dem entgegen steht der Nachteil, dass die Flexibilität bei der Auswahl des Sensors verloren geht. Aufgrund der Vielzahl an

unterschiedlichen Sensoren, ist es wirtschaftlicher, entsprechende Schnittstellen für die Sensoren zur Verfügung zu stellen. Zu diesen Schnittstellen zählt das SPI [Fre07, Kap. 8, S. 291].

SPI gilt, obwohl es keine Normung gibt, als Industriestandard und findet weite Verbreitung. Aufgrund des einfachen Protokolls verfügen MCUs meist bereits über mindestens ein SPI Modul. Außerdem lässt sich das Protokoll auch sehr einfach in Software programmieren, was jedoch sehr ineffizient ist. SPI benutzt vier parallele Leitungen, was im Vergleich zu ADCs, selbst bei moderater Auflösung, wenig ist. Ein weiterer Vorteil von SPI ist, dass es eine Menge verschiedener Sensorhersteller mit Sensoren unterschiedlicher Messgrößen und Auflösungen gibt. Zu den klassischen Vertretern dieser Sensoren gehören Temperatur-, Druck- oder Beschleunigungssensoren. Dabei lässt sich insbesondere die Temperatur eines Sensors unter Laborbedingungen leicht beeinflussen. Aufgrund der genannten Vorteile soll in dieser Arbeit die Evaluierung anhand eines SPI Temperatursensors erfolgen.

Abhängig von der Art des Sensors sind die ermittelten Sensorwerte nur bedingt von außen steuer- oder kontrollierbar. Dies trifft auch auf einen Temperatursensor zu. Mit einem entsprechenden Klimaschrank ließe sich die Temperatur zwar regulieren, durch die Trägheit der Temperaturänderung, wäre die Flexibilität bei der Messung jedoch stark eingeschränkt. Zur Untersuchung des betrachteten Szenarios ist der Energieverbrauch des Sensors von nachrangiger Bedeutung. Denn sowohl die MCU als auch das autonome, rekonfigurierbare Modul müssen den Sensor ansprechen und den Messwert auslesen. Damit ändert der Sensor die Energiebilanz des Gesamtsystems bei beiden Ansätzen in gleichem Maße und kann daher herausgerechnet werden. In dieser Arbeit wird daher ein anderer Ansatz verfolgt. Ausgehend von einem realen Sensor mit realem Protokoll soll ein Emulator geschaffen werden, der nach Vorgabe beliebige Werte bereit stellt. Damit kann garantiert werden, dass zur Untersuchung beider Ansätze die gleichen, kontrollierten Bedingungen vorliegen.

Die Anforderung, für die Untersuchung der beiden betrachteten Ansätze, gleiche Bedingungen zu haben ist jedoch nicht auf den Sensor beschränkt. Auch der gesamte Messablauf soll bei gleichen Bedingungen, reproduzierbar durchgeführt werden. Darum wird ein parametrierbares, automatisiertes Messsystem gefordert. Damit ist im Allgemeinen ein höherer Aufwand bei der Entwicklung verbunden. Im Gegenzug erhält man jedoch ein System, welches sich leichter für weitere Testobjekte anpassen lässt. Die Auswertung der Ergebnisse soll ebenfalls automatisiert erfolgen um jeweils die gleiche Methodik der Auswertung zu garantieren. Die Interpretation der Ergebnisse bildet den Abschluss der Evaluierung und soll auf Basis der gewonnenen Ergebnisse vorgenommen werden.

In diesem Abschnitt wurde erläutert wie der Testchip als Umsetzung eines autonomen, rekonfigurierbaren Moduls eingesetzt und den MCUs gegenübergestellt wird. Des Weiteren wurde ausgeführt, warum SPI mit einem Temperatursensor als Anwendungsfall des Sensorinterface ausgewählt wurde. Im folgenden Abschnitt 3.3 wird der Zyklus der Sensormessung eingehend erläutert. Außerdem wird genauer definiert, welche Beitrage des Energieverbrauches zum Verbrauch pro Sensormessung beitragen. Abschnitt 3.4 geht auf die variablen Parameter und deren Einfluss auf den Energieverbrauch ein. Aus den möglichen Parameter werden die tatsächlich variierten nochmals hervorgehoben. In Abschnitt 3.5 folgen weitere Überlegungen, die zur Berechnung des Energieverbrauches pro Sensormessung notwendig sind. In Abschnitt 3.6 werden drei Möglichkeiten zur Berechnung des Stromverbrauches pro Sensormessung vorgestellt, aus welchen der Energieverbrauch berechnet werden kann. Den Abschluss bildet Abschnitt 3.7 mit der Bewertung der berechneten Energieverbrauchswerte.

3.3 Für Evaluierung notwendige Ergebnisse

In diesem Abschnitt wird der Zyklus der Sensormessung ausführlich erläutert. Des Weiteren wird auf die relevanten Komponenten eingegangen, die einen Beitrag zum Energieverbrauch pro Sensormessung liefern.

Wie bereits in Abschnitt 3.1 ausgeführt, wird die Evaluierung anhand des Energieverbrauches pro Sensormessung durchgeführt. Die Sensormessung wird zyklisch ausgeführt und wiederholt sich nach t_C. Die Zykluszeit wird bei den MCUs von einem internen Timer, welcher mit dem Takt der RTC betrieben wird, festgelegt. Beim Testchip erfolgt die zeitliche Steuerung mit einem Zählerregister, welches mit der Core-Taktfrequenz betrieben wird. Um den Energieverbrauch zu senken, wird dabei eine Core-Taktfrequenz in der Größenordnung des MCU RTC-Taktes verwendet.

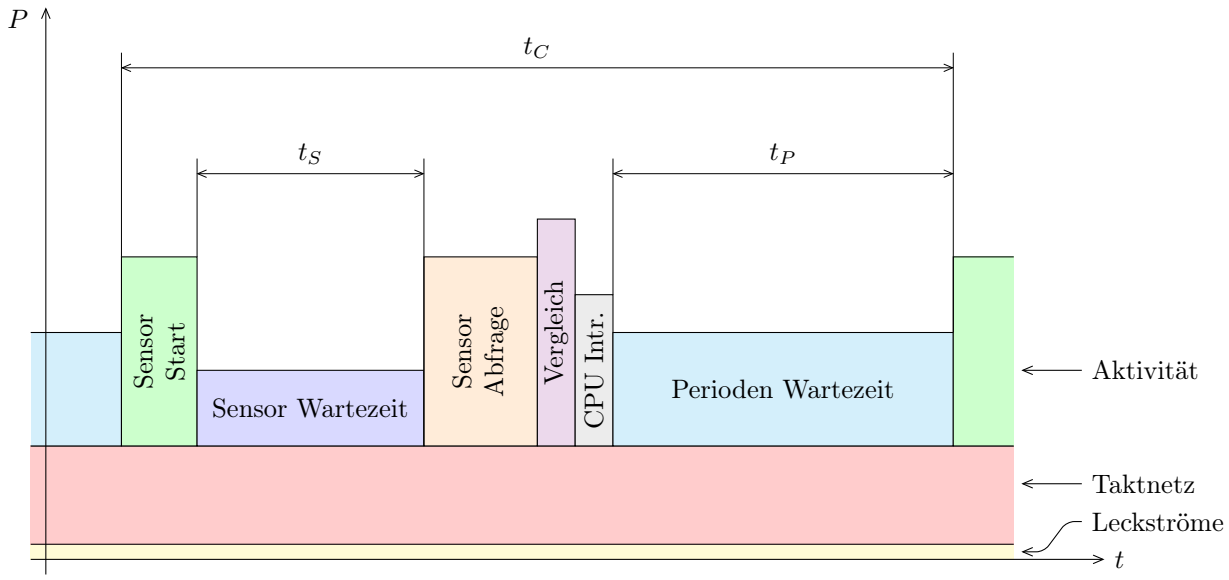


Abbildung 3.1: Messzyklus einer Sensormessung mit Leistungsaufnahme [Gla15, Figure 5.9, S. 137, mit Änderungen]

Die Abbildung 3.1 zeigt einen Zyklus dieser Sensormessung. Der Zyklus beginnt mit dem „Sensor Start“. Dabei wird der Mikrocontroller aus dem LPM in den aktiven Betrieb versetzt. In weiterer Folge müssen sowohl MCU als auch Testchip den externen Sensor aktivieren. Anschließend wird die Sensormessung getriggert. Im Szenario mit SPI-Temperatursensor erfolgt das mit den zwei Kommando Bytes „0x08 0x20“. Anschließend wird die MCU wieder in den LPM versetzt. Danach müssen die Devices die Analog-Digital-Wandlung abwarten, siehe „Sensor Wartezeit“ t_S. Anschließend folgt die „Sensor Abfrage“. Bevor der Sensorwert abgefragt werden kann, muss jedoch die MCU wieder vom LPM in den aktiven Betriebszustand versetzt werden. Dann wird der Sensorwert abgefragt. Der SPI-Temperatursensor erwartet dazu das Kommando Byte „0x50“ und zwei weitere Dummy-Bytes, die lediglich den SPI-Transfer initiieren. Im Anschluss sind Schiebeoperationen notwendig, da der Sensorwert nur 13 Bit hat und am höchstwertigen Bit ausgerichtet ist. Danach erfolgt bei allen Devices der „Vergleich“ des aktuellen Sensorwertes mit dem zuletzt gespeicherten Wert. Dabei wird überprüft, ob die Differenz beider Werte eine zuvor definierte Schaltschwelle überschreitet. Wird die Schaltschwelle überschritten, wird der neue Wert gespeichert und der

„CPU Interrupt“ ausgelöst. Die Weiterverarbeitung oder Übertragung des Sensorwertes würde in zu diesem Zeitpunkt erfolgen, wird in dieser Arbeit jedoch nicht näher betrachtet. In weiterer Folge wird die MCU wieder in den LPM versetzt. Abschließend warten MCU und Testchip die „Perioden Wartezeit“ t_{P} ab, bevor ein neues Zyklus startet. Die Perioden Wartezeit muss von den Devices entsprechend umgesetzt werden, so dass in Summe die Zykluszeit erreicht wird.

Der gesamte Energieverbrauch ist dabei die Produktsumme der dargestellten Beiträge im Zeitraum t_{C}. Nach der Definition des Energieverbrauches pro Sensormessung sind nicht alle Beiträge relevant. Die benötigten Komponenten sind:

- Sensor Start
- Sensor Abfrage
- Vergleich
- CPU Interrupt

Die Summe der Komponenten lässt sich mit multipler linearer Regression berechnen. Dazu sind viele Einzelmessungen und die gezielte Variation der Parameter wie Versorgungsspannung und Frequenz notwendig. Diese Einzelmessung bilden im Weiteren die Grundlage für die Berechnung von E_{MCU} und E_{TC}, wie in Abschnitt 3.6 beschrieben. Das Taktnetz ist bei MCUs während der Wartezeiten t_{S} und t_{P} abgeschaltet. Zur Einfachheit der Darstellung wird der Beitrag des Taktnetzes in Abbildung 3.1 konstant dargestellt.

Die formale Forderung nach gleichen Randbedingungen lässt sich strenger als Reproduzierbarkeit auslegen. Die Basis dafür bildet ein automatisiertes Messsystem. Dieses muss jedoch parametrierbar gestaltet sein, um den gesamten Betriebsbereich des einzelnen Prüflings messen zu können. Eine weitergehende Analyse der beeinflussbaren Parameter wird im Folgenden im nächsten Abschnitt diskutiert.

3.4 Messraum (abhängige und unabhängige Variablen)

Die Evaluierung kann als wissenschaftliches Experiment betrachtet werden, bei welchem es eine Menge Eingangsgrößen und eine Menge Ausgangsgrößen gibt. Die Eingangsgrößen werden dabei als unabhängige Variablen bezeichnet, die Ausgangsgrößen als abhängige Variablen. Dieser Abschnitt befasst sich mit der Analyse, welche Veränderung der abhängigen Variablen bei Änderung der unabhängigen Variablen zu erwarten sind. Des weiteren wird die Kombination der unabhängigen Variablen zu möglichen Testfällen näher betrachtet und eine Optimierungsstrategie vorgestellt.

Zu den abhängigen Variablen zählt unmittelbar der messbare Stromverbrauch des untersuchten Prüflings. Wesentlichen Einfluss auf den Stromverbrauch hat die gewählte Versorgungsspannung. Des weiteren wird der Stromverbrauch von der aktiven Zeit bestimmt. Eine weitere abhängige Variable ist der Energieverbrauch, welcher sich aus einer Menge von Strommessungen berechnen lässt.

In der folgenden Aufzählung werden die unabhängigen Variablen und ihr erwarteter Einfluss eingehender betrachtet.

- Eine grundlegende Einflussgröße wird hier der Vollständigkeit wegen nochmals mit aufgenommen. Die Betrachtung des SoC mit Mikrocontroller und **mit und ohne Testchip**. Wie die Analyse in [GHDG09] zeigt, ist mit Testchip eine Verringerung des Energieverbrauches zu erwarten. Aufgrund des gewünschten Vergleiches zwischen MCU und Testchip, wird jeweils der Stromverbrauch der Mikrocontroller und des Testchips alleine in gesonderten Messungen bestimmt.

- Einen Einfluss hat ebenfalls der verwendete **Mikrocontroller** im Betrieb ohne Testchip. Für diese Arbeit wurden 5 Mikrocontroller ausgewählt, welche laut jeweiligem Hersteller für Ultra-Low-Power konzipiert sind. Diese Mikrocontroller werden in Abschnitt 4.6 genauer vorgestellt. Zu erwarten ist, dass der Energieverbrauch der einzelnen Typen wesentlich vom jeweiligen Betriebspunkt abhängt, womit keine generelle Vorhersage getroffen werden kann.

- Der zur Messung verwendete **Sensor** hat ebenfalls Einfluss auf den Energieverbrauch. Das benutzte Protokoll wirkt sich auf die aktive Zeit des Mikrocontrollers aus. Je mehr einzelne Informationen für das vom Sensor verwendete Protokoll übertragen werden müssen, desto länger ist auch die Übertragungszeit. Im betrachteten Fall werden alle Mikrocontroller mit dem gleichen Sensor oder Emulator gemessen, womit nur die unterstützte Übertragungsgeschwindigkeit des jeweiligen Mikrocontrollers in das Ergebnis eingeht. Diese Übertragungsgeschwindigkeit wird in weiterer Folge Busgeschwindigkeit genannt.

- Die **Busgeschwindigkeit** des Sensorinterface wirkt sich ebenfalls auf die aktive Zeit des Mikrocontrollers aus. Grundsätzlich ist bei höchstmöglicher Übertragungsgeschwindigkeit, welche entweder durch den Sensor oder den Mikrocontroller vorgegeben wird, die geringste aktive Zeit und damit der geringste Energieverbrauch zu erwarten. Es wird die jeweils höchste unterstützte Busgeschwindigkeit für den Vergleich herangezogen. Um den Einfluss der Busgeschwindigkeit auf den Energieverbrauch beurteilen zu können, sollen zusätzlich weitere, langsamere Busgeschwindigkeiten untersucht werden. In der Praxis wird bei SPI die Busgeschwindigkeit durch den Busteiler definiert. Dieser bestimmt das Teilungsverhältnis von Systemtakt zu Bustakt und beträgt bei synchronen Schaltungen mindestens 2. Aufgrund der Frequenzunabhängigkeit wird in dieser Arbeit der Busteiler als Referenzgröße für die Busgeschwindigkeit herangezogen.

- Die **Spannung des Testchips** ist eine Versorgungsspannung und wirkt sich sowohl auf den Stromverbrauch, als auch nach Gleichung (3.2) direkt auf den Energieverbrauch aus. Eine Reduktion der Spannung verringert den Energieverbrauch.

- Die **Spannung des Mikrocontrollers** ist analog zur Spannung des Testchips eine Versorgungsspannung. Das bedeutet, dass eine Verringerung der Spannung den Energieverbrauch senkt.

- Die **Taktfrequenz** mit welcher der Testchip und die MCUs betrieben werden, beeinflussen den Stromverbrauch der Chips. Mit steigender Frequenz werden im selben Zeitraum mehr Umladevorgänge der Transistorkapazitäten durchgeführt und der Stromverbrauch steigt an, siehe Gleichung (2.2). Bei den MCUs steigt im Gegenzug die Ausführungsgeschwindigkeit an und die aktive Zeit reduziert sich. Wenn die Totzeiten der Interruptroutinen vernachlässigt werden, heben sich die Effekte gegenseitig auf und es gibt keine Änderung des durchschnittlichen Stromverbrauches eines Zyklus bei den MCUs. Dadurch, dass der Testchip permanent läuft, profitiert er nicht von der schnelleren Ausführungsgeschwindigkeit und der Stromverbrauch steigt mit steigender Frequenz an.

- Die **Taktfrequenz der RTC** ist nur für die MCUs ein Parameter. Diese Frequenz hat große praktische Bedeutung, da sie die zyklische Aktivierung der MCUs steuert. Meist wird eine Frequenz von 32,768 kHz verwendet, da der Überlauf eines 15-Bit Zählers bei dieser Frequenz nach genau einer Sekunde passiert. Der Einfluss auf den Energieverbrauch ist nur im Standby nennenswert. In dieser Arbeit wird die Frequenz der RTC fest mit 32,768 kHz gewählt.

- Die **Zykluszeit der Sensormessung** t_C bestimmt die Häufigkeit der Messung pro Sekunde. Sie ist nach unten durch die Wartezeit zur Analog-Digital-Wandlung des Sensors und nach oben durch den 16-Bit Zähler der MCUs begrenzt. Durch die Verwendung des Sensor-Emulators lassen sich jedoch auch kleinere Zykluszeiten als beim physischen Vorbild einsetzen. Eine Reduktion der Zykluszeit erhöht die Häufigkeit der Sensormessung und damit auch die aktive Zeit der MCUs. Dadurch steigt der Strom- und der Energieverbrauch im untersuchten Zeitintervall an. Durch die in dieser Arbeit gewählte Definition des „Energieverbrauches pro Sensormessung“ hat die Änderung der Zykluszeit keinen Einfluss auf das Ergebnis.

- Wenn der Absolutbetrag der Differenz zwischen aktuellem und zuletzt gespeichertem Sensorwert die **Schaltschwelle** (engl. Threshold) überschreitet wird der aktuelle Sensorwert gespeichert und die Weiterverarbeitung oder Übertragung getriggert. Wenn die Schaltschwelle überschritten wird, verlängert sich die aktive Zeit der MCU. Der Einfluss der Schaltschwelle ist stark vom jeweiligen Messwert und der Dynamik der Messgröße abhängig. Generell lässt sich nur festhalten, dass kleinere Schaltschwellen öfter eine Weiterverarbeitung triggern werden als große Schaltschwellen. Die Weiterverarbeitung erfolgt jedenfalls unabhängig von der Implementierung mit oder ohne Testchip in der MCU. Darum werden nur der Messwerte-Vergleich und das allfällige Speichern des neuen Messwertes in die Untersuchung mit aufgenommen. Tabelle 3.1 zeigt den für die praktische Messung gewünschten Zusammenhang zwischen der Häufigkeit des Triggers und der für die Messung gewählten Schaltschwelle. Im folgenden Punkt wird auf die dazu benötigten Messwerte näher eingegangen.

Häufigkeit des Triggers	10	9	8	7	6	5	4	3	2	1	0
Schaltschwelle	0	1	2	3	4	5	6	7	8	9	100

Tabelle 3.1: Zusammenhang der Häufigkeit des Triggers und der Schaltschwelle bei 10 Messungen

- Die **Werte-Bibliothek** des Sensor-Emulators hat Einfluss auf die Häufigkeit des Triggers für die Weiterverarbeitung. Die fiktiven Sensorwerte können wie folgt konstruiert werden, um den Anforderungen aus Tabelle 3.1 zu entsprechen. Es wird die Reihe bei einem beliebigen Zahlenwert gestartet und anschließend die jeweils nächste Schaltschwelle (außer Null) dazu addiert. Wenn die Reihe bei 1 startet ergeben sich die Zahlenwerte wie in Tabelle 3.2 angegeben. Wenn die Sensorwerte von 1 bis 46 abgerufen wurden, wird wieder bei 1 gestartet. Die hohe Schaltschwelle von 100 in Tabelle 3.1 wurde bewusst gewählt, damit sich der Wert im Messaufbau klar abhebt.

Sensorwert	1	2	4	7	11	16	22	29	37	46
Delta	+1	+2	+3	+4	+5	+6	+7	+8	+9	

Tabelle 3.2: Werte-Bibliothek des Sensor-Emulators

- Die Temperatur des Halbleiters stellt ebenfalls eine Einflussgröße dar. Dabei sind CMOS-Schaltungen geringerem Einfluss als Bipolar-Schaltungen unterworfen, aber dennoch temperaturabhängig [TS02, S. 191f]. Für die Praxis bedeutet das, dass die Temperatur in der Messumgebung für vergleichbare Ergebnisse konstant sein soll.

Durch die Kombination aller unabhängigen Variablen ergeben sich die möglichen Testfälle für das jeweilige DuT. Als DuT werden die einzelne MCU oder der Testchip gesehen, für welche jeweils eine Messreihe aufgenommen wird. Die Tabelle 3.3 zeigt praxisnahe Werte für die unabhängigen Variablen, welche auch für die praktische Messung der MCUs verwendet wurden. Die Spannung wird von 1,8 V bis 3,3 V variiert und ist den Datenblättern der MCUs entnommen. Die Taktfrequenz wird von 0 bis 10 MHz in 1 MHz Schritten vorgegeben. Dabei entspricht eine Taktfrequenz von 0 einem deaktivierten Takt und 1 MHz stellt den aktiven Punkt mit der geringsten Frequenz der MCUs dar. Die RTC Frequenz wird nur zwischen 0 und 32,768 kHz geändert, die RTC also nur ein- oder ausgeschaltet. Als Zykluszeiten werden Werte gewählt, die sich nach Division von 200 ms durch die Anzahl Sensormessungen ergeben. 200 ms sind dabei die Integrationszeit der Messung, auf welche in Abschnitt 4.3 genau eingegangen wird. 0 stellt eine Zykluszeit von einem DuT im Dauerbetrieb, also ohne Schlafphasen und eine Zykluszeit von ∞ ein DuT ohne aktive Phase dar. Der Busteiler wird als ganzzahlige Vielfache von zwei gewählt, wobei zwei den kleinsten möglichen Wert in synchronen Schaltungen darstellt. Die Schaltschwellen werden direkt aus Tabelle 3.1 übernommen. Bei der Kombination der unabhängigen Variablen ist auf ungültige Kombinationen zu achten. Denn abhängig vom untersuchten DuT geben Hersteller einen jeweils sicheren Betriebsbereich bezogen auf Versorgungsspannung und Taktfrequenz an. Insbesondere Spannungen am unteren Ende des Betriebsbereiches lassen nicht immer die maximale Taktfrequenz des Chips zu. Um Fehlmessungen zu vermeiden, werden diese ungültigen Punkte bereits bei der Erstellung der Testfälle ausgeschlossen.

unabhängige Variable	Wertebereich und Schrittweite	Anzahl Messpunkte
Spannung	1,8 V ... 3,3 V in 0,1 V Schritten	16
Taktfrequenz	0 ... 10 MHz in 1 MHz Schritten	11
RTC Frequenz	[0, 32,768] kHz	2
Zykluszeit	[0, 5, 10, 20, 40, 50, 100, 200, ∞] ms	9
Busteiler	[2, 4, 8]	3
Schaltschwelle	[1, 2, 4, 7, 11, 16, 22, 29, 37, 46]	10

Tabelle 3.3: Unabhängige Variablen und Wertebereich bei MCUs

Durch die Kombination der in Tabelle 3.3 angegebenen Anzahl Messpunkte und ohne Berücksichtigung der zuvor erwähnten ungültigen Betriebsbereiche und Optimierung wären in Summe 95040 Testfälle zu messen. Der Strommessung wird eine Integrationszeit von 200 ms zugrunde gelegt. Um Ausreißer statistisch ermitteln und ausschließen zu können werden 50 Einzelmessungen pro Testfall geplant. Daraus ergibt sich eine Messzeit pro Testfall von 10 s. Für Konfiguration der Messumgebung und Parametrierung des DuT wird eine Setup-Zeit von 5 s pro Testfall angenommen. Damit benötigt die Messung eines kompletten Testfalles 15 s. Auf die Anzahl der Testfälle hoch gerechnet, beträgt die gesamte Messdauer 396 Stunden oder umgerechnet 16,5 Tage. Darum wird eine Reduktion der Anzahl der Testfälle angestrebt.

Das Ziel ist dabei, die Anzahl der Testfälle zu reduzieren und gleichzeitig die Aussagekraft zu erhalten. Erreicht wird dies einerseits durch gezielte Vermeidung von Redundanzen und andererseits durch Vergröberung des Rasters. Da durch die theoretische Analyse des Einflusses der

Parameter wie Zykluszeit, Busteiler und Schaltschwelle bereits Vorzugswerte bekannt sind, kann die Messung bei den Vorzugswerten durchgeführt werden und jeweils nur ein Parameter verändert werden. Außerdem wird die Aussagekraft der Messung durch Variation dieser Parameter bei alle Spannungs- und Frequenz-Kombinationen nicht erhöht. Des Weiteren ist aus praktischer Sicht die Variation der Taktfrequenz bei deaktiviertem RTC Takt nicht sinnvoll. Eine Charakterisierung des DuT kann auch bei gröberem Raster erfolgen, wenn sowohl Grenzbereiche als auch typische Werte berücksichtigt werden. In diesem Sinne werden Messreihen mit minimaler, maximaler und einer Spannung im Mittel der beiden Extrema über alle Frequenzpunkte untersucht. Analog gilt diese Überlegung auch für die Frequenz. In der Praxis ergibt sich dann abhängig vom Betriebsbereich des jeweiligen DuT ein Satz von 200 bis 300 Testfällen mit einer gesamten Messdauer von maximal 3 Stunden. Umgesetzt wird die Reduktion der Anzahl der Testfälle mit einem Testfallgenerator, welcher systematisch die Testfälle erzeugt und in Abschnitt 4.5 durch Abbildung 4.7 genau beschrieben wird.

In diesem Abschnitt wurden die Einflussgrößen der Messung aufgezählt und der jeweilige Einfluss auf den Stromverbrauch abgeschätzt. Des Weiteren wurde auf die in der Messung variierten Größen und deren Wertebereiche eingegangen. Abschließend wurde die Notwendigkeit einer Reduktion der Testfälle erläutert.

3.5 Messtechnische Erfassung

In diesem Abschnitt werden grundlegende Überlegungen zur Wahl der Integrationszeit der Strommessung angestellt. Außerdem wird die Notwendigkeit der Synchronisation der Messung erklärt. Des Weiteren werden drei Möglichkeiten zur Berechnung des Energieverbrauches pro Sensormessung und die dazu notwendigen Messergebnisse vorgestellt.

Damit das Messergebnis nicht durch 50 Hz Netzeinkopplungen verfälscht wird, müssen weitere Maßnahmen bedacht werden. Durch die Wahl der Integrationszeit als Vielfaches der Netzperiode wird die Störung eliminiert. Hochwertige Messgeräte bieten direkt eine entsprechende Option, unter anderem als Number of Power Line Cycles (NPLC) bekannt, mit welcher die Integrationszeit auf die Netzperiode abgestimmt wird. Die Wahl der Integrationszeit und der NPLC wird in Abschnitt 4.3 eingehender besprochen.

Um die Berechnung des Verbrauchs eines Messevents zu ermöglichen, muss die Integrationszeit $T = 200\,\text{ms}$ ein Vielfaches der Zykluszeit t_C der Sensormessung sein, siehe Abschnitt 3.4. Wenn beide Zeiten bekannt sind, lässt sich die Anzahl der Messevents und damit der Stromverbrauch eines Messevents berechnen. Durch eine möglichst große Anzahl an Messevents wird die Auflösung der Messung vergrößert.

Zusätzlich zur Abstimmung der Integrationszeit, ist eine Synchronisation zwischen Sensormessung und Messumgebung notwendig. Aufgrund unterschiedlicher Taktdomänen in MCU, Testchip und Messgeräten kann keine Synchronizität vorausgesetzt werden. Des Weiteren müssen auch die Taktgeber als nicht perfekt angenommen werden, womit immer geringfügige Abweichungen in den Zeitperioden auftreten können. Ohne Synchronisation können somit einzelne Sensormessungen aus dem Messfenster rutschen. Insbesondere bei hohen Taktfrequenzen der MCU ist die aktive Zeit der MCU kurz und damit die Wahrscheinlichkeit eines Messfehlers erhöht. Außerdem ist gerade bei hohen Taktfrequenzen der Stromverbrauch höher und der Fehler wirkt sich stärker auf das Messergebnis aus.

Die Takterzeugung wird in der Messumgebung vorgesehen und die internen Oszillatoren abgeschaltet. Dadurch liefern diese keinen Beitrag zum Stromverbrauch. Des Weiteren hat die externe Takterzeugung noch einen zusätzlichen, praktischen Vorzug. Die MCUs und der Testchip können bei gleicher Konfiguration mit unterschiedlichen Frequenzen gemessen werden.

Zur Berechnung des Energieverbrauches pro Sensormessung muss die Summe der nach Abschnitt 3.3 relevanten Komponenten berechnet werden. Für diese Arbeit wurden 3 Berechnungsmöglichkeiten in Betracht gezogen:

- Die erste Berechnungsmöglichkeit basiert auf der Modellbildung des Stromverbrauches der Chips kombiniert mit den Parametern die in Abschnitt 3.4 identifiziert wurden. Anschließend folgt die Berechnung der Koeffizienten des Modells durch multiple lineare Regression. Anhand des modellierten Stromverbrauches lassen sich die Stromwerte pro Sensormessung berechnen und daraus der Energieverbrauch. Der Vorteil dieser Methode ist, dass die Devices durch das Modell gut analysiert werden werden können, ein Nachteil ist jedoch, dass die Treffsicherheit des Modells nicht im Vorfeld bestimmt werden kann.

- Eine weitere Berechnungsmöglichkeit setzt Anpassungen an den DuT-Programmen voraus. Durch zusätzliche Initialisierungsroutinen können die Prüflinge von der Messumgebung parametriert werden. Damit lässt sich der Ruhestrom des Chips direkt messen. Dazu ist lediglich die Zykluszeit als unendlich zu definieren und keine weitere Anpassung an der Sensormessroutine notwendig, welche das Ergebnis verfälschen würde. An dieser Methode vorteilhaft ist die einfache Berechnung des Strom- und Energieverbrauches pro Sensormessung. Ein Nachteil ist jedoch, dass prinzipbedingt der Energieverbrauch des Testchip damit nicht berechnet werden kann, da der Testchip zur Messung des Ruhestromes umkonfiguriert werden müsste, was wiederum das Ergebnis verfälschen würde.

- Die letzte betrachtete Berechnungsmöglichkeit basiert auf der Variation der Häufigkeit der Sensormessung für einen Messpunkt. Das bedeutet, dass lediglich die Zykluszeit variiert wird während die weiteren Parameter konstant gehalten werden. Die Berechnung des Stromverbrauches erfolgt dann wiederum durch lineare Regression, allerdings bei festen Spannungs- und Frequenzwerten. Vorteilhaft an dieser Methode ist, dass sie pro Berechnungspunkt ebenfalls nur zwei Messpunkte, wie Methode zwei, benötigt. Ein weiterer Vorteil besteht in dem einfacheren Modell gegenüber Methode eins und damit einer höheren Treffsicherheit der Modellierung.

In dieser Arbeit wurden alle drei Berechnungsmöglichkeiten entwickelt und im folgenden Abschnitt 3.6 ausführlich erklärt. Anhand der praktischen Messung wurde jedoch festgestellt, dass für die Mikrocontroller nur die zweite Berechnungsmethode richtige Werte liefert, während für die Berechnung beim Testchip die dritte Methode angewandt werden musste. Die genauen Hintergründe dazu sind in den Kapiteln 5 und 6 angeführt.

In diesem Abschnitt wurde der Zusammenhang zwischen Integrationszeit der Messung und der Zykluszeit sowie der Bedarf einer Synchronisation der Messung erläutert. Außerdem wurden drei Berechnungsmöglichkeiten zur Berechnung des Energieverbrauches pro Sensormessung vorgestellt, welche im folgenden Abschnitt näher erklärt werden.

3.6 Berechnung des Strom-/Energieverbrauches

Die im vorhergehenden Abschnitt vorgestellten Berechnungsmöglichkeiten des Energieverbrauches pro Sensormessung werden in diesem Abschnitt hergeleitet und ausführlich erklärt. Dabei wird die Berechnung in zwei Schritten über den Stromverbrauch pro Sensormessung entwickelt. In Abschnitt 3.4 wurden die Einflussgrößen und ihre jeweilige Auswirkung betrachtet. Die Tabelle 3.4 fasst die für die Berechnung verwendeten Formelzeichen der Einfluss- und Ergebnisgrößen zusammen.

gegebene Eingangsgrößen	
U	Versorgungsspannung
f	Taktfrequenz
n	Anz. Sensormessungen pro Integrationszeitraum T
m	Anz. Schwellwertüberschreitungen pro Integrationszeitraum T
d	Teilerverhältnis SPI
T	Integrationszeitraum der Strommessung
t_C	Zykluszeit einer Sensormessung
gemessene Ausgangsgröße	
I_T	Strom im Integrationszeitraum T
I_0	Ruhestrom im Integrationszeitraum T
t_a	Zeitdauer einer aktiven Sensormessung
berechnete Ausgangsgröße	
I_SM	Strom pro Sensormessung
E_SM	Energie pro Sensormessung

Tabelle 3.4: Zusammenfassung der verwendeten Formelzeichen

Die Berechnung des Energieverbrauches pro Sensormessung erfolgt in dieser Arbeit in zwei Schritten:

- Im ersten Schritt erfolgt die **Berechnung des Stromverbrauches pro Sensormessung**. In Abschnitt 3.5 wurden dazu bereits drei Berechnungsmöglichkeiten kurz vorgestellt. Allen drei Methoden gemein ist die Trennung in statischen und dynamischen Verbrauch wie in Gleichung (3.5). Gleichung (3.3) beschreibt den tatsächlich gemessenen Strom im Integrationszeitraum T.

$$I_\mathrm{T} = \frac{1}{T}\int_0^T i(t)\mathrm{d}t \tag{3.3}$$

  Unter Berücksichtigung der in Abschnitt 3.3 angestellten Überlegungen, kann I_T wie folgt angeschrieben werden.

$$I_\mathrm{T} = \frac{1}{T}\int_0^T i_0(t) + i_\mathrm{nSM}(t)\mathrm{d}t \tag{3.4}$$

  Der Strom $i_0(t)$ stellt dabei den statischen Anteil dar, während $i_\mathrm{nSM}(t)$ den zeitlich veränderlichen Strom bei n Sensormessungen darstellt. Als Resultat der Integration folgt die Gleichung (3.5).

$$I_\mathrm{T} = I_0 + n \cdot I_\mathrm{SM} \tag{3.5}$$

- Im zweiten Schritt erfolgt die eigentliche **Berechnung des Energieverbrauches pro Sensormessung**. Die elektrische Energie ist als Integral über dem Produkt von Spannung und Strom, wie in Gleichung (3.2) definiert und wird hier nochmals angeschrieben.

$$E = \int_0^T u(t) \cdot i(t) \mathrm{d}t \tag{3.6}$$

  Unter Laborbedingungen mit einer geregelten Spannungsquelle kann die Versorgungsspannung als konstant angenommen werden. Damit kann die Spannung U vor das Integral gezogen werden und folgende Gleichung ergibt sich.

$$E = U \cdot \int_0^T i(t) \mathrm{d}t \tag{3.7}$$

  Nach Ersetzen des Stromintegrals durch Gleichung (3.3) und Gleichung (3.5) lässt sich der Energieverbrauch pro Sensormessung einfach isolieren und folgendermaßen anschreiben.

$$E_{\mathrm{SM}} = U \cdot I_{\mathrm{SM}} \cdot T \tag{3.8}$$

  Zu beachten ist, dass das Zeitintervall der Energieberechnung nicht gleich der aktiven Zeit einer Sensormessung t_{a} ist, sondern dem Integrationsintervall der Strommessung T entspricht.

Auf jede der drei erwähnten Berechnungsmethoden soll nun genauer eingegangen werden. Die erste betrachtete Berechnungsmethode ist eine **Berechnung nach Modell** basierend auf multipler linearer Regression. Wesentliche Grundlage ist dabei die Gestaltung des Modells. Ausgangspunkt ist hierbei die Gleichung (2.1), welche die Komponenten der Verlustleistung in Halbleitern beschreibt und hier nochmals angeführt wird.

$$P_{\mathrm{tot}} = P_{\mathrm{switch}} + P_{\mathrm{sc}} + P_{\mathrm{stat}} \tag{3.9}$$

Die Gleichungen (3.10) [vgl. Hit07, S. 44, Gl. 1.14] und (3.11) [vgl. Hit07, S. 32, Gl. 1.4] geben näherungsweise die dynamischen Verluste wieder. K_1 und K_2 sind Koeffizienten, welche die Faktoren der Originalgleichungen zusammenfassen, da die Originalwerte in dieser Betrachtung von untergeordneter Bedeutung sind.

$$P_{\mathrm{switch}} = K_1 \cdot U^2 f \tag{3.10}$$

$$P_{\mathrm{sc}} = K_2 \cdot U^3 f \tag{3.11}$$

Die statische Verlustleistung wird mit der parasitären Diode, die sich bei der Integration von CMOS-Schaltungen ergibt, modelliert. Grundlage dazu ist die Diodengleichung [vgl. Hit07, S. 34, Gl. 1.5].

$$\begin{aligned} I_{\mathrm{DL}} &= I_{\mathrm{S}} \cdot \left(e^{\frac{U_{\mathrm{D}}}{U_{\mathrm{th}}}} - 1 \right) \\ \text{mit} \quad I_{\mathrm{DL}} &= \text{Dioden-Leakage Strom} \\ I_{\mathrm{S}} &= \text{Rückwärtssättigungsstrom} \\ U_{\mathrm{D}} &= \text{Drain-Spannung} \\ U_{\mathrm{th}} &= \text{thermische Spannung} \end{aligned} \tag{3.12}$$

Die Diodengleichung wird durch eine Taylorreihen-Entwicklung bis zum linearen Term linearisiert. K_3 und K_4 sind die Koeffizienten der Terme Nullter und Erster Ordnung.

$$I_{\mathrm{DL}} = K_3 + K_4 \cdot U \tag{3.13}$$

Unter Annahme einer konstanten Spannung werden die Terme für die dynamischen Verluste durch die Spannung dividiert und zusammen mit dem modellierten statischen Strom der Summenstrom gebildet. Dabei stellt K_1 den Koeffizienten für den Strom durch die Umladung der Transistorkapazitäten dar. K_2 ist der Koeffizient für den Kurzschlussstrom. K_3 und K_4 sind die Koeffizienten für statischen Verluststrom.

$$I_{\text{tot}} = K_1 \cdot Uf + K_2 \cdot U^2 f + K_3 + K_4 \cdot U \tag{3.14}$$

Die Gleichung (3.14) bildet die Grundlage für die Modellgleichung, wobei die Koeffizienten K durch die Modellparameter b ersetzt werden. Da die Umladeverlustleistung wesentlich von der aktiven Zeit des DuT abhängt, wird der Koeffizient K_1 feiner mit Einfluss eines allgemeinen Terms, der Anzahl der Sensormessungen n, der Anzahl der Schwellwertüberschreitungen m und des SPI-Busteilers d modelliert.

$$K_1 = b_3 + b_4 \cdot n + b_5 \cdot m + b_6 \cdot d \tag{3.15}$$

Außerdem werden verbliebenen Koeffizienten mit jeweils einem Modellparameter wie folgt identifiziert.

$$K_2 = b_7 \tag{3.16}$$
$$K_3 = b_1 \tag{3.17}$$
$$K_4 = b_2 \tag{3.18}$$

Aufgrund der Dominanz der Umladeverlustleistung werden die Parameter n, m und d nicht im Kurzschlussstrom modelliert. Nach Ersetzen von K_1 bis K_4 in Gleichung (3.14) und sortieren der Terme nach steigenden Potenzen wird die Modellgleichung wie folgt angeschrieben.

$$I_{\text{T}} = I_{\text{tot}} = b_1 + b_2 \cdot U + b_3 \cdot Uf + b_4 \cdot Uf \cdot n + b_5 \cdot Uf \cdot m + b_6 \cdot Uf \cdot d + b_7 \cdot Uf^2 \tag{3.19}$$

Neben der Charakterisierung des Halbleiterchips beschreibt das Modell mit dem Koeffizienten b_4 den gesuchten Stromverbrauch pro Sensormessung in Abhängigkeit von Spannung und Frequenz.

$$I_{SM} = b_4 \cdot Uf \tag{3.20}$$

Die zweite Berechnungsmethode beruht auf der **Berechnung nach Messung von Aktiv- und Ruhestrom**. Dazu muss zu jedem aktiven Betriebspunkt des DuT auch der entsprechende Betriebspunkt im LPM gemessen werden, was eine Verdopplung der Messpunkte bedeutet. Grundlage der Berechnung bildet die bereits bekannte Gleichung (3.5), die hier nochmals angeschrieben wird.

$$I_{\text{T}} = I_0 + n \cdot I_{\text{SM}} \tag{3.21}$$

Umgeformt ergibt sich daraus Gleichung (3.22) welche direkt zur Berechnung für beliebige Spannungs-Frequenz-Paare herangezogen werden kann.

$$I_{SM} = \frac{(I_{\text{T}} - I_0)}{n} \tag{3.22}$$

Zur Reduktion der Messdauer und Optimierung der Ergebnisse werden in dieser Arbeit Randbedingungen für die Messung und Berechnung nach dieser Methode definiert. Mit $n = n_{\max}$ wird die Anzahl der Sensormessungen auf die maximale Anzahl im Integrationszeitraum festgelegt. Dadurch wird die Auflösung maximal. Die Anzahl der Schwellwertüberschreitungen wird mit $m = 0$

festgelegt, um die aktive Zeit des DuT möglichst kurz zu halten. Der SPI-Busteiler wird mit $d = d_{\min}$ definiert, um eine hohe Übertragungsgeschwindigkeit mit kurzer Übertragungsdauer und dadurch eine kürzere aktive Zeit zu erzielen.

Die letzte mögliche Berechnungsmethode des Stromverbrauchs pro Sensormessung ist die **Berechnung durch lineare Regression**. Entgegen der ersten Methode die ebenfalls auf lineare Regression setzt, ist bei dieser Methode das Modell jedoch wesentlich einfacher. Die Grundlage bildet Gleichung (3.5), welche zugleich auch die Modellgleichung darstellt und hier der Vollständigkeit wegen nochmals angegeben ist.

$$I_{\mathrm{T}} = I_0 + n \cdot I_{\mathrm{SM}} \tag{3.23}$$

Bei der Messung werden für jeden zu bestimmenden Messpunkt Spannung und Frequenz konstant gehalten und lediglich die Häufigkeit der Sensormessung, n, variiert. Dabei sind bereits zwei Messpunkte pro Spannungs-Frequenz-Paar ausreichend. Eine größere Anzahl Messpunkte erlaubt Fehlmessungen zuverlässiger zu identifizieren. Wie bei der zweiten Berechnungsmethode werden wiederum die Anzahl der Schwellwertüberschreitungen $m = 0$ und der SPI-Busteiler gewählt $d = d_{\min}$. Durch numerische Berechnung der linearen Regression über I_{T} und n werden die Koeffizienten I_0 und I_{SM} für die ausgewählten Spannungs-Frequenz-Paare bestimmt.

Aufgrund der Tatsache, dass es zwischen den benötigten Messpunkten der drei Berechnungsmethoden große Überlappungen gibt, wird die Messung so ausgelegt, dass jeweils alle betrachteten Berechnungsmethoden eingesetzt werden können. Wie in den Kapiteln 5 und 6 gezeigt wird, konnte die erste Berechnungsmethode nicht zur Berechnung des Energieverbrauches pro Sensormessung verwendet werden, wurde aber zur Charakterisierung der MCUs verwendet. Die zweite Methode wurde zur Berechnung des Energieverbrauches bei den MCUs und die dritte Methode für den Testchip verwendet.

In diesem Abschnitt wurde die Methodik zur Berechnung des Stromverbrauches pro Sensormessung und in weiterer Folge des Energieverbrauches pro Sensormessung erläutert. Es wurden drei Methoden zur Bestimmung des Stromverbrauches pro Sensormessung vorgestellt, deren benötigte Messdaten in einer gemeinsamen Messung erfasst werden können.

3.7 Bewertung der Ergebnisse

In den vorangegangenen Abschnitten wurde gezeigt, warum der Energieverbrauch pro Sensormessung als Kenngröße verwendet wird und wie diese Kenngröße gemessen und berechnet werden kann. In diesem Abschnitt wird erläutert, welche Überlegungen dazu führen, die Bewertung der Ergebnisse anhand des optimalen Betriebspunktes durchzuführen und welcher Betriebspunkt optimal ist.

Nach der Berechnung des Energieverbrauches pro Sensormessung muss noch die Frage geklärt werden, welcher Ansatz, also mit Mikrocontroller oder mit autonomem, rekonfigurierbarem Modul, energiesparender ist. Dazu müssen zuvor Bedingungen definiert werden, um einen fairen Vergleich zu gewährleisten. Der gleiche Messpunkt für alle Prüflinge, umfassend jeweils die selbe Spannung, Frequenz und ebenfalls die gleichen weiteren Parameter, scheint im ersten Augenblick geeignet zu sein. Dabei darf jedoch nicht übersehen werden, dass bereits die Wahl des Messpunktes das Ergebnis beeinflussen kann und unter anderen Bedingungen ein anderer Prüfling im Vergleich besser abschneidet.

Da in dieser Arbeit keine konkrete Applikation untersucht wird, stehen keine Informationen über Randbedingungen zur Verfügung. Eine solche Randbedingung könnte zum Beispiel die eingesetzte Spannungsquelle sein, welche als Referenz für den Betrieb herangezogen werden könnte. Eine andere Überlegung orientiert sich ebenfalls an der Praxis und greift den Gedanken des Energiesparens nochmals auf. Wie in der Praxis soll der Energieverbrauch minimal werden und deswegen wird der optimale Betriebspunkt des Prüflings mit geringstem Energieverbrauch als Vergleichswert betrachtet.

Umgelegt auf diese Arbeit bedeutet das, dass der Ansatz mit autonomem, rekonfigurierbarem Modul erfolgreich ist, wenn $E_{\mathrm{MCU}} > E_{\mathrm{TC}}$ für alle Prüflinge im optimalen Arbeitspunkt gilt.

In diesem Kapitel wurde ausgehend von der Aufgabenstellung gezeigt, wie die Evaluierung methodisch angegangen wird. Es wurde mit dem SPI-Temperatursensor ein Testfall definiert anhand dessen die Evaluierung durchgeführt wird. Des Weiteren wurden die für Evaluierung notwendigen Ergebnisse erläutert. Es wurde die Menge an unabhängigen, variablen Parametern der Messung und ihr jeweiliger Einfluss auf das Ergebnis vorgestellt. Ein wesentlicher Teil des Kapitels befasst sich mit der Herleitung der Berechnung des Energieverbrauches pro Sensormessung. Dazu wurden drei Berechnungsmöglichkeiten vorgestellt. Den Abschluss des Kapitels bildet die Bewertung der Ergebnisse für welche der optimale Betriebspunkt des jeweiligen DuT vorgeschlagen wurde.

4 Messumgebung

Das Ziel der entwickelten Messumgebung ist die systematische, automatisierte und reproduzierbare messtechnische Erfassung des Stromverbrauches ausgewählter Mikrocontroller und des Testchips beim Abarbeiten eines Sensorprotokolls. Die Teilfunktionen der Messumgebung sind grob drei Themenbereichen zuzuordnen. Der erste Bereich stellt die Steuerung des Messsystems dar. Der zweite Bereich beinhaltet die Prüflinge, also die MCUs und den Testchip. Die dritte Themenbereich ist die eigentliche Messung. Abbildung 4.1 zeigt die Komponenten des Messaufbaus, welche in der folgenden Auflistung erwähnt werden.

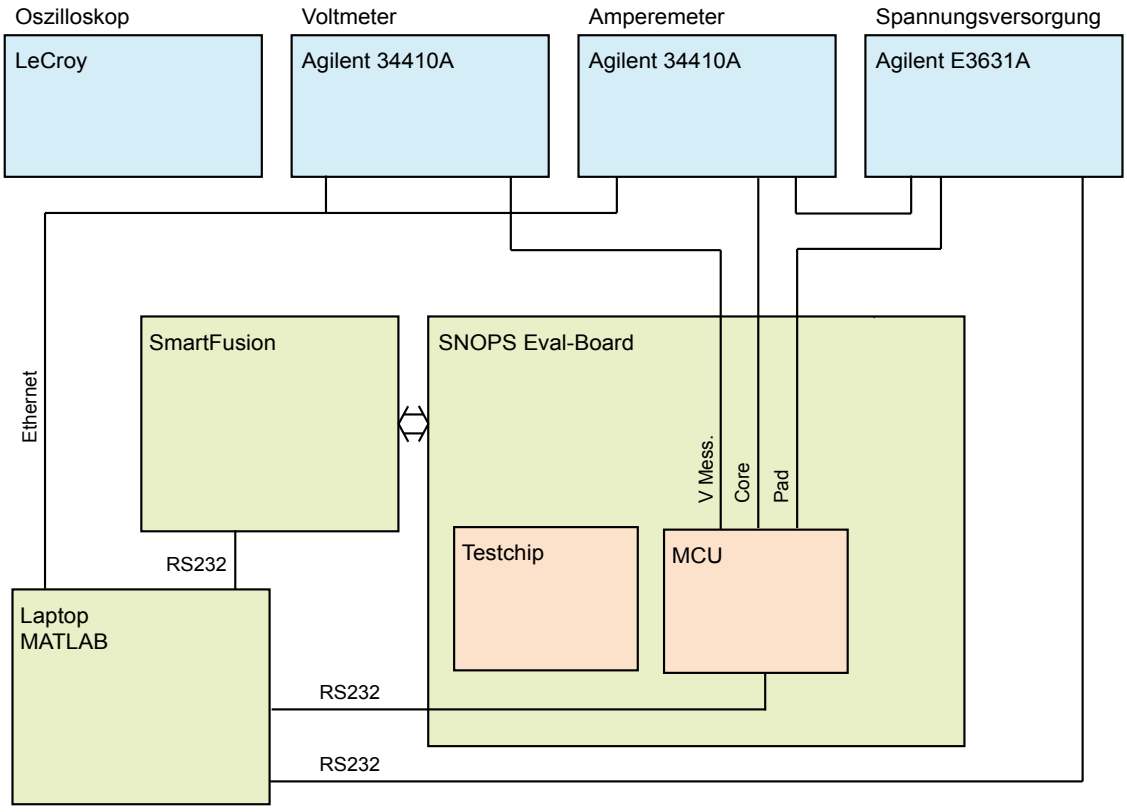


Abbildung 4.1: Blockschaltbild der Messumgebung

- Die Steuerung besteht wiederum aus mehreren Teilfunktionen. Zur koordinierten Ansteuerung aller Komponenten wurde ein Laptop mit MATLAB eingesetzt, siehe Abschnitt 4.5. Für die Spannungsversorgung wurde auf ein Labornetzteil zurückgegriffen wie in Abschnitt 4.2 beschrieben. Die Takterzeugung und Sensor Emulation wurden mit einem SmartFusion FPGA realisiert und sind in Abschnitt 4.4 ausführlich erklärt.

- Aufgrund der Anzahl der Prüflinge wurde auf eine bestehende Testplatine, das Sensor Node Optimization through Power Simulation (SNOPS) Eval-Board, gesetzt. Diese Platine bietet Sockel für Mikrocontroller-Module und das Testchip-Modul und eignet sich optimal für unterschiedliche Konfigurationen, siehe Abschnitt 4.1. Für jeden zu messenden Mikrocontroller wurde ein entsprechendes, kompatibles Modul entwickelt und programmiert. Die Mikrocontroller-Module und die typspezifischen Anpassungen sind in Abschnitt 4.6 erläutert.

- Für die eigentliche Messung wurden Labor-Messgeräte eingesetzt. Diese bieten eine Genauigkeit, die bei Eigenentwicklungen, nur mit großem Aufwand zu erreichen sind. Die Messgeräte und ihre Konfiguration sind in Abschnitt 4.3 beschrieben. Die Messung der aktiven Periode und der Interrupt-Latenz der Mikrocontroller wurden mit einem Oszilloskop gemessen und komplettieren die Messumgebung, siehe Abschnitt 4.8.

4.1 SNOPS Eval-Board

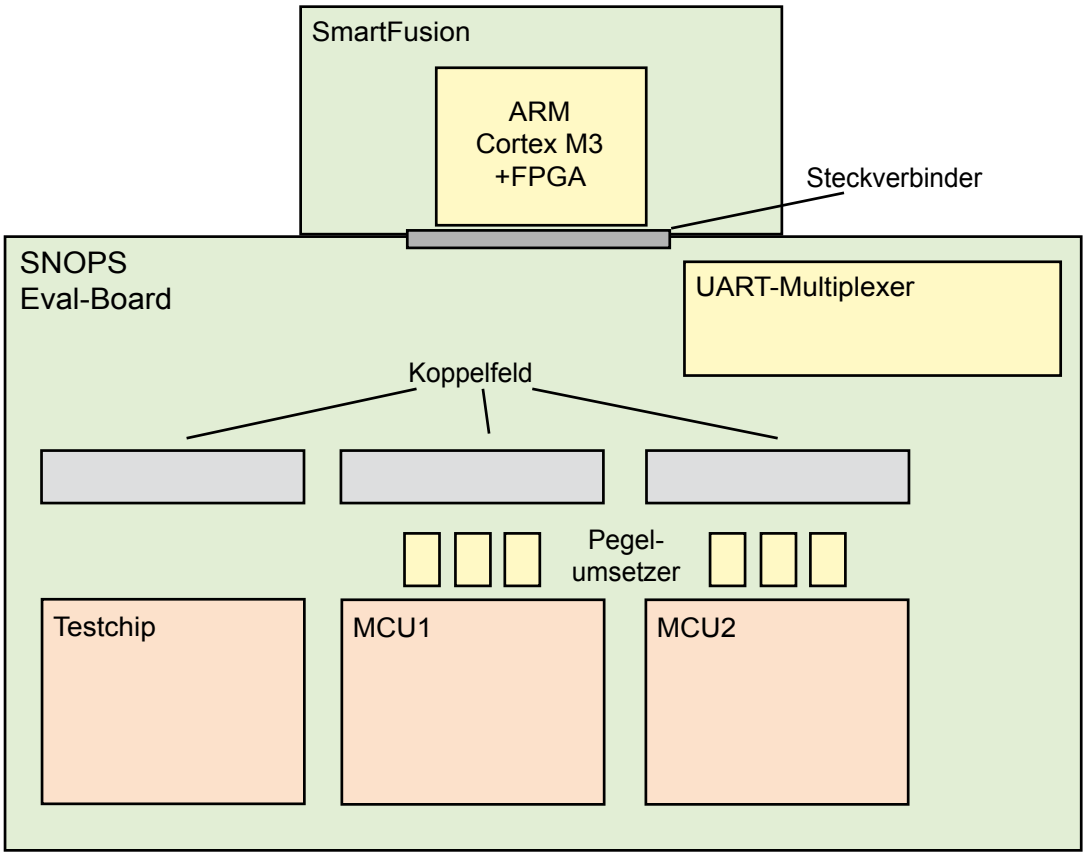


Abbildung 4.2: SNOPS Eval-Board mit SmartFusion Evaluation Kit

Kernstück der Messumgebung ist das SNOPS Eval-Board (Abbildung 4.2). Am Board werden die Prüflinge eingesteckt und sowohl Spannungsversorgung als auch Messgeräte angeschlossen. Über eine Erweiterungsleiste wird ein SmartFusion Evaluation Kit angeschlossen, welches Takterzeugung und SPI-Sensor-Emulation zur Verfügung stellt. Als Ergänzung wurde das Board mit einem digitalen UART Multiplexer aufgerüstet um Mikrocontroller direkt über eine weitere Schnittstelle ansprechen zu können.

Das SNOPS Eval-Board wurde am Institut für Computertechnik entwickelt und stand bereits zu Beginn der Arbeit zur Verfügung [GGHG11]. Es handelt sich dabei um eine Doppel-Layer-Platine. Die Platine verfügt über einen Steckplatz für ein Testchip-Modul, welches den Testchip über einen PLCC-68 Sockel aufnimmt. Weiters verfügt das Board über zwei unabhängige Steckplätze für Mikrocontroller-Module.

Insbesondere komplexe ICs unterstützen eine getrennte Versorgung der logischen Schaltung (Core) und der Pins für Ein- und Ausgabe (Pads). Alle Steckplätze des Eval-Boards sind mit separaten Versorgungsleitungen für Pads und Core ausgestattet. Damit ist es möglich, die Versorgung des Core getrennt zu messen. Die meisten einfachen Mikrocontroller unterstützen jedoch keine getrennte Versorgung. Deswegen sind für beide MCU-Steckplätze zusätzlich Pegelumsetzer vorhanden. Diese dienen dazu, die 3,3 V CMOS-Pegel der SmartFusion Platine in die jeweilige MCU-Versorgungs-Spannung zu übersetzen. Dabei kommen sowohl unidirektionale, für Takt- und Resetleitungen, als auch bidirektionale Umsetzer, für General-Purpose-Input-Outputs (GPIOs), zum Einsatz. Die Pegelwandler werden sekundärseitig von der Pad-Spannung versorgt und nehmen somit keinen Einfluss auf die Messung der Mikrocontroller.

Auf dem SNOPS Eval-Board sind Buchsenleisten als „Koppelfelder“ ausgeführt. Dies ermöglicht eine einfache Umkonfiguration durch Drahtbrücken und bietet darüber hinaus einfache Zugriffsmöglichkeiten auf Signalleitungen für die Fehlersuche. Buchsenleisten bieten weiters einen besseren Schutz vor unbeabsichtigten Kurzschlüssen als Stiftleisten.

Die Platine verfügt weiters über einen Steckverbinder zu einem SmartFusion Evaluation Kit der Firma Microsemi (vormals Actel). Es besitzt einen SmartFusion A2F200M3F-FGG484 Chip, der einen Advanced RISC Machines (ARM) Cortex M3 Prozessor mit FPGA-Zellen (200 000 Gatter, 4608 D-Flip-Flops [Mic13, S. II]) verbindet. Dieses Evaluation Kit wurde nachträglich manuell angepasst um weitere digitale Ausgangsleitungen mit dem SNOPS Eval-Board zu verbinden. Beide Platinen werden über einen USB-Host, welcher mit dem Evaluation Kit verbunden ist, mit Betriebsspannung versorgt.

UART-Multiplexer

Das SNOPS Eval-Board verfügte zu Beginn der Arbeit über einen Analog-Multiplexer. Dieser konnte die serielle Verbindung zwischen dem SmartFusion Prozessor und wahlweise dem USB-Seriell-Umsetzer, dem Testchip und den beiden Mikrocontroller-Steckplätzen umschalten. Die Auswahl der Verbindung legte der SmartFusion Prozessor fest.

Nachteil dieser Umsetzung ist, dass der SmartFusion Prozessor zuerst alle Befehle verarbeiten und anschließend gegebenenfalls weiterleiten muss. Außerdem müssen Übertragungspausen in der PC-basierten Steuerung vorgesehen werden, da der SmartFusion Prozessor immer nur mit einem anderen Device gleichzeitig Daten austauschen kann.

Um Befehle an die serielle Schnittstelle der MCUs nicht durch den Prozessor verarbeiten und weiterleiten zu müssen, wurde eine kompakte Platine mit Complex Programmable Logic Device (CPLD) entwickelt, welche den analogen Multiplexer ersetzt. Zusätzlich zu den bestehenden Verbindungen wurde die Schaltung außerdem um eine weitere serielle Schnittstelle mit 3,3 V Pegel ergänzt. Diese ist über eine 6-Pin-Stiftleiste ausgeführt und kompatibel zu einem integrierten USB-Seriell-Umsetzer-Kabel. Dadurch besitzt der SmartFusion Prozessor nun eine exklusive Verbindung zur PC-Steuerung und über einen zweiten USB-Seriell-Umsetzer kann die jeweils benötigte MCU ebenfalls direkt angesprochen werden.

Damit die Kompatibilität zu existierenden Messprogrammen besteht, sind ein Vierpoliger DIP-Schalter und Leuchtdioden auf der Platine verbaut. Damit können die im CPLD vorprogrammierten Verbindungen ohne Programmiergerät umgestellt werden. Als Chip wird ein XC9572XL der Firma Xilinc, Inc. verwendet. Dieses CPLD verfügt über die entsprechende Anzahl Ein- und Ausgänge und eine maximale Verzögerung von 10 ns [Xil07, S. 4]. Des Weiteren war der IC kurzfristig lieferbar und ein entsprechendes Programmiergerät stand am Institut zur Verfügung.

4.2 Spannungsversorgung

Die Spannungsversorgung wird sowohl zur Versorgung des Testchips, als auch zur Versorgung des Mikrocontrollers eingesetzt. Der geforderte Spannungsbereich reicht von 0,8 V bis einschließlich 3,3 V als obere Grenze, da die meisten Low-Power-Mikrocontroller in Teilen diesen Bereiches betriebsfähig sind. Dabei soll die Spannungsauflösung als wesentliches Kriterium nicht vergessen werden. Im betrachteten Fall wird die geforderte Auflösung mit 0,8 mV spezifiziert damit der relative Fehler bei der Minimalspannung unterhalb von 1 ‰ liegt. Aufgrund der umfangreichen Testmöglichkeiten ist eine wirksame Strombegrenzung unerlässlich. Damit sollen bleibende Schäden, die durch Fehlkonfigurationen entstehen können, vermieden werden. Eine weitere wesentliche Forderung, die sich aufgrund der Automatisierung der Messung ergibt, ist die Fernsteuerbarkeit.

Das SNOPS Eval-Board bietet mehrere, über Trimm-Potentiometer einstellbare, Spannungsversorgungen an, welche mit Operationsverstärkerschaltungen ausgeführt sind. Durch die fehlende Möglichkeit, die Potentiometer im automatischen Messaufbau anzusteuern, wurde diese Spannungsversorgung nur für anfängliche Tests und bei der Fehlersuche in den Mikrocontroller-Programmen benutzt.

Aufgrund der beschriebenen Anforderungen wurde das externe Labor-Netzgerät „Agilent E3631A“ [Agi12b] ausgewählt. Es verfügt über 3 Kanäle mit +6 V, +25 V und -25 V. Der +6 V Kanal ist mit einer Auflösung von 0,5 mV angegeben und liegt damit unter der Spezifikation. Die 25 V Kanäle sind jeweils mit 1,5 mV angegeben und liegen damit über dem gewünschten Wert. Deswegen wird der +6 V Ausgang für jeweils Mikrocontroller und Testchip Core verwendet. Der +25 V Output versorgt die Pegelwandler beziehungsweise die Pads des Testchips. Vorteilhaft ist, dass sich beide Ausgänge nur zusammen ein- und ausschalten lassen. Damit erfordert die Ansteuerung des Netzgerätes weniger Aufwand, um Sicherzustellen, dass beide Spannungen ohne große Zeitverzögerung anliegen.

Angesprochen wird das E3631A über Standard Commands for Programmable Instruments (SCPI) Kommandos. Dafür bietet das Netzgerät General Purpose Interface Bus (GPIB) und RS-232 als Schnittstelle an. Aufgrund der besseren Verfügbarkeit wurde RS-232 der Vorzug gegeben.

4.3 Messgeräte

Zur Strommessung und zur Spannungsmessung am DuT wurde das „Agilent 34410A“ [Agi12a] ausgewählt. Es handelt sich dabei um ein 6,5-stelliges Tischmultimeter. Neben der benötigten Auflösung verfügt das Messgerät auch über einen Trigger-Eingang und ist fernsteuerbar. Des Weiteren stand das Messgerät am ICT in doppelter Ausführung zur Verfügung, womit gleichzeitig sowohl Strom als auch Spannung gemessen werden konnten.

Die Auflösung der Strommessung wurde wie folgt festgelegt. Der geringste Stromverbrauch der MCUs liegt laut Datenblättern im LPM vor und ist im einstelligen μA-Bereich, vgl. Abschnitt 4.6. Wenn bei einem Messwert von 1 μA 5 signifikante Stellen gefordert werden, ergibt sich eine benötigte Auflösung von 10 pA.

Nach Tabelle 4.1 ist der kleinste Messbereich 100 µA und eine Auflösung von 10 pA entsprechen 0,1 ppm des Messbereiches. Die Tabelle 4.2 zeigt die dafür benötigte Integrationszeit von 200 ms. Diese Integrationszeit entspricht 10 NPLC und erfüllt damit die in Abschnitt 3.5 erwähnte Forderung, dass die Integrationszeit ein Vielfaches der Netzperiode sein soll. Zur Vereinfachung der Messung wurde bei der Spannungsmessung mit dem zweiten 34410A die gleiche Integrationszeit gewählt. Um einzelne Fehlmessungen besser kompensieren zu können, werden 50 unmittelbar aufeinander folgende Messungen mit jeweils 200 ms Integrationszeit durchgeführt. Anschließend wird das arithmetische Mittel über die 50 Messungen gebildet und Werte mit gravierender Abweichung ausgeschlossen.

Messbereich	Messshunt
3 A	0,1 Ω
1 A	0,1 Ω
100 mA	2,0 Ω
10 mA	2,0 Ω
1 mA	200,0 Ω
100 μA	200,0 Ω

Tabelle 4.1: Strommessbereich und jeweiliger Messshunt [vgl. Agi12a, S. 127]

NPLC	Integrations-zeit	Auflösung (ppm Messbereich)
0,2	4 ms	0,7
1,0	20 ms	0,3
2,0	40 ms	0,2
10,0	200 ms	0,1
100,0	2000 ms	0,03

Tabelle 4.2: Integrationszeit gegenüber Auflösung [vgl. Agi12a, S. 126]

Die Ansteuerung der Messgeräte erfolgt mit SCPI über Ethernet. Das Messgerät verfügt auch über eine USB- sowie eine GPIB-Schnittstelle. Nachdem der Steuerungs-Laptop über keine GPIB- und nur eine begrenzte Anzahl USB-Schnittstellen verfügt, wurde Ethernet ausgewählt. Durch SCPI lässt sich das Messgerät vollständig konfigurieren und die Messung starten. Die Messwerte werden dann in einem 50000 Werte fassenden Pufferspeicher zwischengespeichert und können nach der Messung von der Steuerung des Messsystems abgefragt werden. Zur Synchronisation der Messung

wie in Abschnitt 3.5 erwähnt, besitzt das Messgerät einen Trigger-Eingang. Während der Trigger-Impuls eine Mindestdauer von 1 µs haben muss, ist die auslösende Flanke konfigurierbar.

Wie bereits in Abschnitt 3.2 ausgeführt, sind bei der Strommessung kleine Ströme zu erwarten, welche bis in den µA-Bereich reichen können. Bei der verwendeten Messmethode des gewählten Messgerätes mit Messshunt und anschließender Integration tritt durch den ohmschen Widerstand des Shunts ein Spannungsabfall auf. Tabelle 4.1 zeigt den ohmschen Widerstand des Messshunts in Abhängigkeit des Messbereiches. Dieser Spannungsabfall muss ausgeregelt werden, um systematische Messfehler zu vermeiden. Dies kann entweder durch direkte Regelung der Spannungsquelle mit Messfühler am DuT oder durch Spannungsregelung von der Messumgebung aus erfolgen. Dabei wird jedoch ein Voltmeter am DuT benötigt, welches sonst nicht notwendig wäre. Weil die in Abschnitt 4.2 ausgewählte Spannungsversorgung keine direkte Regelung mit Messfühler ermöglicht, wird die Spannungsregelung von der Steuerung des Messsystems übernommen, siehe Abschnitt 4.5.

Die Verbindungen zwischen Spannungsversorgung, Messgeräten und Testobjekt sind bewusst mit Koaxialleitungen ausgeführt. Der höhere Leitungswiderstand im Vergleich zu konventionellen Laborleitungen verursacht, aufgrund der geringen Ströme, einen vernachlässigbaren Spannungsabfall. Im Gegenzug erhält man eine wesentlich bessere Störfestigkeit gegen Einstrahlungen des Messumfeldes.

4.4 Takterzeugung und Sensor Emulation

Die Takterzeugung und die Sensor Emulation sind im FPGA-Teil des SmartFusion Chip umgesetzt. Die Module sind in VHDL erstellt und bieten eine Advanced High-Performance Bus (AHB)-Schnittstelle zur direkten Anbindung an den ARM Prozessor. Über den AHB werden Register in den Modulen gesetzt, welche die Steuerungsinformation enthalten. Dadurch wird die kurze Reaktionszeit eines FPGAs mit der Flexibilität des Prozessors verbunden.

Abbildung 4.3 zeigt die Anbindung der VHDL-Module an den Prozessor Kern. Der AHB dient dabei für den Datenaustausch zwischen den einzelnen Modulen. Als Prozessor kommt der ARM Cortex M3 zum Einsatz. Am SmartFusion Chip ist eine serielle Schnittstelle nach RS232 hardcoded vorhanden, welche ebenfalls über den AHB angesprochen wird. Die im FPGA realisierten Module besitzen jeweils eine eigene AHB Schnittstelle. Zur Takterzeugung werden die Module „OSC“ und „RTC“ verwendet, welche ident sind und sich nur durch ihre Adresse am AHB unterscheiden. Die Takterzeugung ist eingehend in Abschnitt 4.4.1 beschrieben. Ebenfalls im FPGA realisiert ist der SPI-Sensor Emulator. Der Sensor Emulator besitzt eine eigene Adresse am AHB und kann direkt vom Prozessor konfiguriert werden. Das Modul ist in Abschnitt 4.4.2 beschrieben. Das Ein-Ausgabe-Modul verwendet einen Multiplexer um die benötigten Signale über die Pins des SmartFusion Chips direkt auf die entsprechenden Signalleitungen am SNOPS-Eval Board zu mappen. Dadurch kann zwischen MCU und Testchip einfach umgeschaltet werden und der Verdrahtungsaufwand am SNOPS-Eval Board reduziert sich.

4.4.1 Takterzeugung

Die Takterzeugung ist als synchroner Frequenzteiler ausgeführt. Als Basisfrequenz wird der interne 100 MHz Oszillator des SmartFusion Chips genutzt [Mic13]. Das Frequenzteiler-Modul verfügt

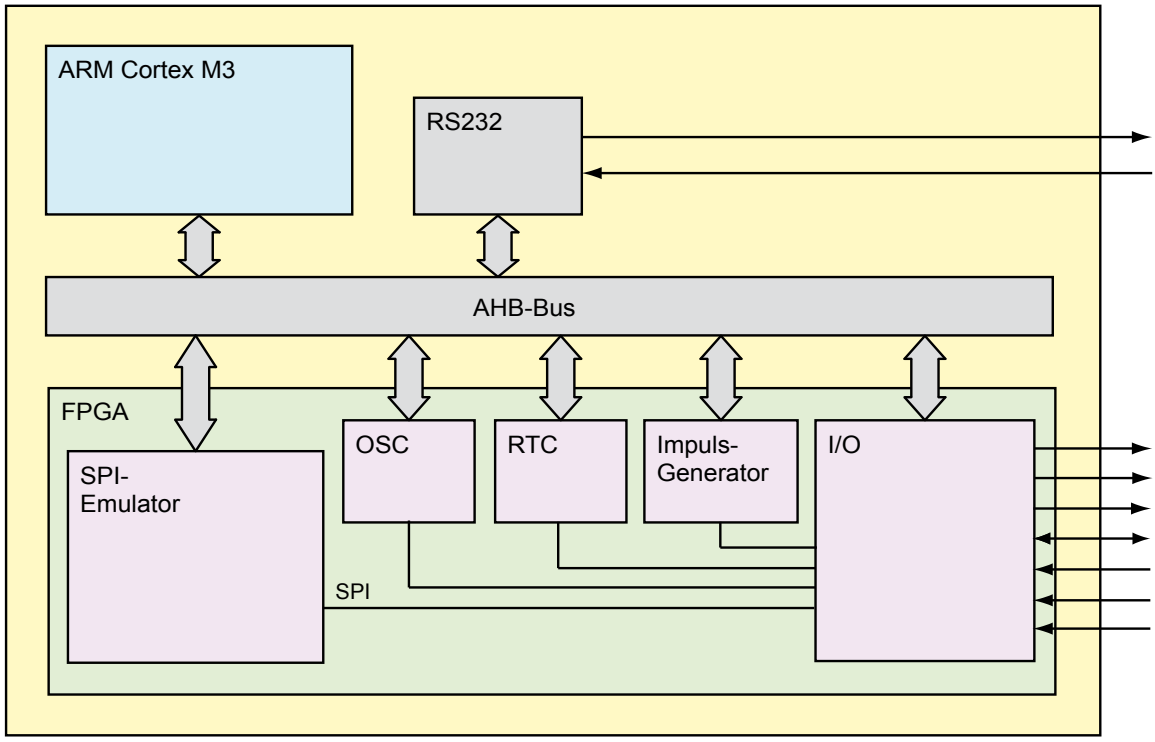


Abbildung 4.3: Blockschaltbild des SmartFusion Chips

über zwei 16 Bit Register. Das Register an Adresse 0 enthält den Teilerwert und an Adresse 1 kann ein Zähler angegeben werden, wie viele Taktzyklen ausgegeben werden sollen. Dabei wird der Frequenzteiler bei Zählerstand gleich Null deaktiviert, während der Wert 0xFFFFFFFFh Dauerbetrieb entspricht. Das Modul unterstützt auch ungerade Teilerverhältnisse, jedoch wird prinzipbedingt die aktive Phase um einen 100 MHz Takt länger. Das Frequenzteiler-Modul ist in dem Design doppelt, jedoch mit unterschiedlichen Adressen, instanziiert. Die erste Instanz wird als Taktgeber der Mikrocontroller und des Testchips verwendet und die zweite als 32 kHz RTC-Quelle. Die erzielte Ausgangsfrequenz lässt sich mit Gleichung (4.1) berechnen.

$$f = \frac{100\,\mathrm{MHz}}{k_{\mathrm{Teiler}} + 1} \tag{4.1}$$

Nachteil der Umsetzung als Frequenzteiler ist, dass nicht beliebige Frequenzen eingestellt werden können. Eine Phase Locked Loop (PLL) mit Skalierungs- und Teilerstufen würde eine große Flexibilität bringen, stand jedoch am eingesetzten Chip nicht zur Verfügung. Außerdem erhält man durch die gewählte Implementierung die Möglichkeit, eine definierte Anzahl an Taktzyklen ablaufen zu lassen.

4.4.2 Sensor Emulation

Der Sensor Emulator ist für SPI als VHDL-Modul ausgeführt. Beim Design wurde darauf geachtet, den Emulator möglichst universell zu gestalten, sodass nicht nur eine bestimmte SPI-Sensortype

emuliert werden kann. Der Emulator ist für byteweise Übertragung ausgelegt, wie sie von den meisten MCUs und auch vom Testchip unterstützt wird. Durch Konfiguration lassen sich sowohl adressierbare SPI-Sensoren als auch nur lesbare Sensoren nachbilden.

Der Sensor Emulator ist komplexer als das Takterzeugungsmodul und verfügt über eine Reihe von Registern, siehe Tabelle 4.3. Aufgrund der Umsetzung in einem FPGA wurde die Registerbreite jeweils auf die minimal notwendige Anzahl Bits reduziert. Dadurch ergeben sich die unterschiedlichen Registerbreiten. Alle Register sind sowohl lesbar als auch beschreibbar, wobei einschränkend erwähnt werden muss, dass die mit (ro) gekennzeichneten Register bei Schreibzugriff, unabhängig von den zugewiesenen Daten, auf Null gesetzt werden. Zusätzlich zu den Registern verwendet das Modul einen 256 Byte großen RAM in welchem die Sensorwerte, welche über das SPI-Interface ausgelesen werden können, gespeichert sind.

Bevor auf die Bedeutung der einzelnen Register genauer eingegangen wird, werden einige Grundlagen von SPI kurz erläutert. SPI ist als Master-Slave System aufgebaut, welches genau einen Master und beliebig viele Slaves enthält. Mit dem Slave Select (SS)-Signal wählt der Master den gewünschten Slave aus und aktiviert dessen Eingangssignale. Mit Serial Clock (SCLK) gibt der Master den Takt für die serielle Übertragung vor. Der eigentliche Datenaustausch erfolgt mit Master Out Slave In (MOSI) und Master In Slave Out (MISO). Logisch bilden MOSI und MISO ein in sich geschlossenes, verteiltes Schieberegister.

Im folgenden werden die einzelnen Register des Moduls in Tabelle 4.3 genauer erklärt. Erst wenn das Bit von „Modul Ein“ gesetzt ist, verarbeitet der Sensor Emulator die Eingangssignale. Die Anzahl Bytes pro Übertragung gibt an, wie viele Bytes in einem kompletten Abfrage-Zyklus des Sensors übertragen werden. Dabei kann das SS Signal nach jedem vollständigen Byte deaktiviert und aktiviert werden. Die Anzahl Datenbytes gibt an, wie viele Bytes an Daten bei einer Übertragung gesendet werden. Wenn die Anzahl Datenbytes gleich der Gesamtzahl der übertragenen Bytes ist, wird ein nur lesbarer Sensor emuliert. Das Register RAM-Adresse kann nur gelesen oder zurück gesetzt werden und enthält die Adresse des nächsten übertragenen Sensorwertes. Die maximale RAM-Adresse gibt des Ende des Speicherbereiches an, in welchem die Sensorwerte abgelegt sind. Wenn die RAM-Adresse die maximale Adresse erreicht, wird sie nach der kommenden Übertragung wieder auf Null gesetzt. Fehlerzähler wird inkrementiert, wenn ein Protokollverletzung detektiert wird. Die Ursache dafür kann ein Unterschied zwischen empfangenem und Kommando Byte sein oder auch wenn das SS-Signal vor einem vollständig übertragenem Byte wieder inaktiv wird. Die jeweiligen Kommando Bytes werden mit dem empfangenen Byte verglichen und erfüllen außerdem keine weitere Aufgabe. Sie dienen also lediglich der Fehlererkennung und erleichtern die Synchronisation bei Übertragungsfehlern. Der Synchronizer wird an späterer Stelle in diesem Abschnitt ausführlicher erklärt.

Im Gegensatz zu realen Sensoren in welchen das SPI-Interface direkt mit SCLK getaktet wird, werden die Flanken von SCLK im Emulator detektiert und in ein Enable-Signal umgewandelt. Ebenso wird mit den Flanken des SS-Signals verfahren. Die Detektion der Flanken wird dadurch möglich, dass der FPGA-Kern mit 100 MHz getaktet ist, während die maximale Geschwindigkeit auf SPI durch die Taktfrequenz des Master und den Busteiler bestimmt wird. In dieser Arbeit ist die maximale Frequenz des Master 10 MHz und der minimale Busteiler ist bei SPI immer 2. Daraus ergibt sich die maximale Busfrequenz von 5 MHz. Somit ist eine entsprechende Überabtastung gewährleistet und die Flanken werden sicher detektiert.

Kritischer ist die Synchronisation der Eingangsignale des Emulators mit dem FPGA-Takt. Aufgrund des Übergangs zwischen unterschiedlichen Taktdomänen kann es zu Setup- und Hold-Time Verletzungen kommen, die sich wiederum in unbestimmten Zuständen in den Flip-Flops

Adresse	Registerbreite (Bit)	Funktion
0d	1	Modul Ein/Aus
1d	3	Anzahl Bytes pro Übertragung
2d	3	Anzahl Datenbytes
3d	8	RAM-Adresse (ro)
4d	8	Max. RAM-Adresse
5d	8	Fehlerzähler (ro)
6-13d	8	Kommando Bytes
14d	20	Synchronizer Verzögerung
15d	20	Synchronizer Pulsdauer

Tabelle 4.3: Register des Sensor Emulators

äußern und in weiterer Folge zu Fehlern führen. Aufgrund des schnellen FPGA-Taktes lassen sich die Eingangssignale mit zwei vorgeschalteten Flip-Flops synchronisieren, ohne eine wesentliche Verzögerung zu verursachen. In praktischen Tests führte das Aussparen dieser zweistufigen Synchronisations-Flip-Flops zu einem drastischen Anstieg der Übertragungsfehler.

In Abbildung 4.4 ist der Zustandsautomat des Sensor Emulators dargestellt. Nach einem Reset befindet sich der Emulator im „Idle“-Zustand. Solange das Modul nicht eingeschaltet ist, bleibt es in diesem Zustand. Wenn das Modul aktiviert ist, geht es bei einer fallenden Flanke des SS-Signals in den „Slave Asserted“-Zustand über. Mit einer fallenden Flanke des SCLK-Signals wird der Beginn einer Übertragung detektiert und der Automat geht in den „Run“-Zustand über. In diesem Zustand vergleicht das Modul die übertragenen Bits mit den Bits des jeweiligen Kommando Bytes und legt die Ausgangsbits an das MISO-Signal. Wenn eine steigende Flanke des SS-Signals erkannt wird und das Byte vollständig übertragen wurde, geht der Zustandsautomat wieder in den „Idle“-Zustand über. Wenn im „Run“-Zustand ein empfangenes Byte nicht dem Kommando Byte entspricht oder während eines Bytes das SS-Signal eine steigende Flanke aufweist, wird in den „Error“-Zustand gewechselt. Dieser Zustand wird nach dem Erhöhen des Fehlerzählers um Eins umgehend verlassen und wieder in den „Idle“-Zustand übergegangen. Anschließend werden Byte- und Bit-Zähler wieder initialisiert und das Modul somit betriebsbereit gemacht.

Der Fehlerzähler stellte sich im Besonderen bei der Inbetriebnahme der einzelnen MCUs als hilfreich heraus, da jede Protokollverletzung registriert wird. Im automatischen Messbetrieb wird der Fehlerzähler hingegen nicht ausgewertet, da nach erfolgreicher Inbetriebnahme keine Fehler auftraten.

Ein Submodul des Sensor Emulators ist der Synchronizer. Die Aufgabe des Synchronizers ist es, einen Trigger für die Synchronisation der Messgeräte bereit zu stellen. Abgeleitet wird der Impuls von einer fallenden Flanke des SS-Signals. Die letzten beiden Register in Tabelle 4.3 sind die Konfigurationsregister des Synchronizers. Damit lässt sich die Verzögerung zwischen fallender SS-Flanke und Impuls und die Impuls-Dauer konfigurieren. Beide Werte lassen sich zwischen 10 ns und 10,5 ms einstellen.

Der ADT7310 von Analog Devices [Ana09] ist ein SPI Temperatursensor von moderater Komplexität. Der Sensor ist konfigurierbar und bietet eine hohe Auflösung. Außerdem weist der Sensor einen Messzyklus wie in Abbildung 3.1 dargestellt auf. Aufgrund der bereits in Abschnitt 3.2 genannten Gründe wird nicht der Sensor direkt, sondern die Emulation des Sensorprotokolls für die Messung verwendet. Abbildung 4.5 und Tabelle 4.4 zeigen die Konfiguration des Sensor Emulators für das Protokoll des ADT7310. Das Protokoll des ADT7310 erfordert zwei Aktivierungen des

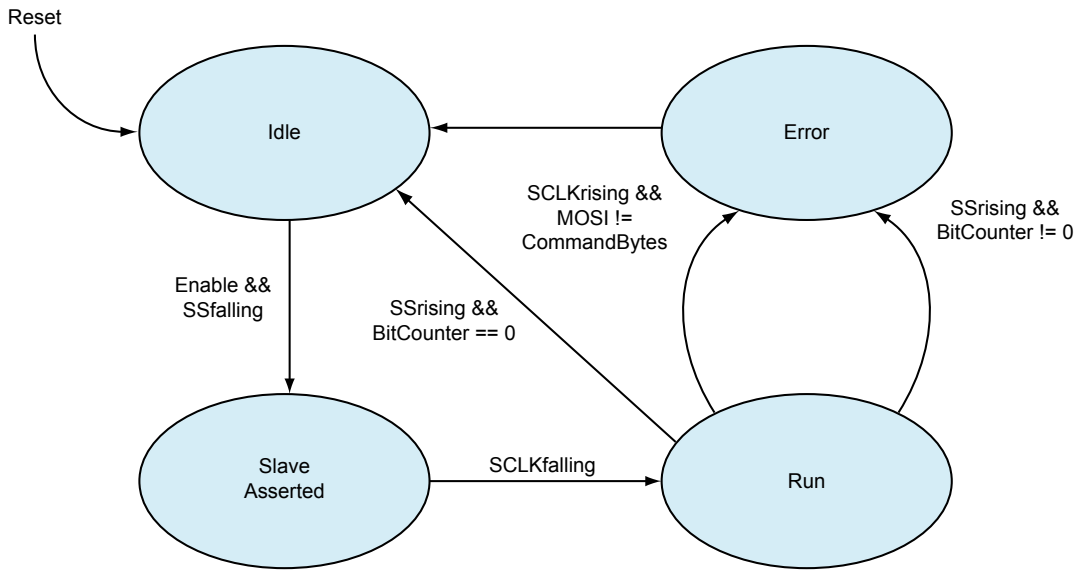


Abbildung 4.4: Zustandsautomat des Emulators

Slave Select Signals. Bei der ersten Übertragung wird der Sensor aktiviert und eine Temperaturmessung mit anschließender Analog-Digital-Wandlung getriggert. Für die Wandlung benötigt der reale Sensor mindestens 240 ms, während der Emulator das Ergebnis direkt liefern kann. Bei der zweiten Übertragung wird nochmals ein Kommando Byte übertragen, welches die entsprechende Speicherstelle im Sensor adressiert. Danach kann der Sensor-Wert aus dem Speicher des Sensors gelesen werden.

Zur Konfiguration des Emulators muss nun in folgender Weise vorgegangen werden. Die Anzahl der Bytes pro Übertragung und die Anzahl der Datenbytes werden entsprechend des Protokolls gesetzt. Dabei muss beachtet werden, dass aufgrund der Struktur des Emulator-Moduls der Register-Wert jeweils um Eins reduziert wird. Die Kommando Bytes werden entsprechend des Protokolls vom höchsten zum niedrigsten gesetzt. Das bedeutet im Fall des ADT7310, dass das Kommando Byte 4 als erstes Byte der Übertragung erwartet wird, siehe Abbildung 4.5. Die maximale RAM-Adresse wird auf 19 für die 20 im RAM abgelegten Sensorwerte gesetzt. Im RAM selbst werden die in Tabelle 3.2 gefundenen Werte abgelegt. Dabei ist zu beachten, dass der Sensor eine 13-Bit Auflösung hat und die Daten bei der Übertragung mit dem Most Significant Bit (MSB) beginnen, also alle Sensorwerte im RAM um drei Bits nach Links geshiftet werden müssen. Wenn alle Register gesetzt sind, kann das Emulator-Modul aktiviert werden.

4.5 Steuerung des Messsystems

Die Steuerung des Messsystems erfolgte von einem Laptop mit Matlab. Die Verwendung von Matlab für diese Aufgabe hat sich aus mehreren Gründen angeboten.

- Matlab bietet Schnittstellen zur Kommunikation über Ethernet und serieller Schnittstelle.

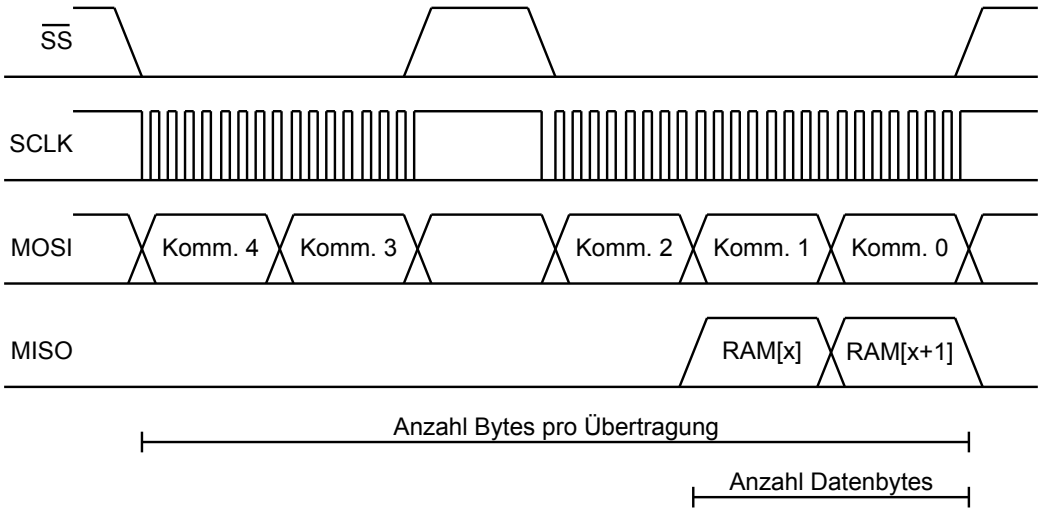


Abbildung 4.5: Sensorprotokoll des ADT7310 im Emulator

Adresse	Funktion	Wert	Bedeutung
0d	Modul Ein/Aus	1d	SPI-Emulator aktivieren
1d	Anzahl Bytes pro Übertragung	4d	5 Bytes
2d	Anzahl Datenbytes	1d	2 Bytes
4d	Max. RAM-Adresse	19d	20 Bytes im RAM
6d	Kommando Byte 0	FFh	Dummy für Datenbyte
7d	Kommando Byte 1	FFh	Dummy für Datenbyte
8d	Kommando Byte 2	50h	drittes Kommando Byte
9d	Kommando Byte 3	20h	zweites Kommando Byte
10d	Kommando Byte 4	08h	erstes Kommando Byte

Tabelle 4.4: Registerwerte zur Konfiguration des Sensor Emulators für ADT7310

- Matlab verfügt über alle benötigten Funktionen zur statistischen Auswertung und Verarbeitung der Messdaten und eignet sich gut zur Erstellung von Diagrammen aus den Messdaten.
- Es existierte zu Beginn der Arbeit bereits ein Mess-Skript welches als Referenz für die weitere Arbeit diente und die Einarbeitungszeit verkürzte.
- Matlab stand mit einer Studentenlizenz günstig zur Verfügung und konnte somit auch außerhalb des Instituts-Labors verwendet werden.

Die Steuerung des Messsystem ist durch Matlab Klassen strukturiert. Die programmierten Klassen sind dabei zwei Teilbereichen zuzuordnen. Den ersten Teilbereich stellen die Treiber für Messgeräte, Spannungsversorgung und Kommunikation mit MCUs und SmartFusion Prozessor dar. Ebenfalls diesem Bereich zuzurechnen sind die zugrundeliegenden Schnittstellen-Treiber. Der zweite Teilbereich umfasst den Ablauf der Messung und die zur Verwaltung der Mess- und Steuerungsdaten verwendeten Container. Bevor auf den Ablauf der Messung genauer eingegangen wird, werden die einzelnen Klassen erläutert. Abbildung 4.6 zeigt die Abhängigkeiten der Klassen.

Die Schnittstellen-Treiber bilden die Basis für die Kommunikation mit den Komponenten der Messumgebung. Zu den benötigten Treibern gehören der für Ethernet und für die serielle Schnittstelle.

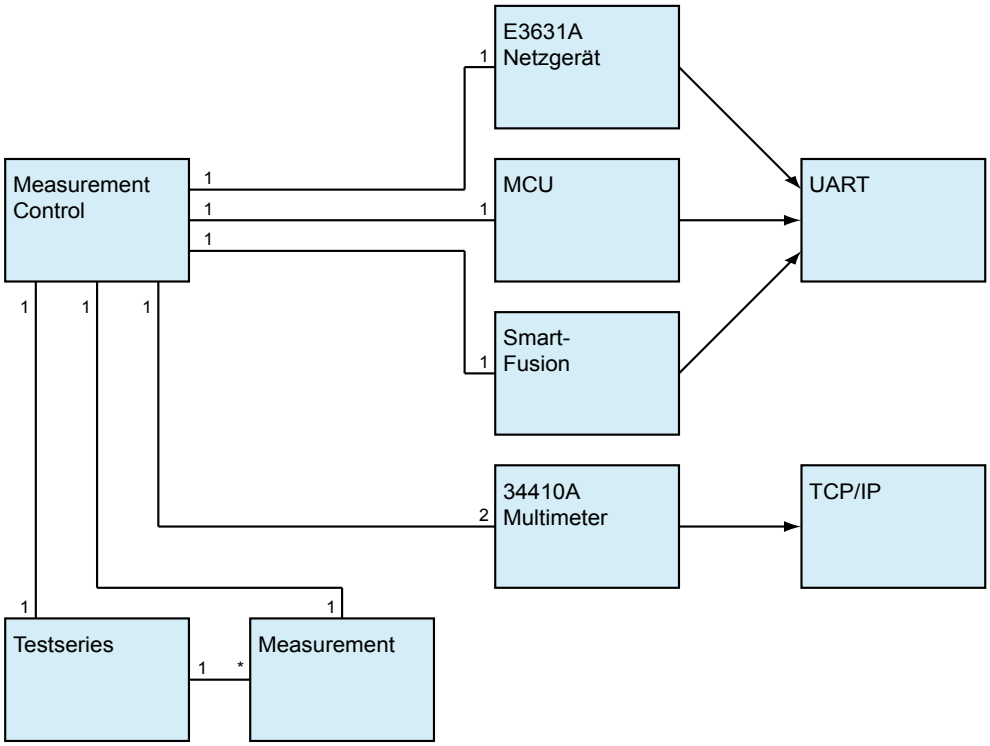


Abbildung 4.6: Klassendiagramm der Matlab Klassen

Beide Treiber verfügen über einheitliche Funktionsaufrufe für Schreib- und Lesebefehle. Dadurch wird es möglich, bei Bedarf, schnell auf die jeweils andere Schnittstelle umzustellen.

Für die Spannungsversorgung und die Messgeräte sind jeweils eigene Klassen definiert, welche von den benötigten Schnittstellen-Klassen abgeleitet sind. Die Voltmeter- und Amperemeter-Funktion sind in der Messgeräte-Klasse vereint, da es große Gemeinsamkeiten bei den Steuerbefehlen gibt. Als Basis der Steuerbefehle dient sowohl beim Netzgerät als auch bei dem Messgerät SCPI, welches um gerätespezifische Befehle erweitert ist.

Die Klasse für den SmartFusion-Treiber erbt die Klasse des seriellen Schnittstellen-Treibers. Zusätzlich sind Funktionen zum Konfigurieren des Frequenzteilers und aktiveren der Takterzeugung vorgesehen. Alle Steuerbefehle werden vom SmartFusion Prozessor quittiert oder der gesetzte Wert zurück übertragen. Die Klasse wertet diese Rückmeldung aus und liefert eine Fehlermeldung, wenn nicht der erwartete Wert empfangen wird. Der Testchip wird auch über den SmartFusion Prozessor konfiguriert, siehe Abschnitt 4.7.

Die Konfiguration der Mikrocontroller wird in Abschnitt 4.6 ausführlich beschrieben. Die zugehörige Matlab-Klasse verfügt über den seriellen Schnittstellen-Treiber und setzt die Matlab Funktionsaufrufe in die entsprechenden Steuerzeichen um.

Zur Verwaltung der Messdaten wurden zwei Klassen programmiert. Die „Testseries“-Klasse enthält alle Informationen, die während eines gesamten Messdurchlaufes erfasst werden. Dazu zählen die Konfigurationsparameter und pro Messpunkt eine Instanz der Klasse „Measurement“ welche die Vorgabeparameter des einzelnen Messpunktes und die gemessenen Strom- und Spannungswerte enthalten. Nach dem vollständigen Messdurchlauf werden die in „Testseries“ zusammengefassten Daten in ein Matlab Datenfile gespeichert und können anschließend ausgewertet werden.

Die Klasse „MeasurementControl“ steuert die Messung und koordiniert die einzelnen Komponenten. Aufgrund der großen Zahl an Konfigurationsparametern wurde zusätzlich ein Skript erstellt, welches die Parameter der Messung und die verwendeten Geräte definiert. Zu diesen Parametern zählen unter anderem die einzelnen Versorgungsspannungs- und Taktfrequenzwerte. Zu beachten ist, dass durch das Laden der Parameter noch keine Verknüpfung der Parameter zu Messpunkten stattgefunden hat.

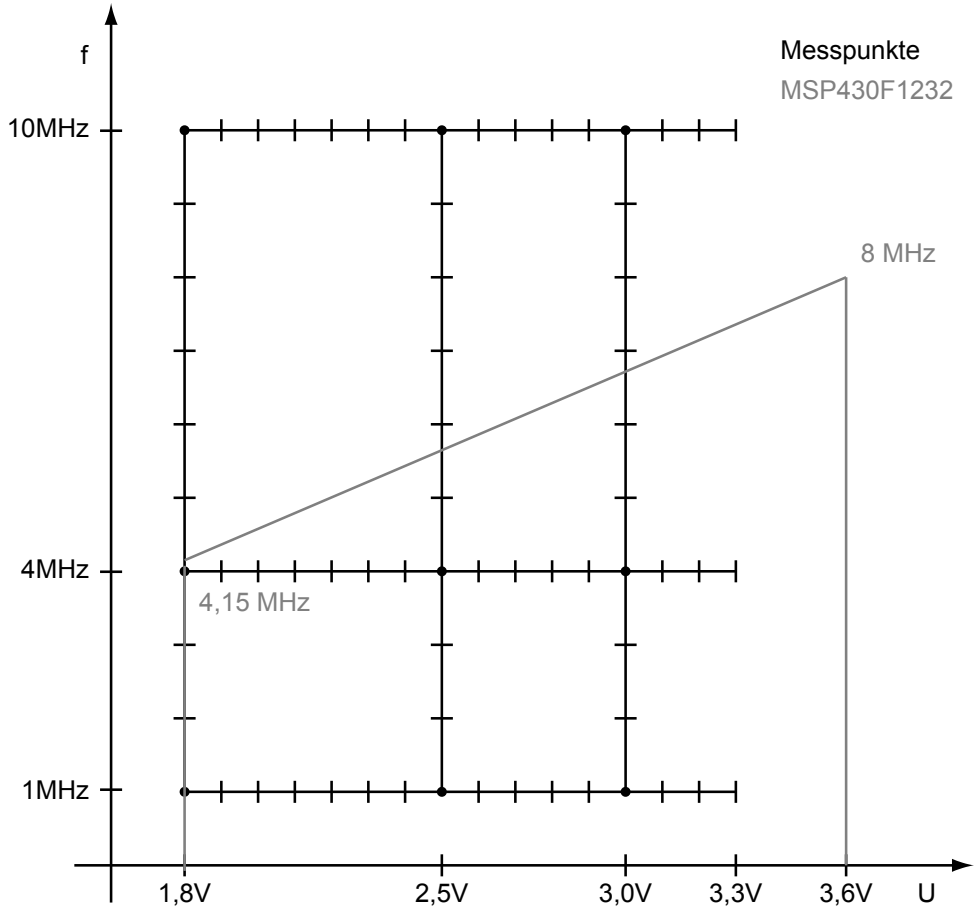


Abbildung 4.7: Messpunkte in Spannungs-Taktfrequenz-Ebene mit MSP430F1232 Betriebsbereich

Der Messablauf startet mit der Instanziierung der „MeasurementControl“-Klasse, welche im Weiteren alle benötigten Klassen lädt. Anschließend wird die Punktegenerator-Funktion aufgerufen. Diese Funktion generiert aus dem Parametersatz der Konfiguration die einzelnen Messpunkte. Wie in Abschnitt 3.4 erwähnt, werden nicht alle möglichen Parameter-Kombinationen gemessen sondern nur eine Auswahl davon. Abbildung 4.7 zeigt die erzeugten Messpunkte in der Spannungs-Taktfrequenz-Ebene. Zur Reduktion der Anzahl der Messpunkte wird jeweils nur ein Parameter variiert, während die anderen auf vordefinierten Standardwerten bleiben. Für Versorgungsspannung und Taktfrequenz werden jeweils drei Standardwerte zugelassen, entsprechend für unterem, mittlerem und oberem Bereich. Dadurch ergibt sich das Raster in Abbildung 4.7. An den Schnittpunkten der Linien werden entsprechend auch die Parameter wie Zykluszeit, Schwellwert und SPI-Busteiler einzeln variiert. Die Wahl von 3 V als obere Grenze wurde aufgrund der Mikrocontroller-Datenblätter so gewählt, weil die Datenblätter den Stromverbrauch der MCUs bei 3 V angeben

und somit ein direkter Vergleich durchgeführt werden kann. Ebenfalls in Abbildung 4.7 aufgenommen wurde der Betriebsbereich eines Mikrocontrollers um die Relation des Messbereiches mit dem sicheren Betriebsbereich gegenüberzustellen. Der Punktegenerator sortiert Messpunkte außerhalb des Betriebsbereiches automatisch aus um die gesamte Messdauer weiter zu verkürzen.

Nach der Definition der Messpunkte erfolgt die eigentliche Messung in zwei Phasen. Die erste Phase oder Setup sorgt dafür, dass die Parameter gesetzt werden und konfiguriert Mikrocontroller oder Testchip. In der zweiten Phase oder Messung werden nach erfolgreichem Setup die Messgeräte getriggert und anschließenden die gemessenen Werte abgefragt und gespeichert. Auf zwei Besonderheiten in der Setup-Phase soll hier noch genauer eingegangen werden. Das sind die Auswahl des optimalen Strommessbereiches und das Ausregeln der Versorgungsspannung.

Während das Messgerät in anderen Betriebsmodi den Messbereich automatisch bestimmen kann, war dies bei externer Triggerung nicht möglich. Um dennoch immer im optimalen Messbereich messen zu können, wurden die Messbereiche beginnend beim kleinsten getestet. Sofern der Messwert außerhalb des Bereiches lag, wurde ein Overflow vom Messgerät zurückgemeldet und die Steuerung wählte den nächstgrößeren Bereich aus. Durch das anschließende Nachregeln der Versorgungsspannung, konnte es passieren, dass der Messbereich dennoch überschritten wurde. Deswegen wurde zusätzlich eine Schranke bei 95 % des Messbereich-Nennwertes eingezogen.

Um trotz Spannungsabfalles am Amperemeter, siehe Abschnitt 4.3, die exakte Versorgungsspannung am DuT anliegen zu haben, wurde mit Netzgerät und Voltmeter ein Regelkreis gebildet. Der Regler wurde in Matlab als Schleife programmiert und hat integrales Verhalten um die Abweichung exakt auszuregeln. Wesentlich anzumerken ist, dass bei dieser Methode keine aktive Regelung während der Messung stattfinden kann, weil das Voltmeter synchron zum Amperemeter getriggert wird, um parallel die tatsächlich vorliegende Spannung zu messen.

4.6 Mikrocontroller

In diesem Abschnitt wird die Umsetzung der Sensormessung mit Mikrocontrollern erläutert. Nach einer allgemeinen Beschreibung wird zuerst auf die Hardware und anschließend auf die Software eingegangen. Abschließend werden noch Besonderheiten einzelner MCUs näher betrachtet.

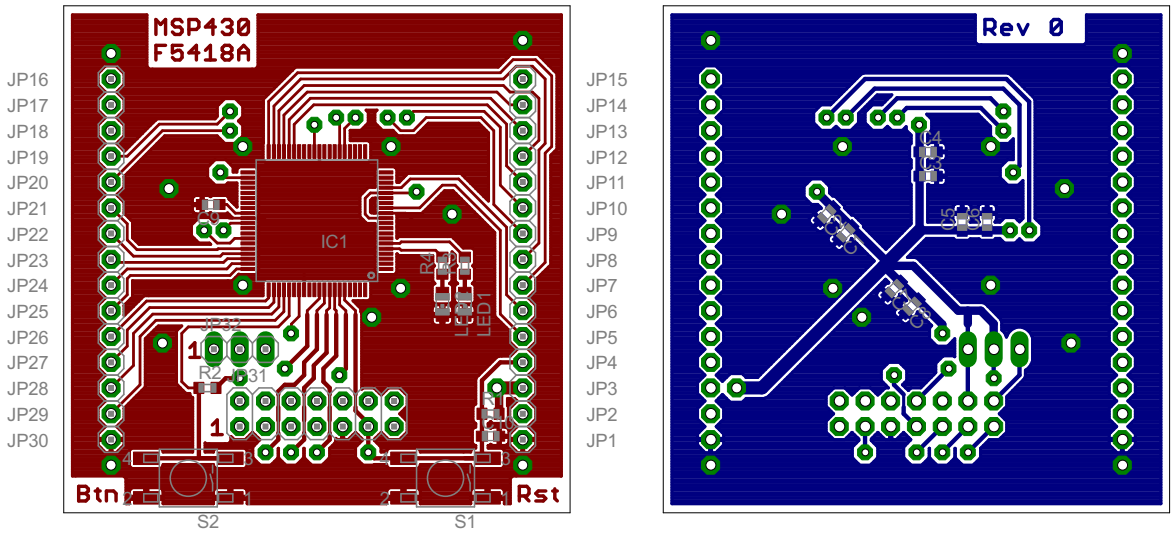
Für diese Arbeit wurden 5 Mikrocontroller ausgewählt, welche von den Herstellern als Ultra-Low-Power Mikrocontroller konzipiert wurden. Für diese Mikrocontroller wurde bereits eine theoretische Analyse des Energieverbrauches in [GHDG09] durchgeführt. Diese Mikrocontroller sind:

- **MSP430F1232** von Texas Instruments
- **MSP430F2232** von Texas Instruments
- **MSP430F5418A** von Texas Instruments
- **ATmega88PA** von Atmel
- **PIC16LF727** von Microchip

Die Mikrocontroller von Texas Instruments stammen aus unterschiedlichen Unterfamilien der MSP430 Architektur, mit unterschiedlichem Funktionsumfang. Darum wurden alle drei Mikrocontroller von Texas Instruments in den Vergleich mit aufgenommen.

Die gewählten Mikrocontroller verfügen über ein integriertes SPI-Modul. Dieses wird benutzt um den SPI-Sensor Emulator mit dem Protokoll des Temperatursensors ADT7310 abzufragen. Abbildung 4.5 zeigt das Sensorprotokoll. Um die in Abschnitt 3.4 aufgezählten Parameter einzustellen, wurde das Mikrocontroller Programm mit entsprechenden Konfigurationsroutinen ausgestattet.

Für jeden der erwähnten Mikrocontroller wurde ein Hardware-Modul entwickelt und gefertigt, welches kompatibel zum MCU-Steckplatz des SNOPS Eval-Boards ist. Abbildung 4.8 zeigt als Beispiel das Platinen-Layout für den MSP430F5418A mit Top- und Bottom-Layer. Die beiden 15-poligen Stiftleisten am Rand der Platine bilden die Schnittstelle zum MCU-Sockel. Außerdem verfügt jede Platine neben dem Mikrocontroller auch über eine entsprechende Programmierschnittstelle, zwei Light-Emitting Diodes (LEDs), einen Reset-Taster und einen frei belegbaren Taster. Die Herausforderung beim Design der Platinen war es, alle benötigten Bauteile und Verbindungen auf den vorgegebenen Maßen von 5 cm mal 5 cm unter Berücksichtigung des Fertigungsprozesses unterzubringen. Durch die Fertigung in der fakultätseigenen Werkstätte mussten die Designrules restriktiver gewählt werden, als dies bei kommerziell gefertigten Platinen üblich ist. Einerseits mussten die Abstände zwischen Leiterbahnen vergrößert und die Lötpunkte für Durchkontaktierungen vergrößert werden, um die größeren Fertigungstoleranzen zu kompensieren. Andererseits gab es fertigungstechnisch keine Möglichkeit metallisierte Durchkontaktierungen herzustellen, wodurch zusätzliche Durchkontaktierungen bei einigen Stiftleisten notwendig wurden.



(a) Top-Layer

(b) Bottom-Layer

Abbildung 4.8: Platine des MSP430F5418A

Die Firmware der Mikrocontroller wurde in C geschrieben. Dadurch war es möglich, sowohl das Protokoll für die Sensormessung mit dem ADT7310, als auch die anschließende Auswertelogik zu vereinheitlichen. Angepasst an die jeweilige MCU wurden lediglich die verwendeten Device-Treiber, unter Verwendung eines einheitlichen Interface analog zum Konzept eines Hardware Abstraction Layer (HAL). Durch Programmierung der Mikrocontroller in Assembler wäre noch weiteres Optimierungspotenzial vorhanden gewesen [vgl. Dav08, S. 55]. Aufgrund der fehlenden Portierbarkeit wäre jedoch auch der Entwicklungsaufwand wesentlich größer gewesen.

Der Programmablauf enthält zwei Teilbereiche. Die Initialisierung und Konfiguration des Mikro-

controllers einerseits und der Ablauf der Sensormessung andererseits. Während der Initialisierung, welche nach jedem Reset ausgeführt wird, werden alle benötigten Module des Mikrocontrollers betriebsbereit gemacht. Dazu zählt auch die serielle Schnittstelle, über welche die Konfigurationsparameter der Steuerung des Messsystems empfangen werden. Dazu sendet die MCU zuerst das Zeichen 'I' (Initialized) um zu signalisieren, dass das Device bereit ist, die Konfigurationsdaten zu empfangen. Anschließend werden von der Steuerung aus die gewünschten Parameter als Zeichen codiert übertragen. Wenn der Mikrocontroller das Zeichen 'C' (Continue) empfängt wird die Konfiguration fortgesetzt. Dabei werden nicht benötigte Ressourcen deaktiviert sowie unbenutzte Pins auf Masse gelegt. Wenn der Mikrocontroller vollständig initialisiert ist, wird das Zeichen 'R' (Ready) an die Steuerung gesendet und der Mikrocontroller setzt mit dem zweiten Teil des Programms, der Sensormessung, fort.

Die Sensormessung ist als Pseudocode in Algorithmus 4.1 angegeben. Mikrocontroller-spezifische Routinen wurden dabei als einzelne Befehle abstrahiert, zB. „Sende SPI()“. Am Beginn des Programms steht die bereits erwähnte Initialisierung. Anschließend startet die Sensormessung in einer Endlosschleife. An der Stelle der Subroutine des ADT7310 könnte ein Protokoll eines weiteren Sensors, für zukünftige Untersuchungen, eingesetzt werden. Zur Übertragung der Kommandos über SPI wurde kein Sende- oder Empfangsinterrupt gesetzt. Grund dafür ist, dass der Mikrocontroller keine weitere Aufgabe hat, welche er zwischenzeitlich bearbeiten könnte. Außerdem verursacht der Sprung in die Interruptroutine zusätzlichen Overhead. Nachdem der Sensorwert abgefragt wurde, kann der Vergleich mit dem bisherigen Wert durchgeführt werden. Wenn der Absolutbetrag der Differenz beider Werte den Schwellwert überschreitet, wird der neue Sensorwert gespeichert und gegebenenfalls weiter verarbeitet. In dieser Realisierung erfolgt keine weitere Verarbeitung. Nach einem Zyklus geht der Mikrocontroller in den LPM. Durch einen Timer-Interrupt wird der Mikrocontroller wieder aktiviert und die Sensormessung startet erneut. Durch das Protokoll des ADT7310 bedingt, wird bereits beim Erfassen des Sensorwertes einmal in den LPM gewechselt. Aufgrund der verschiedenen MCUs und deren unterschiedliche Timer wird der Timer-Interrupt nur zur Reaktivierung aus dem LPM verwendet.

In den folgenden Unterabschnitten wird näher auf die Besonderheiten der einzelnen Mikrocontroller-Typen und Erfahrungen bei der Inbetriebnahme eingegangen.

MSP430F1232
Die Besonderheit am MSP430F1232 war die Überwachung des Oszillator. Nachdem der Mikrocontroller durch den Timer-Interrupt aktiviert wurde, war das Oscillator-Fault-Flag gesetzt und der Chip hat auf den internen Oszillator umgeschaltet. Dieses Verhalten äußerte sich bei der Messung in einem von der Taktfrequenz unabhängigen Stromverbrauch. Zur Lösung musste nach jedem Interrupt in der Interrupt-Routine das entsprechende Flag gelöscht werden. Die Analyse der ersten Messergebnisse zeigte einen erhöhten statischen Strombedarf im LPM bei aktiviertem Takt, siehe Abschnitt 5.1.2. Die Ursache dafür ist wahrscheinlich, dass das Taktnetz nicht direkt beim Takteingang abgeschaltet wird.

MSP430F2232
Beim MSP430F2232 trat der gleiche Fehler des Oszillator wie beim MSP430F1232 auf. Durch eine Anpassung in der Interrupt-Routine wurde der Fehler ebenfalls behoben. Im Gegensatz zum zuvor genannten Chip zeigte der MSP430F2232 allerdings keinen erhöhten Stromverbrauch im LPM.

MSP430F5418A
Beim MSP430F5418A wurde im Gegensatz zu den beiden anderen Mikrocontrollern von Texas Instruments keinen Oszillator-Fehler registriert, jedoch wies der Chip ebenfalls einen erhöhter Stromverbrauch im LPM bei aktivem Takt auf.

```
Initialisiere
Wiederhole für immer
    SensorVersorgung(Ein)
    // Hole Sensorwert
    Beginn
        // Subroutine für ADT7310
        Slave Select(Ein)
        Sende SPI(0x08)
        Sende SPI(0x20)
        Slave Select(Aus)
        Aktiviere LPM()
        Slave Select(Ein)
        Sende SPI(0x50)
        Sende SPI(0xFF)
        Sensorwert ← SPI Empfangs-Register um 5 Bits nach Links geshiftet
        Sende SPI(0xFF)
        Sensorwert ← Sensorwert ∪ (SPI Empfangs-Register um 3 Bits nach Rechts geshiftet)
        Slave Select(Aus)
    Ende
    SensorVersorgung(Aus)
    Differenz ← Sensorwert − alter Sensorwert
    Differenz ← Absolutbetrag(Differenz)
    wenn Differenz > Schwellwert dann
        alter Sensorwert ← Sensorwert
        // Weitere Verarbeitung des Sensorwertes
    Aktiviere LPM()
```

Algorithmus 4.1 : Pseudocode des Mikrocontroller Programms

ATmega88PA

Der ATmega88PA ist nicht für einen Betrieb mit zwei externen Takten ausgelegt. Während die anderen Mikrocontroller einen Timer als asynchronen Eventzähler verwenden können, wird der externe Eventeingang im ATmega88PA abgetaktet. Im LPM fehlt der dazu notwendige Takt und der Chip kann den LPM nicht mehr verlassen. Darum wurde beim ATmega88PA eine andere Herangehensweise gewählt. Ein Pin des Chips wurde als externer Interrupteingang definiert, welcher auf steigende und fallende Flanken reagiert. Zusätzlich war eine Ergänzung im FPGA des SmartFusion Chips notwendig, um die Impulse gemäß der Zykluszeit zu erzeugen. Für ein reales Sensormesssystem wäre der ATmega88PA jedoch geeignet, da der Chip über einen internen Oszillator verfügt und der externe Takteingang für den RTC-Takt verwendet werden kann.

PIC16LF727

Der Timer des PIC16LF727 verfügte nicht über eine entsprechende Reload-Funktion. Darum musste diese manuell implementiert werden und die Zykluszeit verlängerte sich. Darum benötigte der Mikrocontroller eine Kompensation der Taktfrequenz um die vorgegebenen Zykluszeiten einzuhalten. Mit dem Oszilloskop wurde die Differenz zur erwarteten Zykluszeit bei jeder zu messenden Taktfrequenz bestimmt und anschließend die Kompensation der entsprechenden Anzahl RTC-Zyklen in der Initialisierungsroutine des Mikrocontrollers codiert.

4.7 Testchip

Der Testchip ist die Umsetzung des autonomen, rekonfigurierbaren Moduls in Hardware, wie bereits in Abschnitt 2.4 beschrieben. Der Testchip wird wie die MCUs für das Sensorprotokoll des ADT7310 [Ana09] konfiguriert. Wesentlicher Unterschied zu den Mikrocontrollern ist jedoch der Betriebsbereich des Testchip [GBW+11].

- Die Core-Versorgungsspannung ist von 0,8 V bis 1,5 V definiert und in diesem Bereich wird der Testchip gemessen.
- Die Pad-Versorgungsspannung ist mit 2,5 V bis 3,3 V spezifiziert. Durch die getrennte Versorgung von Core und Pads kann die Pad-Versorgungsspannung auf 3,3 V fixiert werden und es werden keine Pegelumsetzer zwischen Pads und SmartFusion Chip benötigt.
- Die Taktfrequenz des Testchip ist von 0 bis 25 MHz definiert. Wie bereits in Abschnitt 3.4 erwähnt, steigt der Stromverbrauch des Testchip mit steigender Frequenz. Darum wird in der Messung der Bereich von 50 kHz bis 10 MHz vorgegeben.

Die Konfiguration des Testchip erfolgte mit folgendem Ablauf. Zu Beginn werden die Parameter wie Schwellwert, Zykluszeit, SPI-Wartezeit, SPI-Teiler und Taktteiler der Taktfrequenz von der Steuerung des Messsystems an den SmartFusion Prozessor übertragen. Anschließend wurde der Testchip resettiert um allfällige Fehlkonfigurationen der vorangegangenen Messung nicht mitzuschleppen. Darauf folgte die Konfiguration des Testchip durch einen vorgefertigten Bitstream, welcher mittels JTAG vom SmartFusion Prozessor aus übertragen wurde.

Nach erfolgter Konfiguration konnten die variablen Parameter übertragen werden. Dazu musste die Takterzeugung einen 1 MHz Takt bereit stellen, da das Baudratenregister des Testchip für 9600 Baud bei 1 MHz durch den Bitstream vorkonfiguriert war. In weiterer Folge konnten die Konfigurationsparameter für Schwellwert, Zykluszeit, SPI-Wartezeit, SPI-Teiler und die SPI-Kommando Bytes gesetzt werden. Anschließend wurde der Takt ausgeschaltet, der von der Steuerung übertragene Taktteiler eingestellt und die Takterzeugung aktiviert. Damit war die Konfiguration und Parametrierung abgeschlossen und die Messung konnte gestartet werden.

Der Bitstream wurde von Herrn Glaser bereit gestellt, da zum Zeitpunkt der Messung die in Abschnitt 2.4 erwähnten Verbesserungen hinsichtlich des Workflows noch nicht verfügbar waren.

4.8 Zeitmessung

Durch die Zeitmessung wurde ein zusätzlicher Wert zur Charakterisierung der MCUs gemessen. Die Messung erfolgte manuell mit einem Oszilloskop vor der automatisierten Messung. Abbildung 4.9 zeigt einen vollständigen SPI-Sensor Zyklus mit entsprechender Aktivierung anhand des PIC16LF727. Kanal 2 und 3 stellen den Spannungsverlauf am Messshunt des Amperemeters dar. Der Kanal M zeigt die Differenz der Kanäle 2 und 3 und kann damit direkt auf den Stromverbrauch umgelegt werden. Kanal 1 zeigt als Referenz dazu ein Signal, welches vom Mikrocontroller direkt erzeugt wird. Dazu wurde das MCU-Programm angepasst und ein Pin unmittelbar in der Interruptroutine auf „high“ gesetzt und direkt vor dem Übergang in den LPM auf „low“ gesetzt.

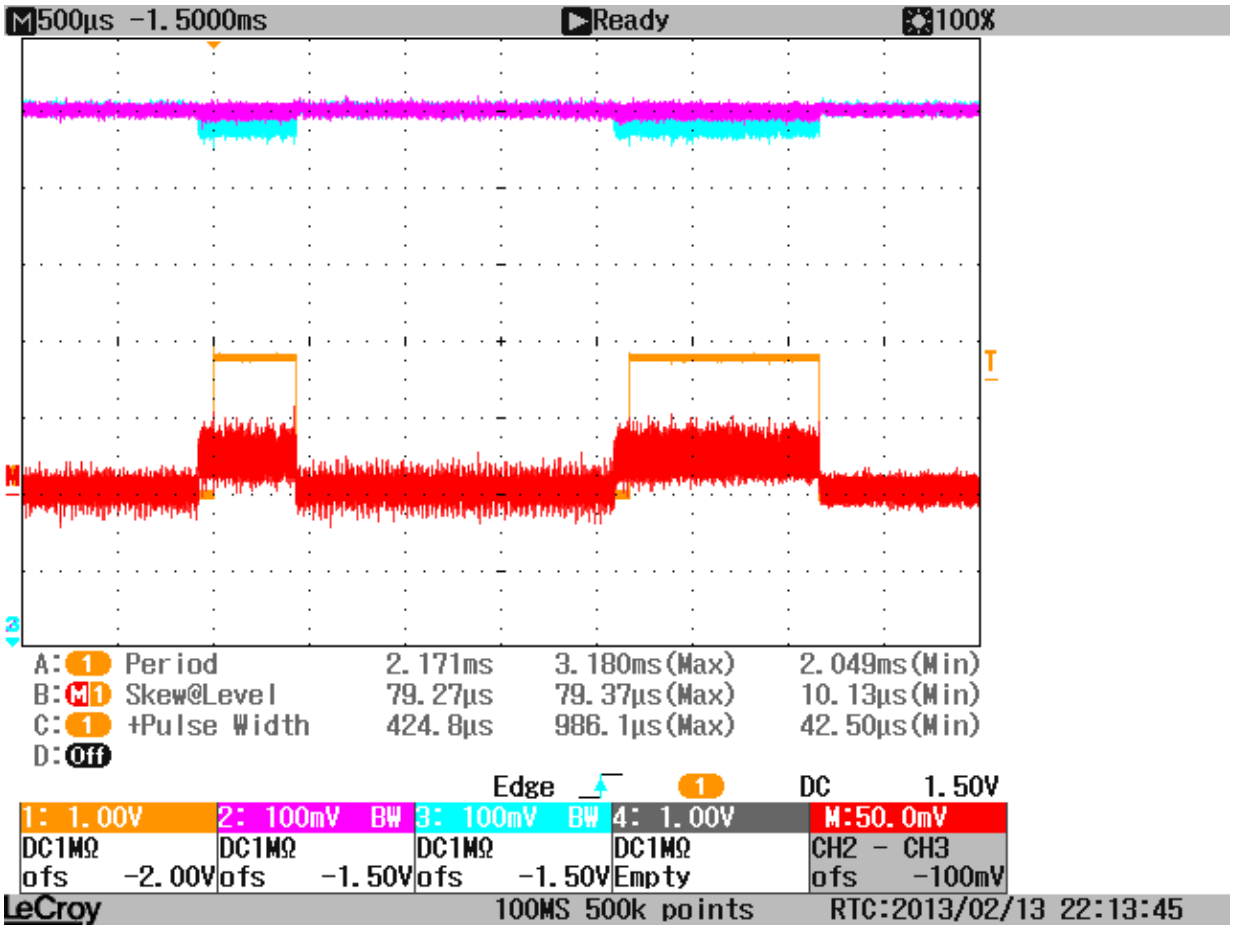


Abbildung 4.9: Aktive Zeit des Mikrocontrollers mit Interruptlatenz

Der Anstieg des Differenzsignals vor dem Interruptsignal zeigt die Verzögerung, welche durch die Aktivierung der MCU nach einem Interrupt entsteht, bis zur Ausführung des ersten Befehls. Wesentlich ist dabei anzumerken, dass die Stützkondensatoren an den Mikrocontroller Pins entfernt werden mussten. Bei ersten Tests mit Stützkondensatoren zeigte sich, dass der Anstieg des Differenzsignals erst nach dem Interruptsignal auftreten konnte. Bei einigen Mikrocontrollern war der anfängliche Anstieg des Stromverbrauches insbesondere bei geringer Versorgungsspannung nicht erkennbar. Darum wurden alle Messungen bei maximaler Betriebsspannung durchgeführt. Um den Messablauf zu beschleunigen, wurde ein eigenes Parameterset für den semi-automatischen Messablauf erstellt, welches alle relevanten Frequenzwerte durchgeht und erst nach Tastendruck den folgenden Messpunkt einstellt.

5 Messergebnisse

In diesem Kapitel werden die Messergebnisse vorgestellt, die bei der automatisierten Messung ermittelt wurden. Die Ergebnisse werden dabei in zwei Kategorien aufgeteilt. Einerseits in den Bereich der Mikrocontroller in Abschnitt 5.1 und andererseits die Messung des Testchips in Abschnitt 5.2.

Die gezeigten Grafiken sind mit MATLAB direkt aus den gemessenen Daten entstanden. Dabei ist zu beachten, dass das Regressions-Modell über alle Vorgabe-Parameter (Versorgungsspannung (U), Taktfrequenz (f), Anzahl der Sensormessungen (n), Anzahl der Schwellwertüberschreitungen (m) und SPI-Busteiler (d)) gerechnet ist. Aufgrund dessen ergibt sich ein hohes Bestimmtheitsmaß R^2 obwohl die grafische Darstellung des Modells in einzelnen Diagrammen teils deutlich vom gemessenen Wert abweicht.

Das Bestimmtheitsmaß R^2 ist eine Kennzahl für den linearen Zusammenhang zwischen der abhängigen und den unabhängigen Variablen. In den Diagrammen wird bei einem Bestimmtheitsmaß von über 0,99 abgerundet, um nicht den Eindruck einer perfekten Passung trotz kleiner Abweichungen zu erwecken.

Die Formeln (5.1) und (5.2) bilden die Grundlage der Regressionberechnung. Die Formeln mussten gegenüber der in Abschnitt 3.6 entwickelten Formel (3.19) angepasst werden, da sich die Signifikanz der Prediktoren der Parameter, Anzahl der Sensormessungen, Anzahl der Schwellwertüberschreitungen und SPI-Busteiler, bei den ermittelten Messwerten als zu gering herausstellte. Die Koeffizienten deren Bedeutung sich geändert hat, werden mit einem Akzent nach der Nummer dargestellt.

Die Tabellen 5.1 und 5.2 zeigen die ermittelten Koeffizienten und das Bestimmtheitsmaß der gemessenen Mikrochips. Das zweite Modell wurde aufgrund der Charakteristik des PIC16LF727 entwickelt um ein höheres Bestimmtheitsmaß zu erreichen und wurde nur bei diesem Chip näher betrachtet, da die grafische Passung bei den anderen Mikrocontrollern mit Modell 1 besser war, obwohl zum Teil geringfügig besseres Bestimmtheitsmaß errechnet wurden. Stellvertretend für die 12 gemessenen Testchips wurde ein Testchip in die Tabelle mit aufgenommen. In den Tabellen sind die für die grafische Darstellung nicht verwendeten Typen in Klammern gesetzt.

$$I_\mathrm{T} = b_1 + b_2 \cdot U + b_3 \cdot Uf + b_4' \cdot U \cdot n + b_5' \cdot m + b_6' \cdot U \cdot d + b_7 \cdot U^2 f \tag{5.1}$$

$$I_\mathrm{T} = b_1 + b_2 \cdot U + b_3 \cdot Uf + b_4' \cdot U \cdot n + b_5' \cdot m + b_6' \cdot U \cdot d + b_7' \cdot U^2 \frac{1}{f} \tag{5.2}$$

M1	ATmega88PA	MSP430F1232	MSP430F2232	MSP430F5418A	(PIC16LF727)	Testchip07	
b_1	−0,2811	−2,237	−2,258	11,26	−15,02	−1,281	µA
b_2	−22,02	−1,232	−1,394	−4,723	11,75	2,582	µA/V
b_3	−1,008	2,922	−0,1229	3,225	−0,496	23,33	pA/(V·Hz)
b_4'	202,5	166,7	215,4	205,7	349,3	−0,4889	nA/V
b_5'	5,669	−1,956	1,227	26,55	71,15	2,166	nA
b_6'	5926	671,3	716,4	753,4	0	11,36	nA/V
b_7	225,2	229,8	16,87	−524,7	−235,7	837,8	fA/(V^2·Hz)
b_7'	0	0	0	0	0	0	mA·Hz/V^2
R^2	0,9746	0,9994	0,9961	0,9883	0,8008	0,9999	

Tabelle 5.1: Regressionskoeffizienten Modell 1

M2	(ATmega88PA)	(MSP430F1232)	(MSP430F2232)	(MSP430F5418A)	PIC16LF727	(Testchip07)	
b_1	−1,39	−6,27	−1,593	28,77	2,933	−2,615	µA
b_2	−23,03	1,048	−2,232	−13,96	−5,886	3,665	µA/V
b_3	−0,1921	3,396	−0,01311	2,082	−0,07223	24,42	pA/(V·Hz)
b_4'	203,4	166,9	216,1	208,2	359,9	−1,444	nA/V
b_5'	3,182	−2,555	−1,22	17,98	38,27	5,01	nA
b_6'	5918	668,9	709,6	731,5	0	17,39	nA/V
b_7	0	0	0	0	0	0	fA/(V^2·Hz)
b_7'	531,4	−219,4	246,8	1043	4484	−1,607	mA·Hz/V^2
R^2	0,9760	0,9990	0,9974	0,9818	0,8894	0,9999	

Tabelle 5.2: Regressionskoeffizienten Modell 2

5.1 Mikrocontroller

In diesem Abschnitt werden der Reihe nach die einzelnen Ergebnisse der Messung der Mikrocontroller besprochen. Außerdem wird auf Besonderheiten der einzelnen Typen eingegangen. Der betrachtete Messzeitraum T beträgt bei allen Messungen 200 ms. Die auf die jeweiligen Parameter reduzierten Modellgleichungen sind unterhalb der Abbildungen angegeben.

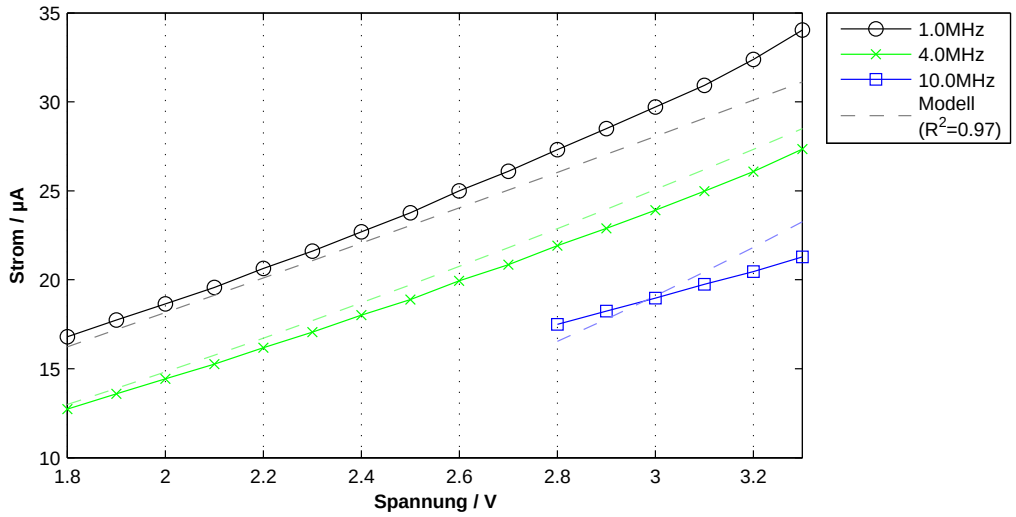
5.1.1 ATmega88PA

Die Messergebnisse für den ATmega88PA werden in den folgenden Abbildungen dargestellt. Abbildung 5.1 zeigt den Stromverbrauch abhängig von der Versorgungsspannung. Auf Abbildung 5.2 wird der Stromverbrauch in Abhängigkeit von der verwendeten Taktfrequenz gezeigt. Beide Grafiken zeigen den Verbrauch bei 40 Sensormessungen im betrachten Messzeitraum. Der sichere Betriebsbereich des Mikrocontrollers (ab 1,8 V max. 4 MHz, ab 2,7 V max. 10 MHz, dazwischen linear interpoliert [Atm11, S. 322]) umschließt nicht alle untersuchten Spannungs-Frequenz-Punkte, weswegen die Graphen für 10 MHz, 1,8 V und 2,5 V nicht die komplette Breite des Diagramms einnehmen. Deutlich erkennbar ist ein sinkender Verbrauch mit steigender Frequenz. Dies widerspricht zwar Gleichung (2.2), ist nach der Analyse in Abschnitt 3.4 im Unterpunkt Taktfrequenz, durch die kürzere Ausführungszeit des Programmcodes zu begründen.

In Abbildung 5.3 ist der Stromverbrauch abhängig von der Anzahl der Sensormessungen im Messzeitraum $T = 200\,\text{ms}$ aufgetragen. Die Abbildung 5.4 zeigt den Zusammenhang zwischen dem Strom und der Anzahl der Schwellwertüberschreitungen bei 40 Sensormessungen pro Messzeitraum. Aufgrund des geringen Beitrages zum Gesamtverbrauch wird bei allen weiteren Mikrocontrollern auf die Darstellung des Diagramms, Stromverbrauch über Schwellwertüberschreitungen, verzichtet. Abbildung 5.5 stellt den Strom abhängig vom SPI-Busteiler dar. Durch einen größeren SPI-Busteiler wird die SPI-Taktfrequenz reduziert was in einer längeren aktiven Zeit der MCU und einem höheren Durchschnittsstrom resultiert.

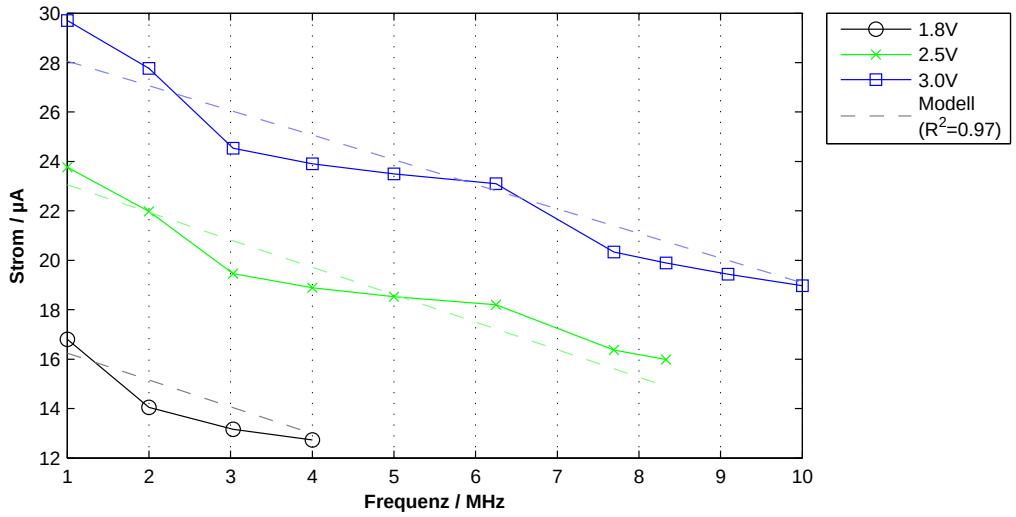
Die Abbildungen 5.6 und 5.7 zeigen den Stromverbrauch im Low-Power-Mode mit aktivem Takt und den statischen Verbrauch. Wie in Abschnitt 4.6 beschrieben, wird der ATmega88PA durch einen externen Interrupt aktiviert. Um die Darstellung des statischen Stromes konsistent über alle Mikrocontroller zu halten, sind zwei Messreihen angegeben, welche unter den gleichen Bedingungen gemessen wurden. Die geringen Abweichungen zwischen beiden Messungen sind auf geringfügige Spannungsunterschiede zwischen Mikrocontroller-Versorgung und Pegelwandler Versorgungsspannung zurückzuführen. Diese wirken sich insgesamt negativ auf die Messung aus und bedingen die sprungartigen Änderungen des statischen Stromes. Des Weiteren fällt im direkten Vergleich der beiden Abbildungen auf, dass der Stromverbrauch bei aktivem Takt mit steigender Frequenz und steigender Spannung abnimmt. Genaue Untersuchungen dieses Sachverhaltes haben ergeben, dass die Probleme auf den Pegelumsetzer der Taktleitung zurückgehen. Dieser macht sich durch ein spannungsabhängiges Überschwingen bemerkbar. Durch das erreichen der Flussspannung der Eingangs-ESD-Diode wird damit die Mikrocontroller Versorgung gespeist. Ein RC-Dämpfungsglied zwischen Takteingang und Masse reduziert diesen Effekt, eliminiert ihn jedoch nicht vollständig.

Abbildung 5.8 zeigt den Zusammenhang zwischen Frequenz und aktiver Periode des Mikrocontrollers. Die Werte wurden dabei manuell mit dem Oszilloskop ermittelt. Im Gegensatz zu der in Abschnitt 4.8 angegebenen Methode, kann aufgrund der externen Interrupt-Quelle die Interruptlatenz direkt gemessen werden.



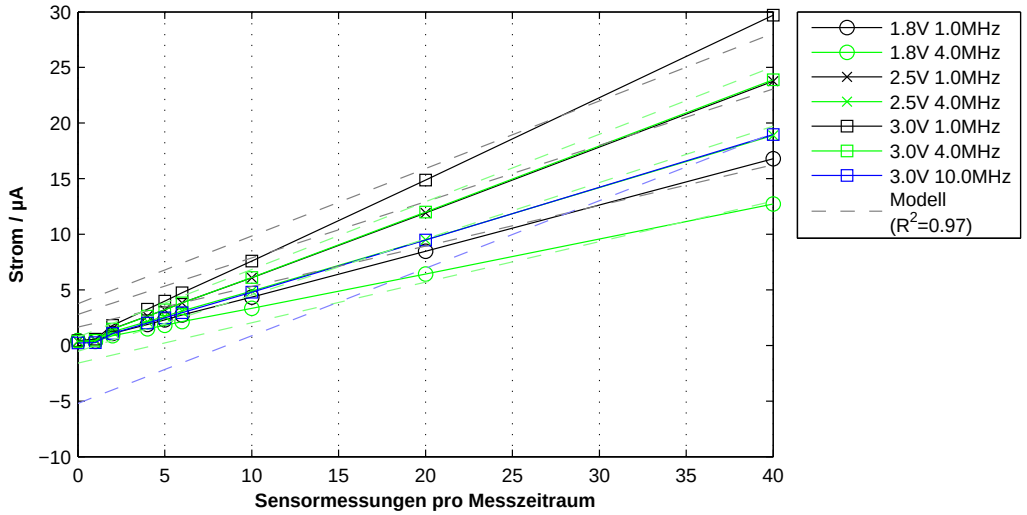
Modell: $I(U,f) = -2.811 \cdot 10^{-7} + 9.778 \cdot 10^{-6} \cdot U - 1.008 \cdot 10^{-12} \cdot U \cdot f + 2.252 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.1: ATmega88PA: Strom abhängig von der Spannung bei 40 Sensormessungen pro Messzeitraum



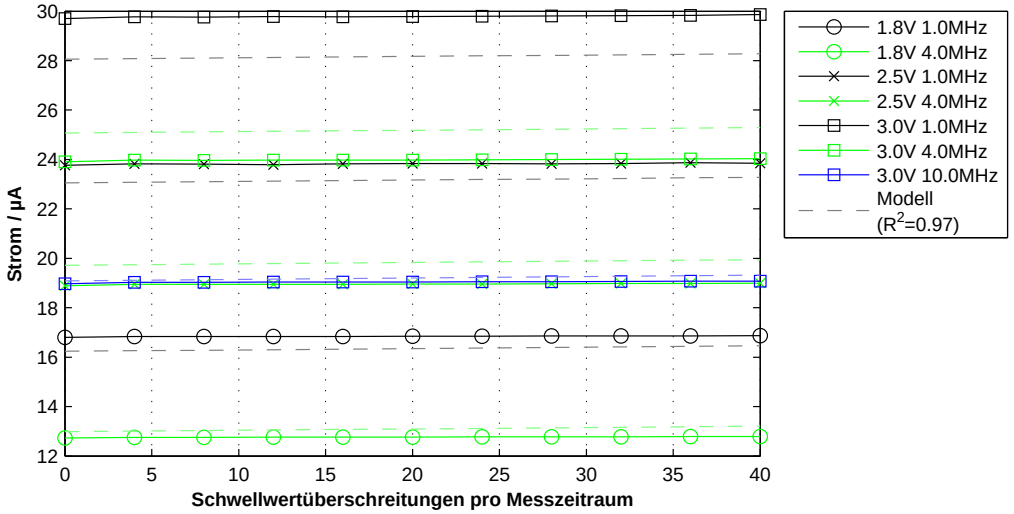
Modell: $I(U,f) = -2.811 \cdot 10^{-7} + 9.778 \cdot 10^{-6} \cdot U - 1.008 \cdot 10^{-12} \cdot U \cdot f + 2.252 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.2: ATmega88PA: Strom abhängig von der Frequenz bei 40 Sensormessungen pro Messzeitraum



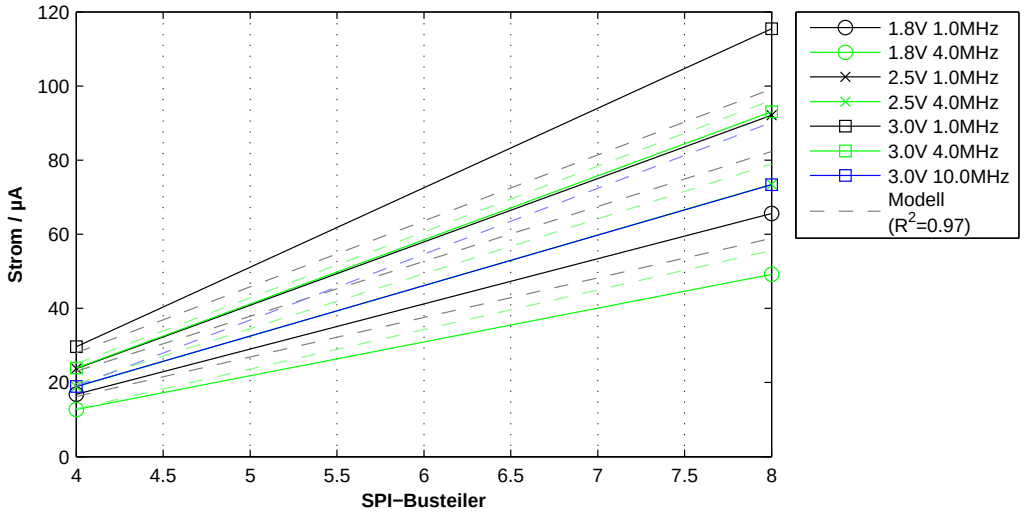
Modell: $I(U, f, n) = -2.811 \cdot 10^{-7} + 1.678 \cdot 10^{-6} \cdot U + 2.025 \cdot 10^{-7} \cdot U \cdot n - 1.008 \cdot 10^{-12} \cdot U \cdot f + 2.252 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.3: ATmega88PA: Strom abhängig von der Anzahl der Sensormessungen pro Messzeitraum



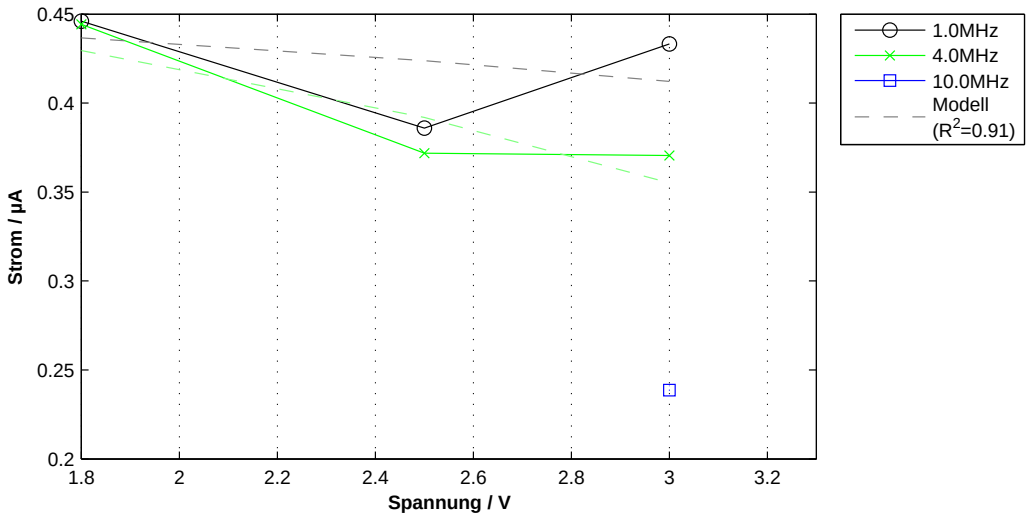
Modell: $I(U, f, m) = -2.811 \cdot 10^{-7} + 9.778 \cdot 10^{-6} \cdot U - 1.008 \cdot 10^{-12} \cdot U \cdot f + 2.252 \cdot 10^{-13} \cdot U^2 \cdot f + 5.669 \cdot 10^{-9} \cdot m$

Abbildung 5.4: ATmega88PA: Strom abhängig von der Anzahl der Schwellwertüberschreitungen pro Messzeitraum



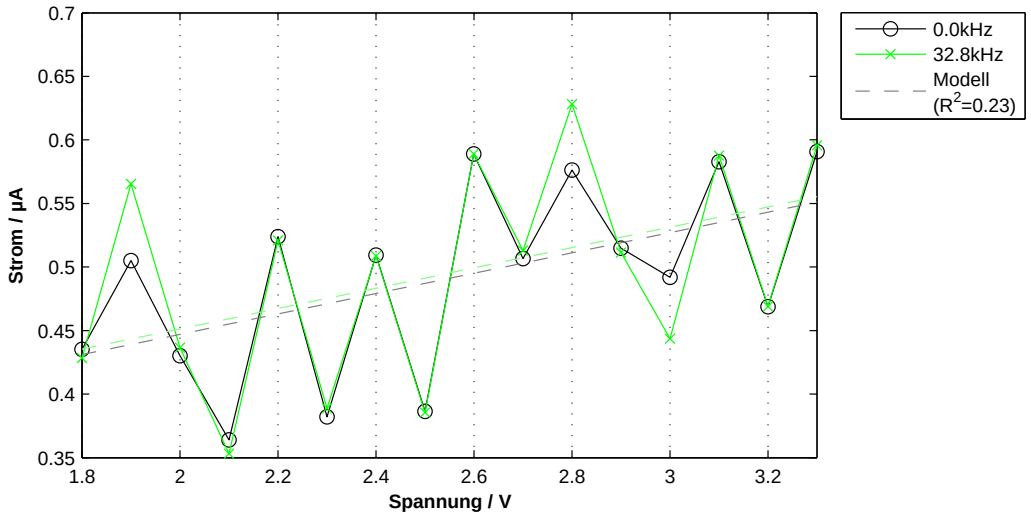
Modell: $I(U, f, d) = -2.811 \cdot 10^{-7} - 1.393 \cdot 10^{-5} \cdot U - 1.008 \cdot 10^{-12} \cdot U \cdot f + 5.926 \cdot 10^{-6} \cdot U \cdot d + 2.252 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.5: ATmega88PA: Strom abhängig vom SPI-Busteiler bei 40 Sensormessungen pro Messzeitraum



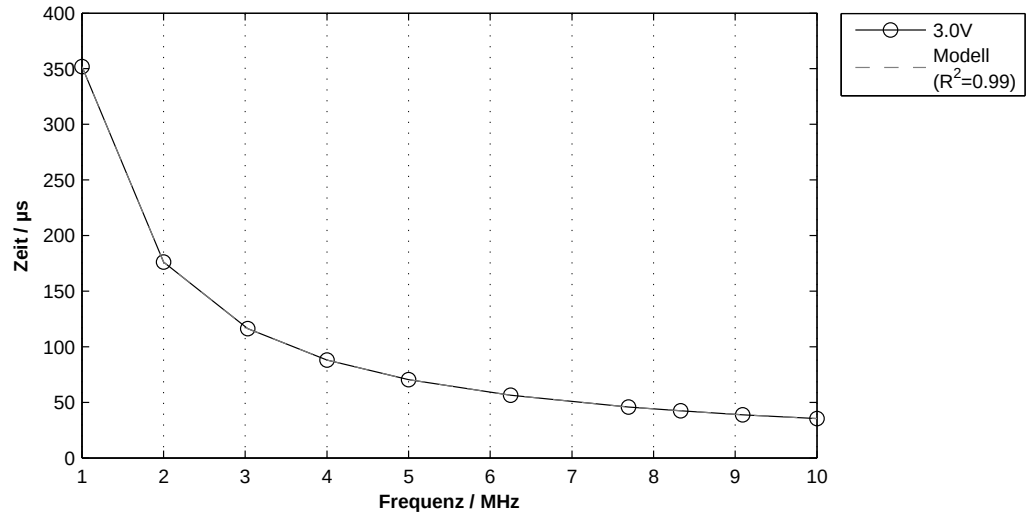
Modell: $I_{lp}(U, f) = +4.509 \cdot 10^{-7} - 6.532 \cdot 10^{-9} \cdot U + 6.169 \cdot 10^{-15} \cdot U \cdot f - 4.170 \cdot 10^{-15} \cdot U^2 \cdot f$

Abbildung 5.6: ATmega88PA: Strom abhängig von der Spannung im Low-Power-Mode bei aktivem Takt



Modell: $I_{st}(U, f_{RTC}) = +2.871 \cdot 10^{-7} + 7.998 \cdot 10^{-8} \cdot U + 1.293 \cdot 10^{-13} \cdot f_{RTC}$

Abbildung 5.7: ATmega88PA: statischer Strom abhängig von der Spannung



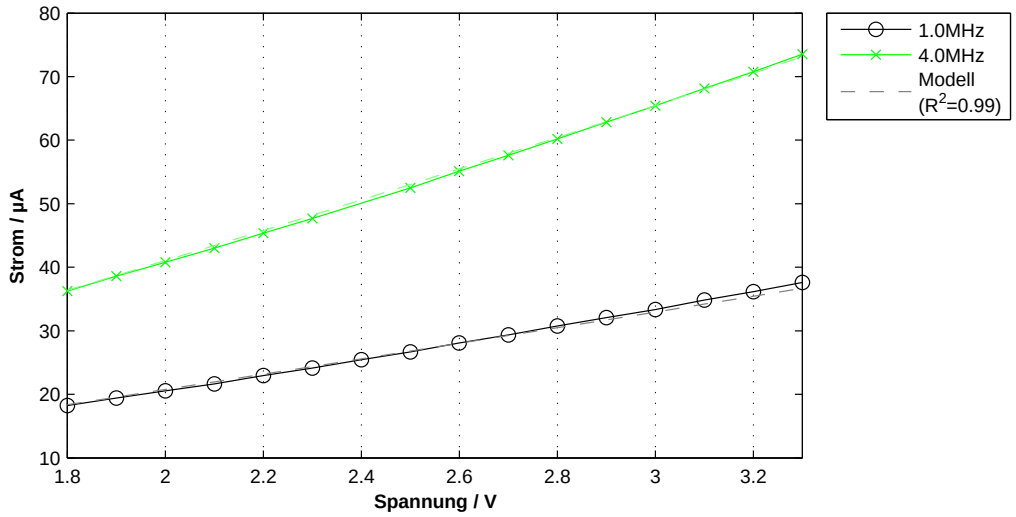
Modell: $t(f) = +1.119 \cdot 10^{-7} + 3.518 \cdot 10^{2} \cdot \frac{1}{f}$

Abbildung 5.8: ATmega88PA: aktive Zeit abhängig von der Frequenz

5.1.2 MSP430F1232

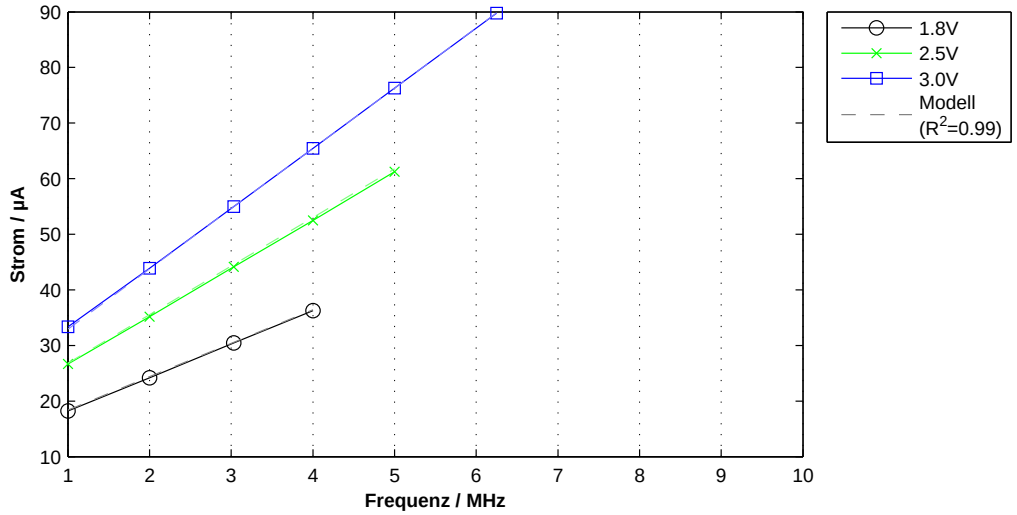
Die Messergebnisse für den MSP430F1232 werden in den folgenden Abbildungen dargestellt. Abbildungen 5.9 und 5.9 zeigen den Stromverbrauch in Abhängigkeit von Spannung bzw. Frequenz. Der sichere Betriebsbereich des Mikrocontrollers (ab 1,8 V max. 4,15 MHz, ab 3,6 V max. 8 MHz, dazwischen linear interpoliert [Tex04, S. 17]) beinhaltet nicht alle untersuchten Spannungs-Frequenz-Punkte. Der Graph für 10 MHz entfällt somit komplett. Außerhalb seines Betriebsbereiches kam es zu wiederholten Fehlmessungen, weswegen nur der als sicher definierte Bereich untersucht wurde. Das Modell passt sehr gut zu den gemessenen Werten. Insgesamt zeigt der Chip zu erwartendes Halbleiterverhalten, d. h. steigender Stromverbrauch mit höherer Frequenz. Dies ist im wesentlichen auf das nicht deaktivierte Taktnetz im LPM zurückzuführen wie im nächsten Absatz erläutert wird.

Abbildung 5.11 zeigt den Strom abhängig von der Anzahl der Sensormessungen pro Messzeitraum. In Abbildung 5.12 ist der Stromverbrauch in Abhängigkeit von der Versorgungsspannung im Low-Power-Mode bei aktivem Takt dargestellt. Im Vergleich der Abbildung 5.12 mit Abbildung 5.13, welche den statischen Stromverbrauch zeigt, erkennt man einen deutlich erhöhten Verbrauch im LPM bei aktivem Takt. Dieser ist durch den speziellen Messaufbau mit externer Taktleitung und der Funktionsweise des Taktnetzes im Mikrocontroller begründet. Während der Oszillator bei Betrieb mit Quarz gänzlich abgeschaltet wird, wird der externe Takt nur durchgeleitet. Dadurch wird das Taktnetz versorgt und erst unmittelbar vor dem Rechenwerk und den Peripherie-Taktnetzen deaktiviert. Der statische Stromverbrauch in Abbildung 5.13 kann damit nahezu vernachlässigt werden, während der Timer, der mit der RTC-Taktfrequenz betrieben wird, in etwa 450 nA verbraucht. Abbildung 5.14 zeigt die aktive Zeit des Mikrocontrollers abhängig von der Frequenz.



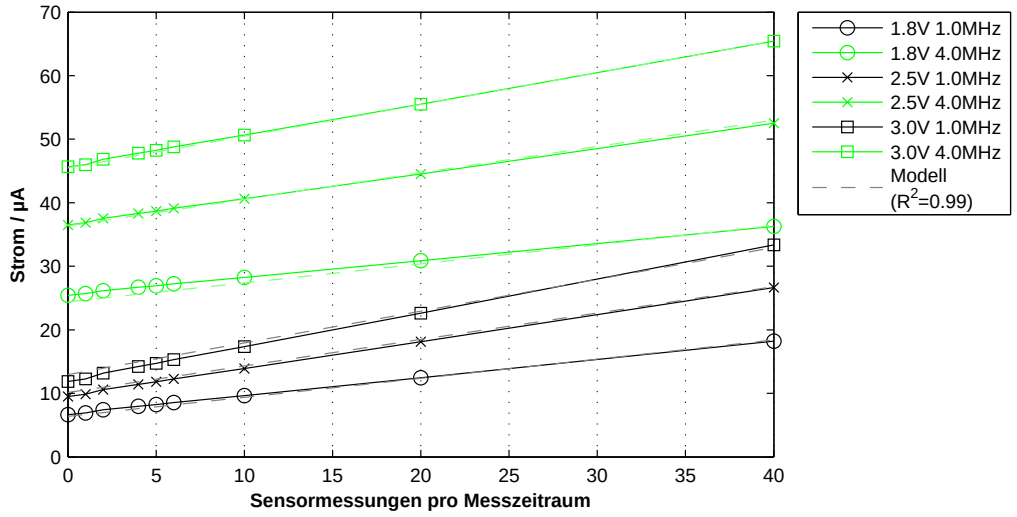
Modell: $I(U,f) = -2.237 \cdot 10^{-6} + 8.119 \cdot 10^{-6} \cdot U + 2.922 \cdot 10^{-12} \cdot U \cdot f + 2.298 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.9: MSP430F1232: Strom abhängig von der Spannung bei 40 Sensormessungen pro Messzeitraum



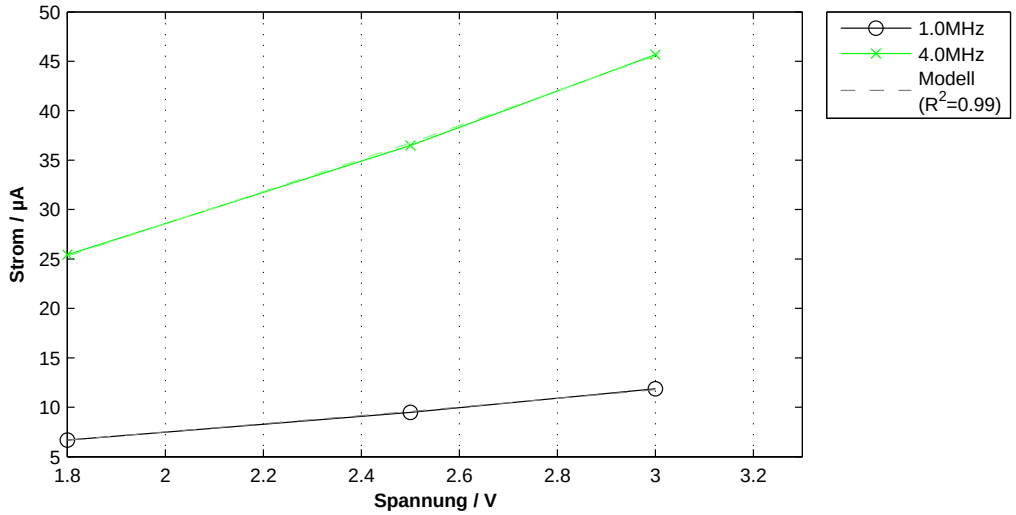
Modell: $I(U,f) = -2.237 \cdot 10^{-6} + 8.119 \cdot 10^{-6} \cdot U + 2.922 \cdot 10^{-12} \cdot U \cdot f + 2.298 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.10: MSP430F1232: Strom abhängig von der Frequenz bei 40 Sensormessungen pro Messzeitraum



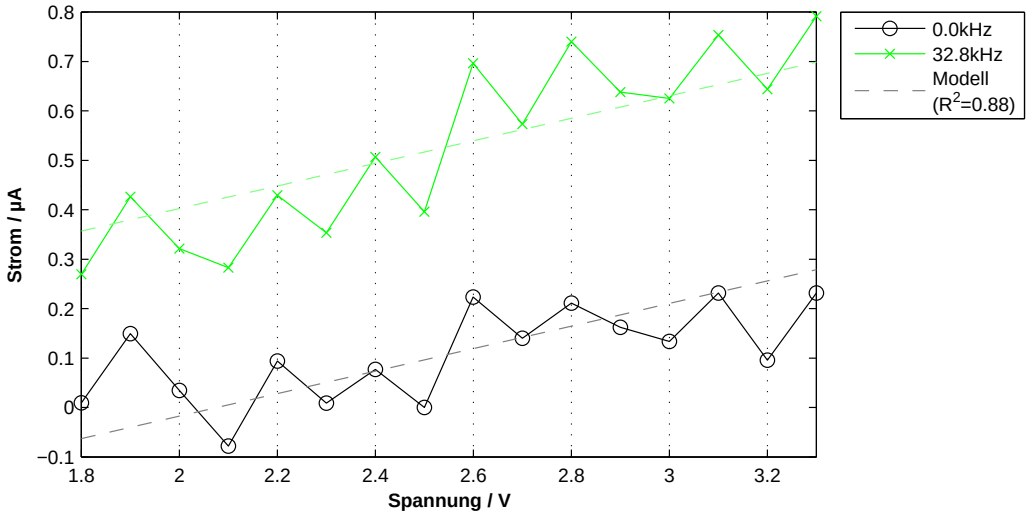
Modell: $I(U,f,n) = -2.237 \cdot 10^{-6} + 1.453 \cdot 10^{-6} \cdot U + 1.667 \cdot 10^{-7} \cdot U \cdot n + 2.922 \cdot 10^{-12} \cdot U \cdot f + 2.298 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.11: MSP430F1232: Strom abhängig von der Anzahl der Sensormessungen pro Messzeitraum



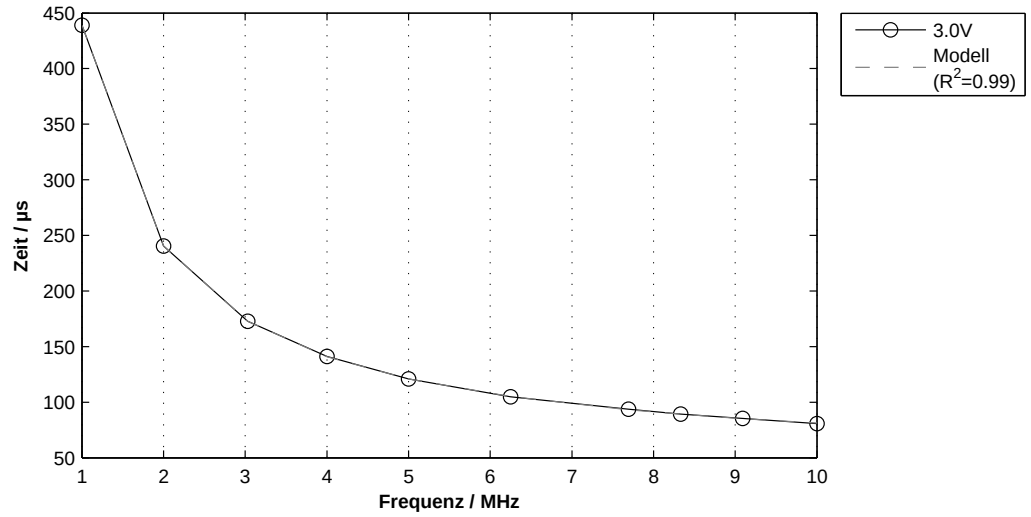
Modell: $I_{lp}(U, f) = +4.308 \cdot 10^{-7} + 3.612 \cdot 10^{-8} \cdot U + 2.976 \cdot 10^{-12} \cdot U \cdot f + 2.584 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.12: MSP430F1232: Strom abhängig von der Spannung im Low-Power-Mode bei aktivem Takt



Modell: $I_{st}(U, f_{RTC}) = -4.735 \cdot 10^{-7} + 2.279 \cdot 10^{-7} \cdot U + 1.283 \cdot 10^{-11} \cdot f_{RTC}$

Abbildung 5.13: MSP430F1232: statischer Strom abhängig von der Spannung



Modell: $t(f) = +4.150 \cdot 10^{-5} + 3.976 \cdot 10^{2} \cdot \frac{1}{f}$

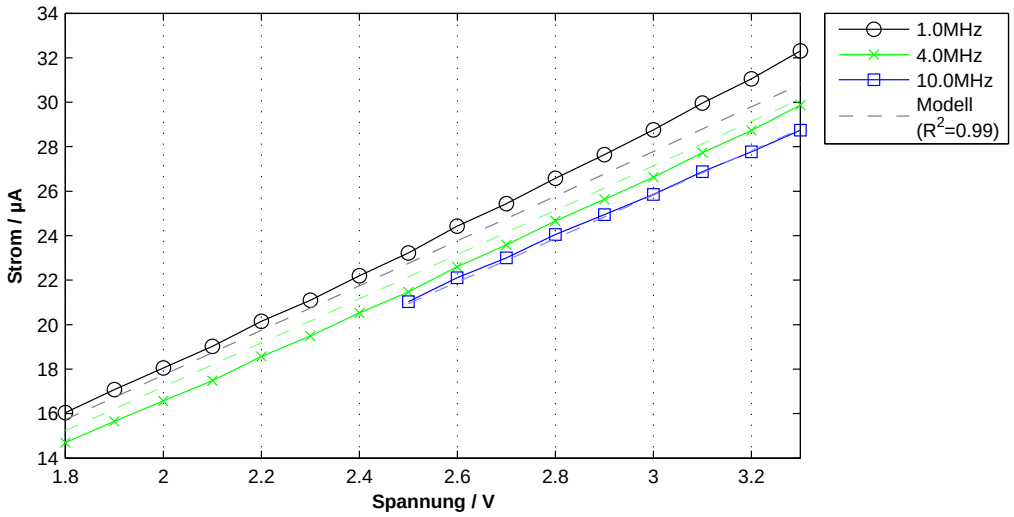
Abbildung 5.14: MSP430F1232: aktive Zeit abhängig von der Frequenz

5.1.3 MSP430F2232

Die Messergebnisse für den MSP430F2232 werden in den folgenden Abbildungen dargestellt. Abbildung 5.15 zeigt den Stromverbrauch in Abhängigkeit von der Spannung. Abbildung 5.16 zeigt den Strom abhängig von der Frequenz. Durch die Einschränkung, die sich durch den sicheren Betriebsbereich ergeben (ab 1,8 V max. 4,15 MHz, ab 2,2 V max. 7,5 MHz, ab 2,7 V max. 12 MHz, dazwischen linear interpoliert [Tex11, S. 26]) erstrecken sich nicht alle Graphen über die ganzen Diagramme.

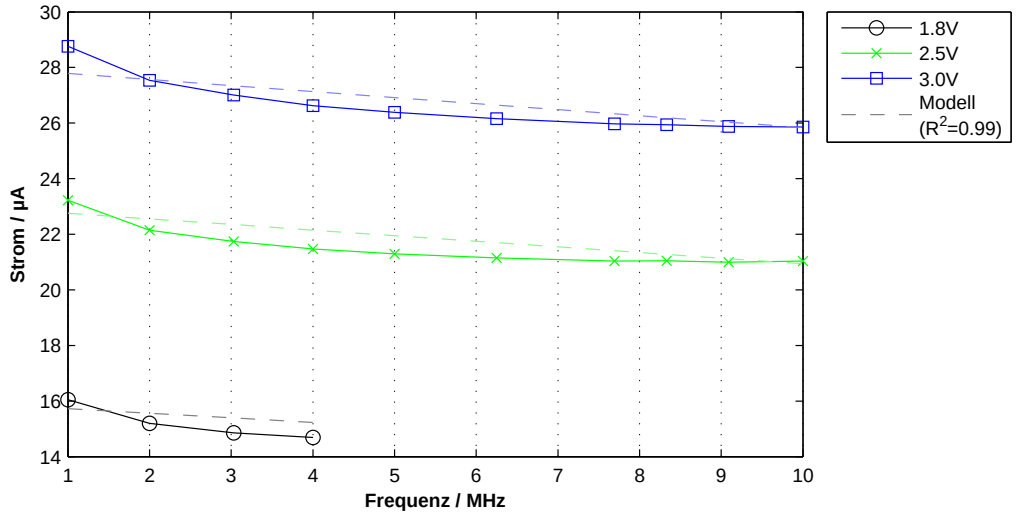
Die Abbildung 5.17 zeigt die Abhängigkeit des Stromes von der Anzahl der Sensormessungen pro Messzeitraum. Das Modell konvergiert in diesem Fall nicht im Nullpunkt. Das ist auf einen, gegenüber dem theoretischen Modell fehlenden Modellparameter zurückzuführen. Aufgrund zu geringer Signifikanz konnte der Koeffizient b_4 des theoretischen Modells nicht bestimmt werden und die vereinfachte Näherung $b_4{'}$ aus Gleichung (5.1) wurde verwendet.

In Abbildung 5.18 ist der Stromverbrauch im Low-Power-Mode abhängig von der Spannung dargestellt. Die Abbildung 5.19 zeigt den statischen Verbrauch. Im Vergleich bewegen sich die Messwerte beider Diagramme im selben Bereich. Der wesentliche Unterschied ist die Auflösung mit der beide Messungen durchgeführt wurden. Somit hat beim MSP430F2232 der Takt einen vernachlässigbaren Einfluss auf den Low-Power-Mode. Den Zusammenhang zwischen Taktfrequenz und aktiver Zeit des Mikrocontrollers zeigt Abbildung 5.20.



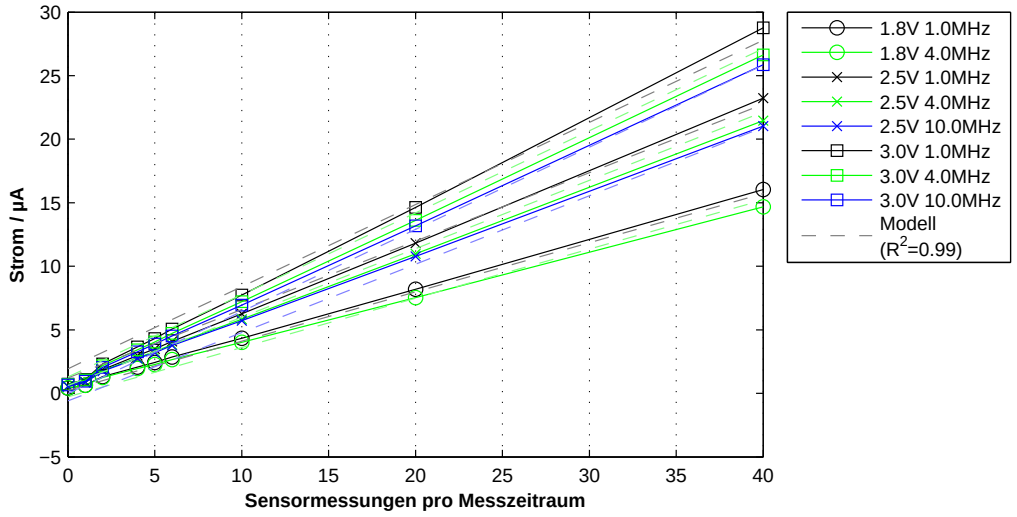
Modell: $I(U,f) = -2.258 \cdot 10^{-6} + 1.009 \cdot 10^{-5} \cdot U - 1.229 \cdot 10^{-13} \cdot U \cdot f + 1.687 \cdot 10^{-14} \cdot U^2 \cdot f$

Abbildung 5.15: MSP430F2232: Strom abhängig von der Spannung bei 40 Sensormessungen pro Messzeitraum



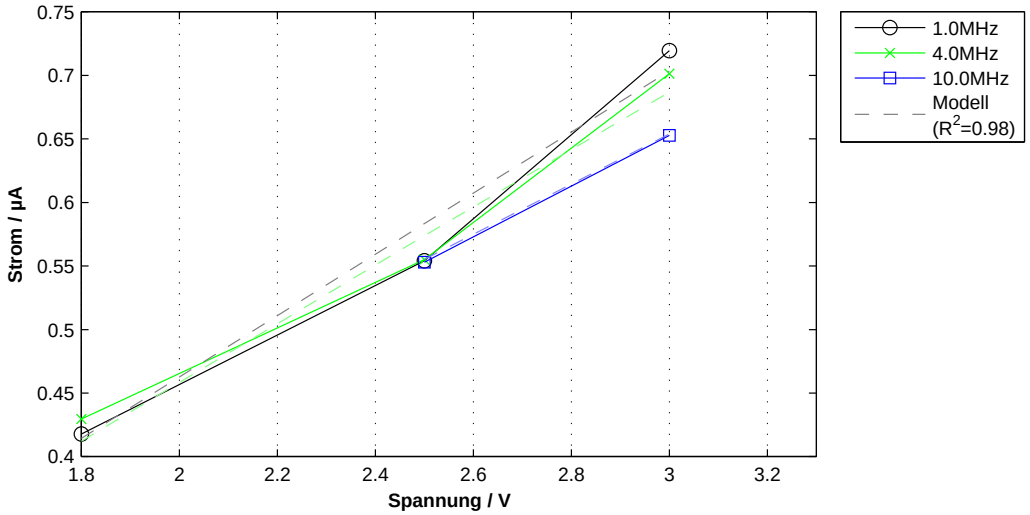
Modell: $I(U,f) = -2.258 \cdot 10^{-6} + 1.009 \cdot 10^{-5} \cdot U - 1.229 \cdot 10^{-13} \cdot U \cdot f + 1.687 \cdot 10^{-14} \cdot U^2 \cdot f$

Abbildung 5.16: MSP430F2232: Strom abhängig von der Frequenz bei 40 Sensormessungen pro Messzeitraum



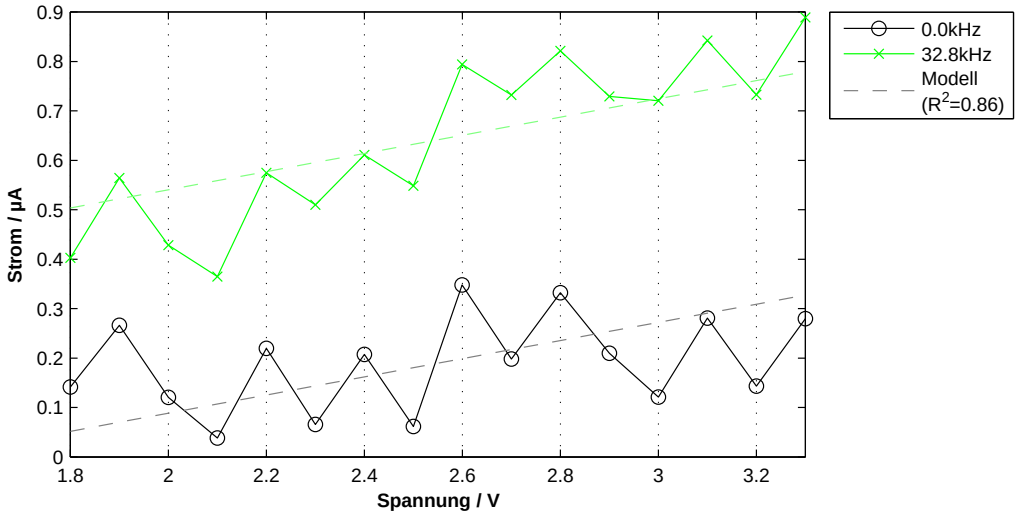
Modell: $I(U,f,n) = -2.258 \cdot 10^{-6} + 1.472 \cdot 10^{-6} \cdot U + 2.154 \cdot 10^{-7} \cdot U \cdot n - 1.229 \cdot 10^{-13} \cdot U \cdot f + 1.687 \cdot 10^{-14} \cdot U^2 \cdot f$

Abbildung 5.17: MSP430F2232: Strom abhängig von der Anzahl der Sensormessungen pro Messzeitraum



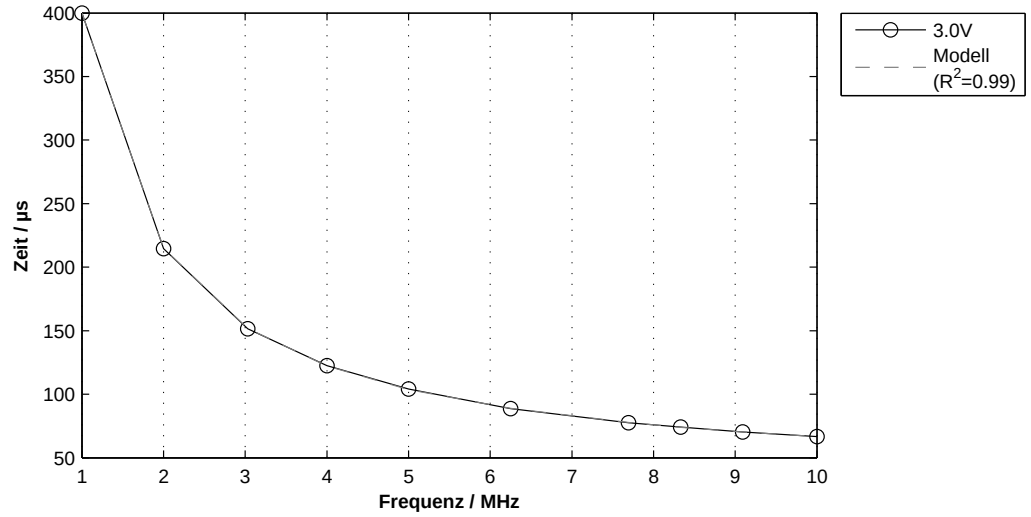
Modell: $I_{lp}(U, f) = -2.499 \cdot 10^{-8} + 2.446 \cdot 10^{-7} \cdot U + 1.617 \cdot 10^{-15} \cdot U \cdot f - 1.147 \cdot 10^{-15} \cdot U^2 \cdot f$

Abbildung 5.18: MSP430F2232: Strom abhängig von der Spannung im Low-Power-Mode bei aktivem Takt



Modell: $I_{st}(U, f_{RTC}) = -2.792 \cdot 10^{-7} + 1.838 \cdot 10^{-7} \cdot U + 1.380 \cdot 10^{-11} \cdot f_{RTC}$

Abbildung 5.19: MSP430F2232: statischer Strom abhängig von der Spannung



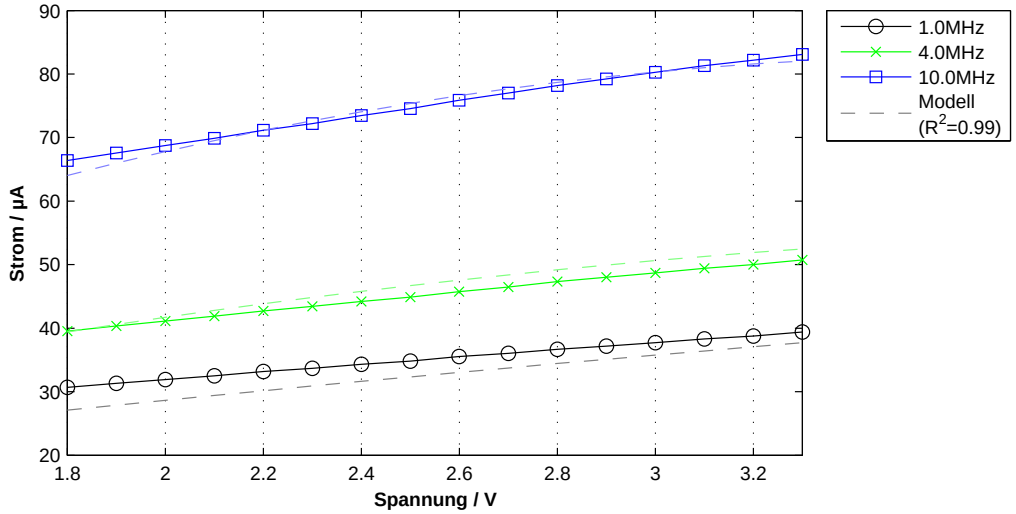
Modell: $t(f) = +2.960 \cdot 10^{-5} + 3.703 \cdot 10^{2} \cdot \frac{1}{f}$

Abbildung 5.20: MSP430F2232: aktive Zeit abhängig von der Frequenz

5.1.4 MSP430F5418A

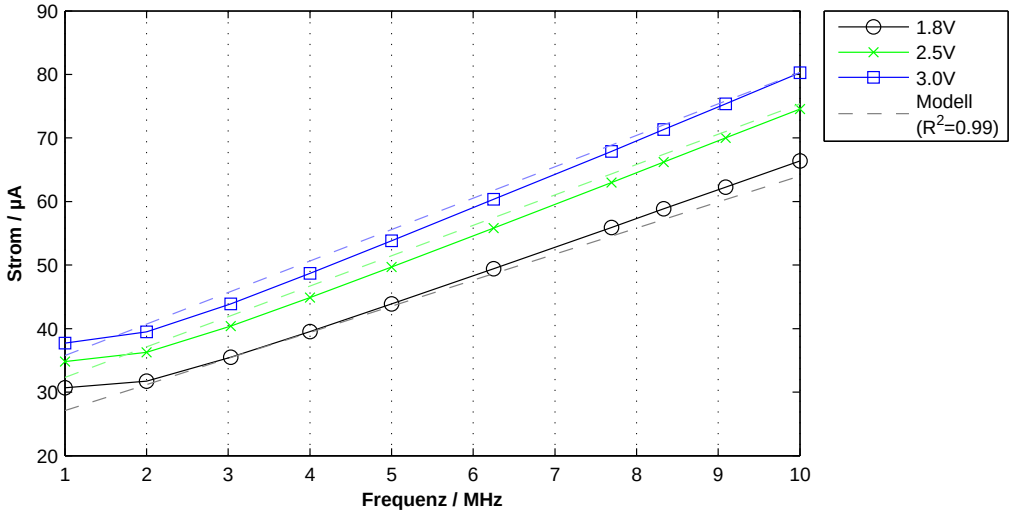
Die Messergebnisse für den MSP430F5418A werden in den folgenden Abbildungen dargestellt. Die Abbildungen 5.21 und 5.22 zeigen die Abhängigkeit des Stromverbrauchs von Spannung und Frequenz. Der sichere Betriebsbereich bei diesem Mikrocontroller (ab 1,8 V max. 8 MHz, ab 2,0 V max. 12 MHz, in Stufen [Tex10, S. 40]) umfasst nicht den ganzen Messbereich. Der untersuchte Chip ließ sich jedoch auch außerhalb der Spezifikation betreiben.

Abbildung 5.23 zeigt die Abhängigkeit des Stromes von der Anzahl der Sensormessungen pro Messzeitraum. Der hohe Stromverbrauch ohne Sensormessung ist, wie bereits in Abschnitt 5.1.2 diskutiert, auf das permanent aktive Taktnetzwerk zurückzuführen. Dies wird auch durch Abbildung 5.24 und 5.25 bestätigt, welche den Strom im Low-Power-Mode und den statischen Verbrauch zeigen. In Abbildung 5.26 wird die Abhängigkeit der aktiven Zeit von der Frequenz gezeigt.



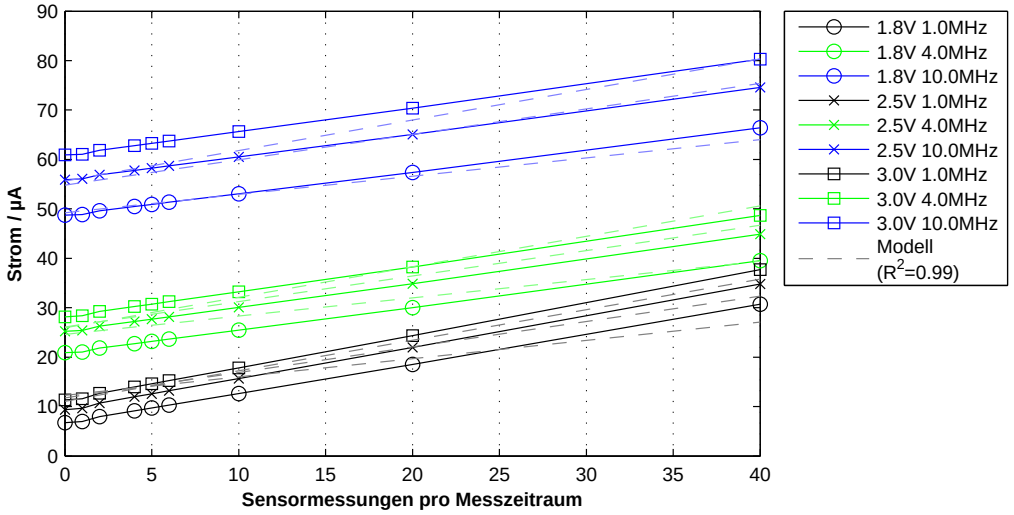
Modell: $I(U,f) = +1.126 \cdot 10^{-5} + 6.520 \cdot 10^{-6} \cdot U + 3.225 \cdot 10^{-12} \cdot U \cdot f - 5.247 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.21: MSP430F5418A: Strom abhängig von der Spannung bei 40 Sensormessungen pro Messzeitraum



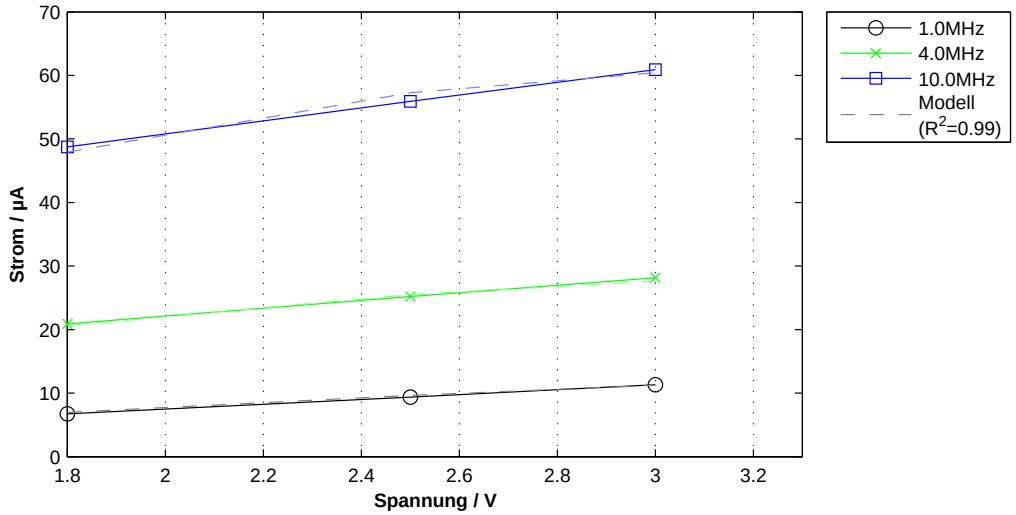
Modell: $I(U,f) = +1.126 \cdot 10^{-5} + 6.520 \cdot 10^{-6} \cdot U + 3.225 \cdot 10^{-12} \cdot U \cdot f - 5.247 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.22: MSP430F5418A: Strom abhängig von der Frequenz bei 40 Sensormessungen pro Messzeitraum



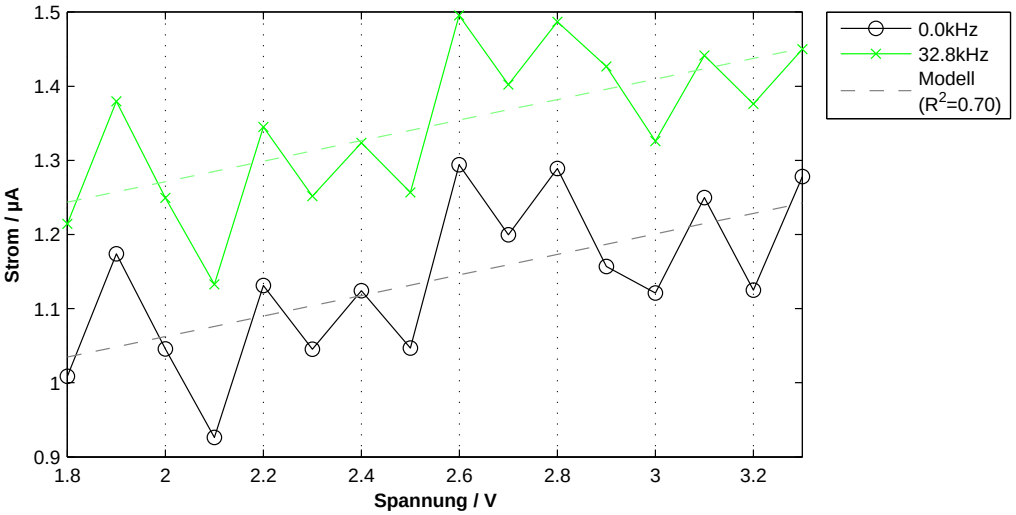
Modell: $I(U,f,n) = +1.126 \cdot 10^{-5} - 1.710 \cdot 10^{-6} \cdot U + 2.057 \cdot 10^{-7} \cdot U \cdot n + 3.225 \cdot 10^{-12} \cdot U \cdot f - 5.247 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.23: MSP430F5418A: Strom abhängig von der Anzahl der Sensormessungen pro Messzeitraum



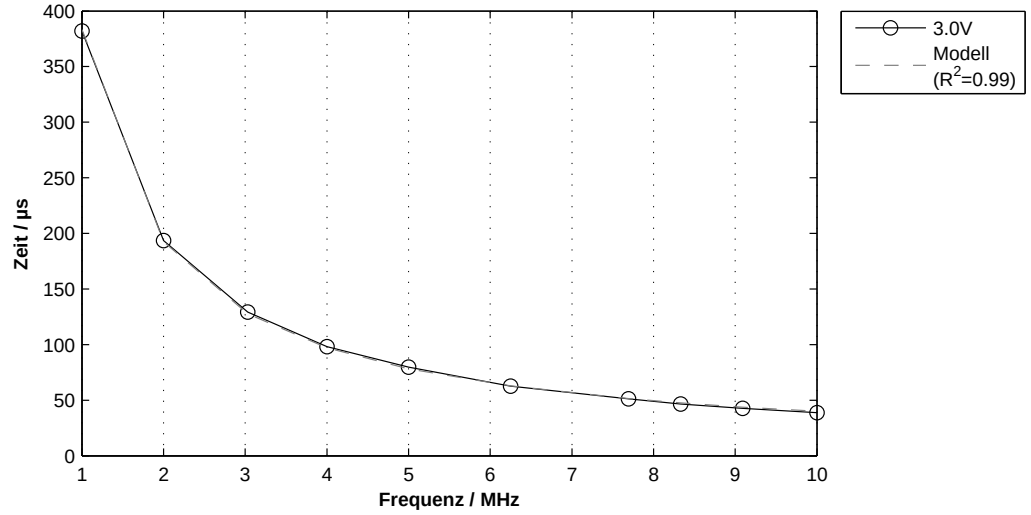
Modell: $I_{lp}(U, f) = -2.545 \cdot 10^{-6} + 2.765 \cdot 10^{-6} \cdot U + 3.588 \cdot 10^{-12} \cdot U \cdot f - 5.887 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.24: MSP430F5418A: Strom abhängig von der Spannung im Low-Power-Mode bei aktivem Takt



Modell: $I_{st}(U, f_{RTC}) = +7.853 \cdot 10^{-7} + 1.385 \cdot 10^{-7} \cdot U + 6.377 \cdot 10^{-12} \cdot f_{RTC}$

Abbildung 5.25: MSP430F5418A: statischer Strom abhängig von der Spannung



Modell: $t(f) = +1.857 \cdot 10^{-6} + 3.813 \cdot 10^{2} \cdot \frac{1}{f}$

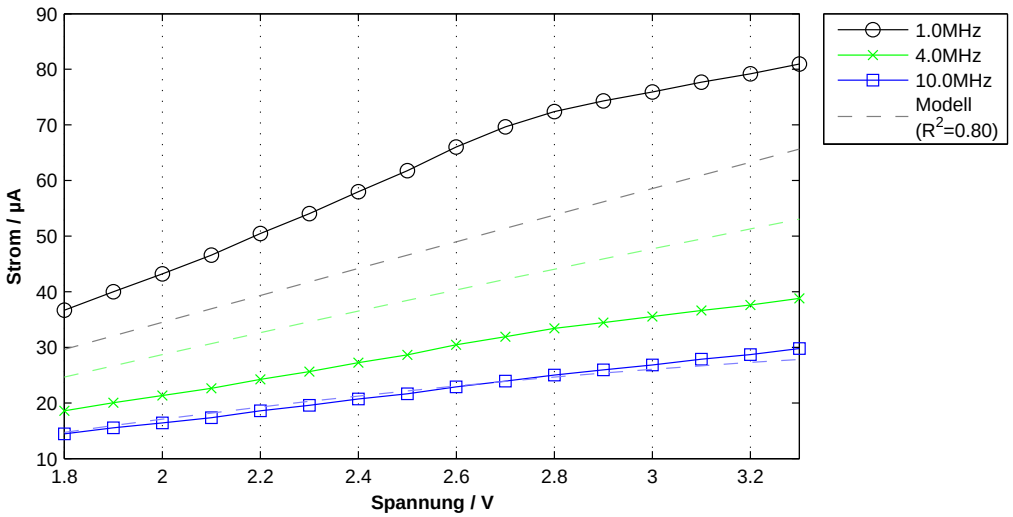
Abbildung 5.26: MSP430F5418A: aktive Zeit abhängig von der Frequenz

5.1.5 PIC16LF727

Die Messergebnisse für den PIC16LF727 werden in den folgenden Abbildungen dargestellt. Die Abbildungen 5.27 und 5.28 zeigen den Stromverbrauch in Abhängigkeit von Spannung bzw. Frequenz. Der sichere Betriebsbereich des Mikrocontrollers (ab 1,8 V max. 16 MHz [Mic09, S. 223]) deckt den gesamten untersuchten Messbereich ab. Der Unterschied zwischen gemessenen Werten und berechnetem Modell ist sowohl durch das Bestimmtheitsmaß als auch durch die grafische Abweichung deutlich zu erkennen. Aufgrund der inversen Proportionalität von Stromverbrauch und Frequenz wurde das zweite Berechnungsmodell eingeführt und der Modellparameter b_7 durch b_7' ersetzt. Eine Berechnung des Modells mit einem weiteren Parameter war mit den zugrundeliegenden Daten nicht möglich. Die Abbildungen 5.29 und 5.30 zeigen die gleichen Messwerte allerdings mit angepasstem Modell. Das Bestimmtheitsmaß konnte mit der Anpassung von 0,80 auf 0,89 erhöht werden. Im Weiteren werden von diesem Mikrocontroller nur Diagramme mit angepasstem Modell gezeigt.

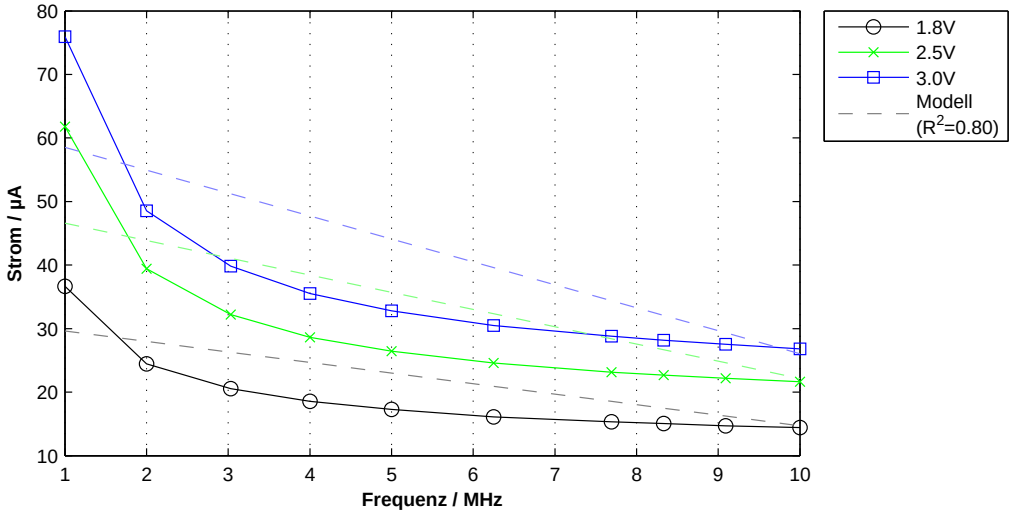
In Abbildung 5.31 ist der Strom abhängig von der Anzahl der Sensormessungen pro Messzeitraum dargestellt. Die Messwerte gehen mit geringerer Anzahl Messungen gegen Null. Die berechneten Werte können diese Tendenz aufgrund der bereits in Abschnitt 5.1.3 erläuterten fehlenden Frequenzkomponente des entsprechenden Parameters, nur ungenügend modellieren.

Abbildung 5.32 zeigt den Stromverbrauch im Low-Power-Mode abhängig von der Spannung. In Abbildung 5.33 ist der statische Strom in Abhängigkeit der Spannung dargestellt. Abschließend zeigt Abbildung 5.34 den Zusammenhang zwischen aktiver Zeit des Mikrocontrollers und der Frequenz.



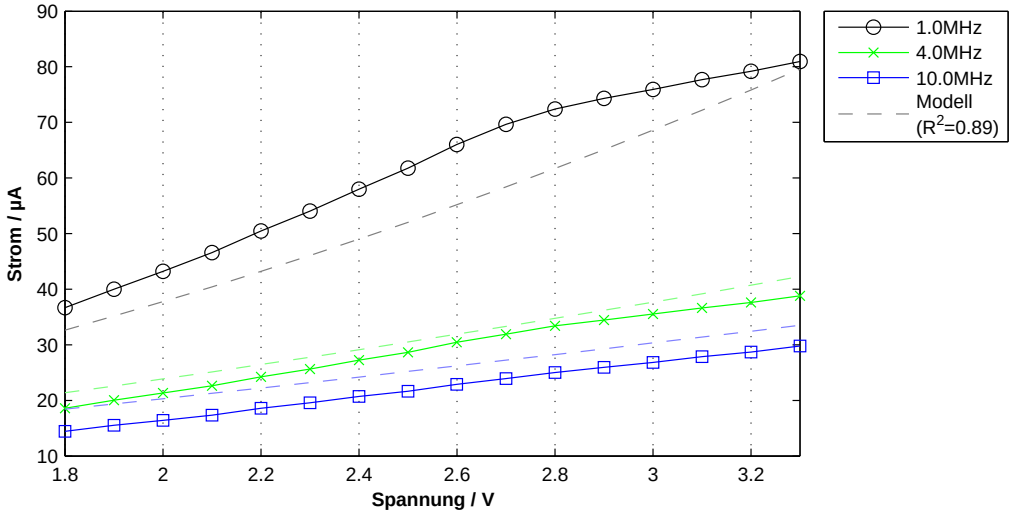
Modell: $I(U,f) = -1.502 \cdot 10^{-5} + 2.572 \cdot 10^{-5} \cdot U - 4.960 \cdot 10^{-13} \cdot U \cdot f - 2.357 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.27: PIC16LF727: Strom abhängig von der Spannung bei 40 Sensormessungen pro Messzeitraum (Modell 1)



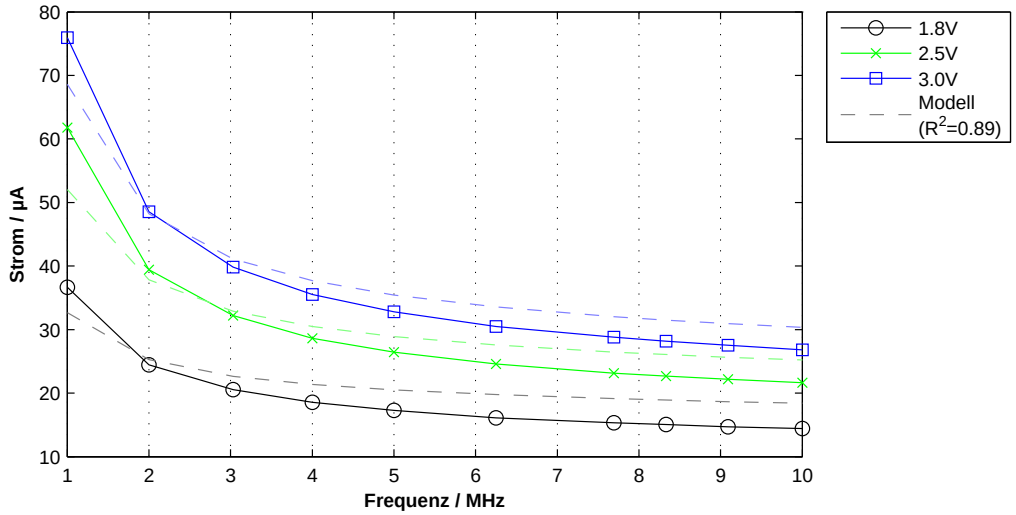
Modell: $I(U,f) = -1.502 \cdot 10^{-5} + 2.572 \cdot 10^{-5} \cdot U - 4.960 \cdot 10^{-13} \cdot U \cdot f - 2.357 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.28: PIC16LF727: Strom abhängig von der Frequenz bei 40 Sensormessungen pro Messzeitraum (Modell 1)



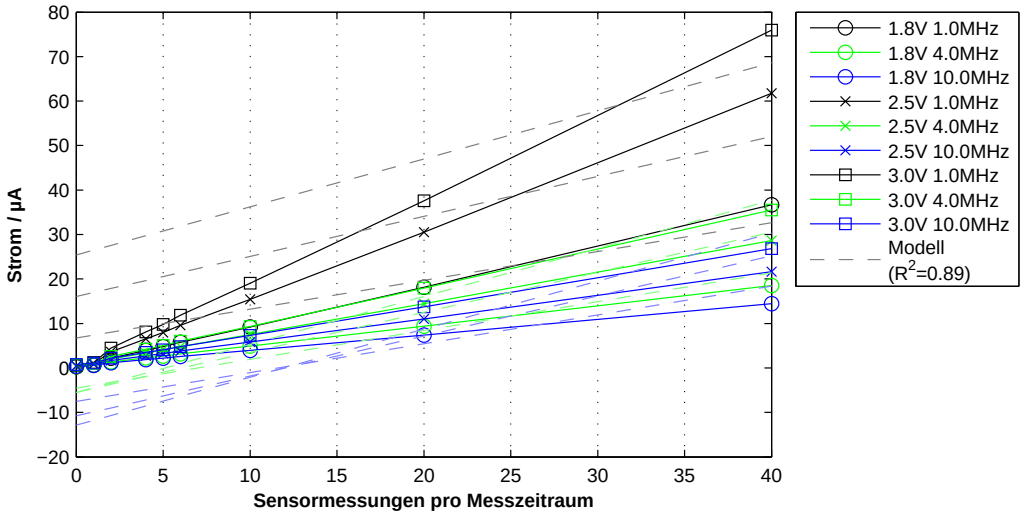
Modell: $I(U,f) = +2.933 \cdot 10^{-6} + 8.508 \cdot 10^{-6} \cdot U - 7.223 \cdot 10^{-14} \cdot U \cdot f + 4.484 \cdot 10^{0} \cdot U^2 \cdot \frac{1}{f}$

Abbildung 5.29: PIC16LF727: Strom abhängig von der Spannung bei 40 Sensormessungen pro Messzeitraum (Modell 2)



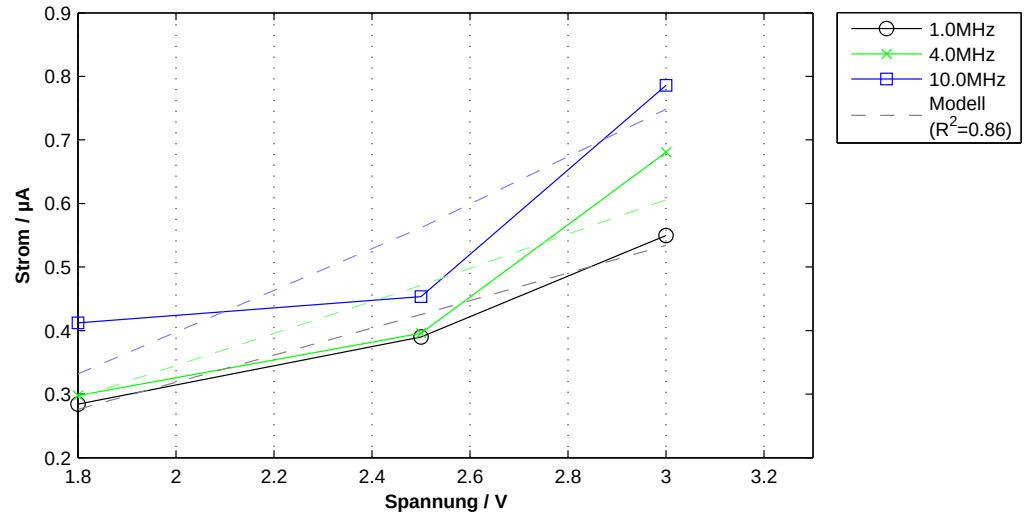
Modell: $I(U,f) = +2.933 \cdot 10^{-6} + 8.508 \cdot 10^{-6} \cdot U - 7.223 \cdot 10^{-14} \cdot U \cdot f + 4.484 \cdot 10^{0} \cdot U^2 \cdot \frac{1}{f}$

Abbildung 5.30: PIC16LF727: Strom abhängig von der Frequenz bei 40 Sensormessungen pro Messzeitraum (Modell 2)



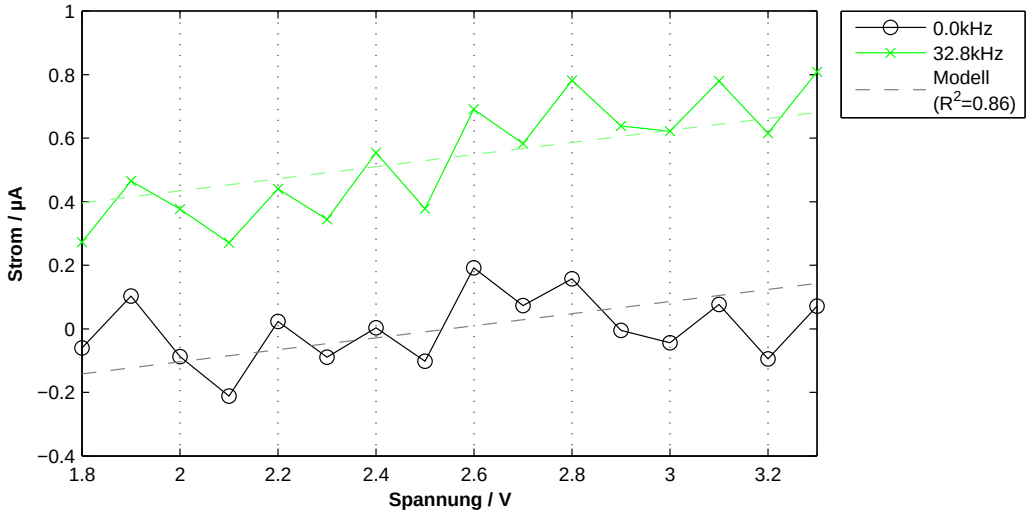
Modell: $I(U,f,n) = +2.933 \cdot 10^{-6} - 5.886 \cdot 10^{-6} \cdot U + 3.599 \cdot 10^{-7} \cdot U \cdot n - 7.223 \cdot 10^{-14} \cdot U \cdot f + 4.484 \cdot 10^{0} \cdot U^2 \cdot \frac{1}{f}$

Abbildung 5.31: PIC16LF727: Strom abhängig von der Anzahl der Sensormessungen pro Messzeitraum



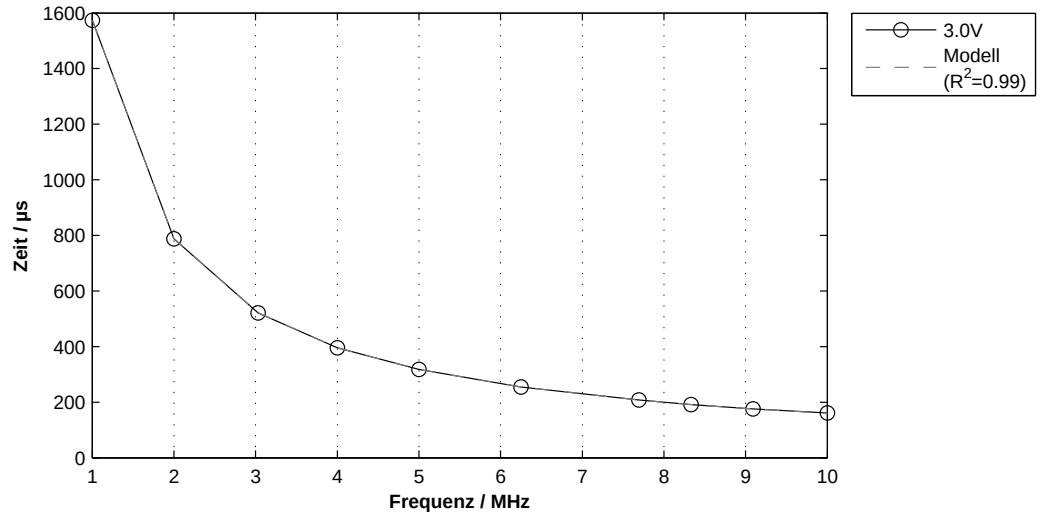
Modell: $I_{lp}(U, f) = -9.121 \cdot 10^{-8} + 2.006 \cdot 10^{-7} \cdot U - 3.260 \cdot 10^{-15} \cdot U \cdot f + 3.728 \cdot 10^{-15} \cdot U^2 \cdot f$

Abbildung 5.32: PIC16LF727: Strom abhängig von der Spannung im Low-Power-Mode bei aktivem Takt



Modell: $I_{st}(U, f_{RTC}) = -4.837 \cdot 10^{-7} + 1.899 \cdot 10^{-7} \cdot U + 1.644 \cdot 10^{-11} \cdot f_{RTC}$

Abbildung 5.33: PIC16LF727: statischer Strom abhängig von der Spannung



Modell: $t(f) = +3.394 \cdot 10^{-6} + 1.570 \cdot 10^{3} \cdot \frac{1}{f}$

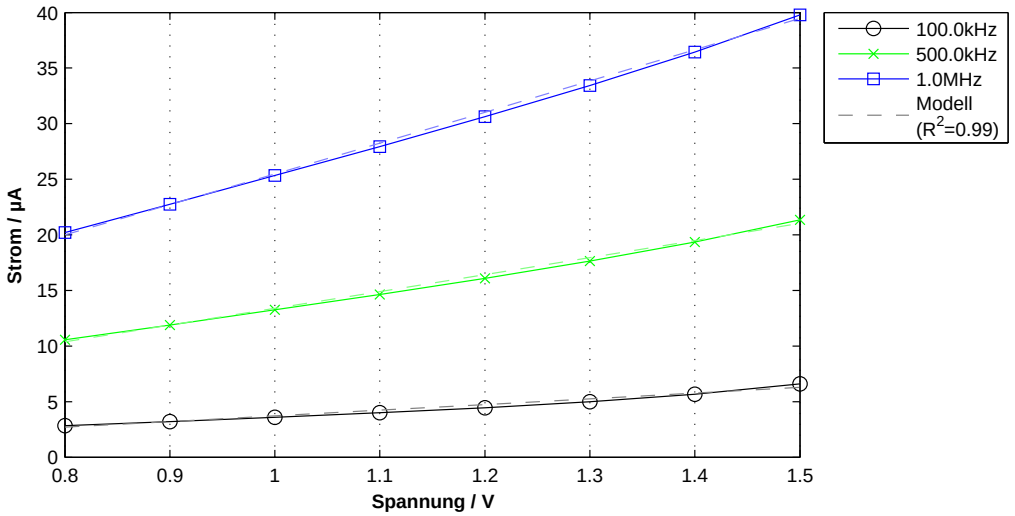
Abbildung 5.34: PIC16LF727: aktive Zeit abhängig von der Frequenz

5.2 Testchip

Es werden nun die Messergebnisse der Messung des Testchips vorgestellt. Die Grafiken stellen einen der untersuchten Chips exemplarisch dar. Der Messzeitraum beträgt, wie bei den Mikrocontrollern, 200 ms. Der Betriebsbereich des Testchip ist mit 0,8 V bis 1,5 V angegeben. Die zulässige Betriebsfrequenz ist laut Datenblatt 0 bis 25 MHz [GBW+11].

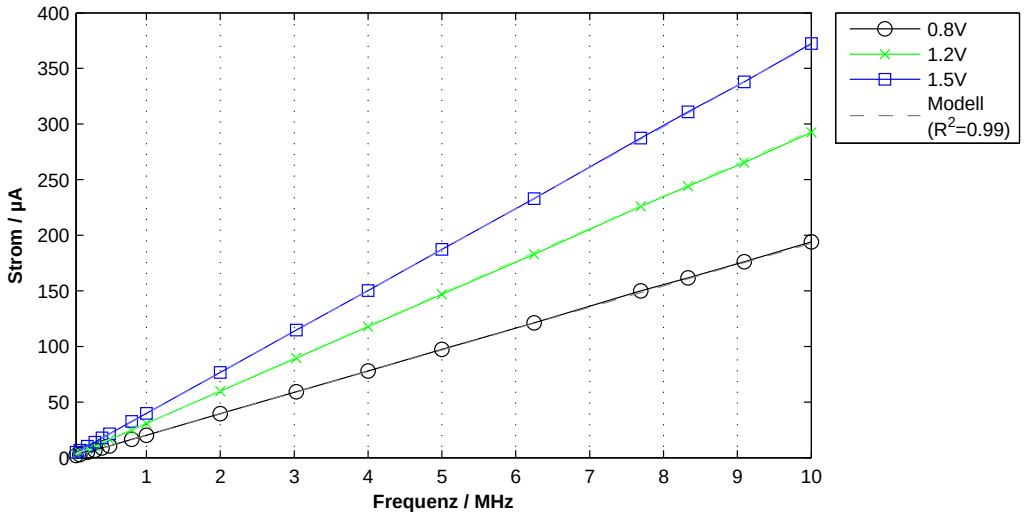
In Abbildung 5.35 ist der Stromverbrauch in Abhängigkeit der Core-Spannung angegeben. Abbildung 5.36 zeigt den Strom abhängig von der Betriebsfrequenz. Optisch passt sich das Modell gut an die gemessenen Daten an, was sich auch im Bestimmtheitsmaß ausdrückt. Die Abbildung 5.37 zeigt den Verbrauch über die Anzahl der Sensormessungen im Messzeitraum. Im Gegensatz zu den Mikrocontrollern zeigt diese Grafik nur nahezu perfekt horizontale Geraden. Der Modellparameter $b_4' = -0,4889\text{nA/V}$ aus Tabelle 5.1 bestätigt diesen Eindruck. Dieses Verhalten ist darauf zurückzuführen, dass der Testchip permanent läuft und die Anzahl der Sensormessungen nur Einfluss auf die intern geschalteten Register nimmt, während bei den Mikrocontrollern direkt die aktive Zeit beeinflusst wird.

In Abbildung 5.38 ist der Stromverbrauch in Abhängigkeit der Anzahl der Schwellwertüberschreitungen pro Messzeitraum dargestellt. Abbildung 5.39 zeigt den Stromverbrauch abhängig vom SPI-Busteiler. Im Gegensatz zu den Mikrocontrollern hat die Wahl des Busteilers beim Testchip einen vernachlässigbaren Einfluss auf den Gesamtverbrauch. Die Darstellung der aktiven Zeit entfällt aufgrund der Struktur des Testchips, siehe Abschnitt 4.7.



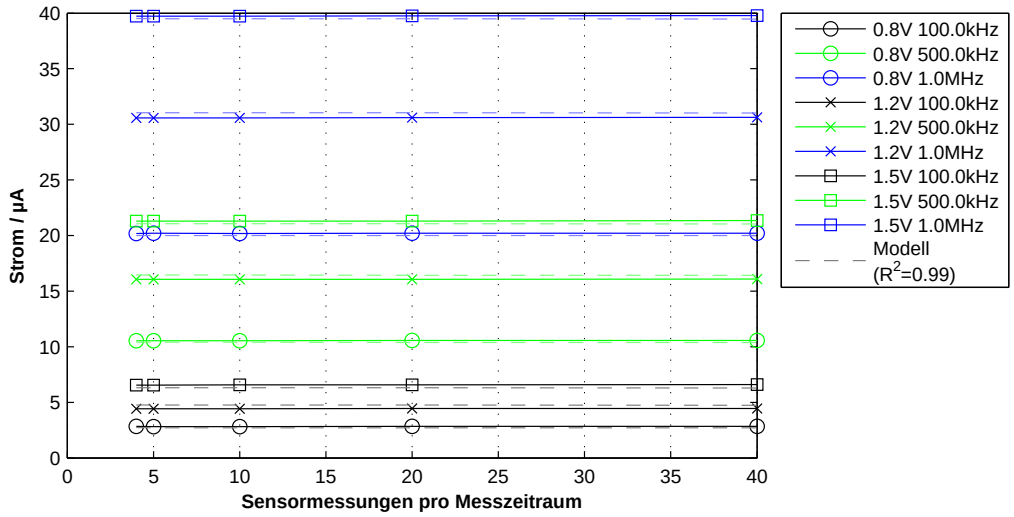
Modell: $I(U,f) = -1.281 \cdot 10^{-6} + 2.586 \cdot 10^{-6} \cdot U + 2.333 \cdot 10^{-11} \cdot U \cdot f + 8.378 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.35: Testchip: Strom abhängig von der Spannung bei 40 Sensormessungen pro Messzeitraum



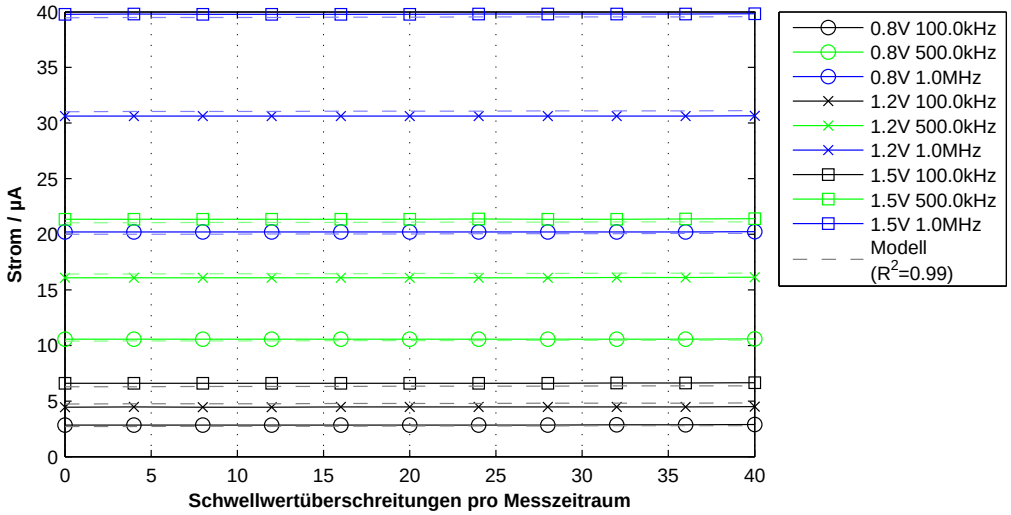
Modell: $I(U,f) = -1.281 \cdot 10^{-6} + 2.586 \cdot 10^{-6} \cdot U + 2.333 \cdot 10^{-11} \cdot U \cdot f + 8.378 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.36: Testchip: Strom abhängig von der Frequenz bei 40 Sensormessungen pro Messzeitraum



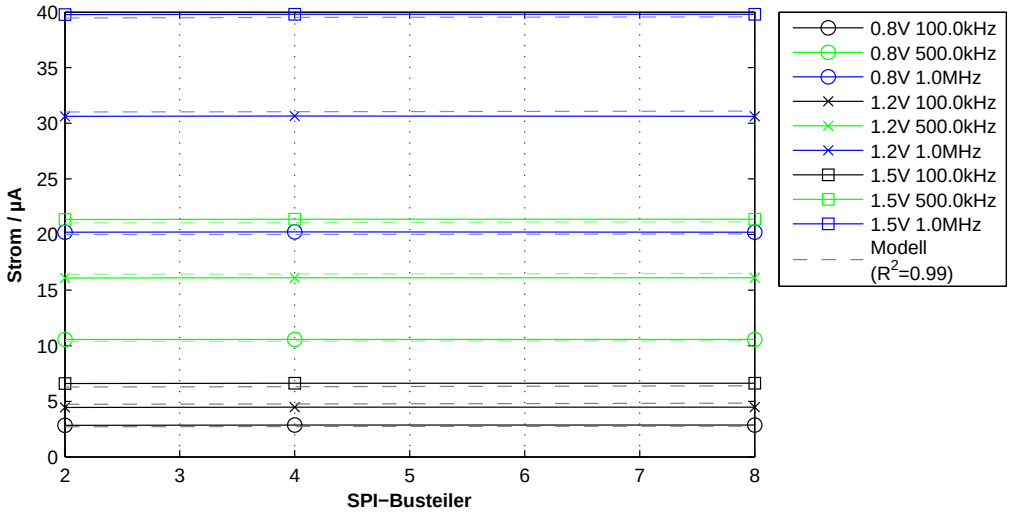
Modell: $I(U,f,n) = -1.281 \cdot 10^{-6} + 2.605 \cdot 10^{-6} \cdot U - 4.889 \cdot 10^{-10} \cdot U \cdot n + 2.333 \cdot 10^{-11} \cdot U \cdot f + 8.378 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.37: Testchip: Strom abhängig von der Anzahl der Sensormessungen pro Messzeitraum



Modell: $I(U, f, m) = -1.281 \cdot 10^{-6} + 2.586 \cdot 10^{-6} \cdot U + 2.333 \cdot 10^{-11} \cdot U \cdot f + 8.378 \cdot 10^{-13} \cdot U^2 \cdot f + 2.166 \cdot 10^{-9} \cdot m$

Abbildung 5.38: Testchip: Strom abhängig von der Anzahl der Schwellwertüberschreitungen pro Messzeitraum



Modell: $I(U, f, d) = -1.281 \cdot 10^{-6} + 2.563 \cdot 10^{-6} \cdot U + 2.333 \cdot 10^{-11} \cdot U \cdot f + 1.136 \cdot 10^{-8} \cdot U \cdot d + 8.378 \cdot 10^{-13} \cdot U^2 \cdot f$

Abbildung 5.39: Testchip: Strom abhängig vom SPI-Busteiler bei 40 Sensormessungen pro Messzeitraum

6 Evaluierung

In diesem Kapitel wird in Abschnitt 6.1 gezeigt, wie aus den ermittelten Messwerten der Energieverbrauch der untersuchten Chips berechnet wurde. Dabei wird auf die Mikrocontroller und den Testchip jeweils gesondert eingegangen. Abschnitt 6.2 stellt die aktiven Zeiten der MCUs bei der Sensormessung gegenüber. Den Abschluss bildet die Gegenüberstellung des Energieverbrauchs der einzelnen Typen und die Diskussion der Ergebnisse in Abschnitt 6.3.

6.1 Energieverbrauch pro Sensormessung

Der Energieverbrauch pro Sensormessung wurde wie in Abschnitt 3.6 ausgeführt, in zwei Schritten berechnet. Im ersten Schritt wurde der Stromverbrauch pro Sensormessung errechnet und im zweiten Schritt daraus der Energieverbrauch pro Sensormessung. Zur Berechnung des Stromverbrauches wurden drei Methoden entwickelt, auf welche an dieser Stelle nochmals eingegangen werden soll. Außerdem werden die Einzelergebnisse der MCUs und des Testchip grafisch dargestellt.

Die erste Berechnungsmöglichkeit basierte auf Modellbildung und anschließender Regressionsberechnung. In der praktischen Messung wurden zwei Probleme bei der Auswertung nach dieser Methode festgestellt. Das erste Problem betraf das Modell selbst. Wie in Kapitel 5 bereits erwähnt wurde, musste das Modell angepasst werden, um die Regressionsberechnung mit den vorliegenden Messdaten durchführen zu können. Dabei wurde der Koeffizient b_4 zu b_4', welcher keine Frequenzabhängigkeit mehr aufweist. Dadurch wird eine wesentliche Einflussgröße nicht mehr berücksichtigt. Das zweite Problem betrifft einzelne MCUs und deren ineffektives internes Taktnetz. Beim MSP430F1232 und beim MSP430F5418A waren trotz permanentem LPM Stromwerte gemessen worden, die im Bereich des halben aktiven Stromes lagen. Somit wären diese MCUs bevorzugt, weil der hohe Stromverbrauch im LPM durch das lineare Modell zu einer geringeren Steigung der Geraden des Stromverbrauches abhängig von der Anzahl der Sensormessungen geführt hätte. Die Steigung dieser Geraden entspricht wiederum genau dem Stromverbrauch pro Sensormessung. Aufgrund dieser Probleme wurde die Berechnung nach dieser Methode nicht verwendet.

Die zweite Berechnungsmethode basierte auf Messung von Aktiv- und Ruhestrom. Aufgrund des bereits erwähnten, erhöhten Stromverbrauches im LPM zweier MCUs, musste die Gleichung (3.22) aus Abschnitt 3.6 angepasst werden. Um die Berechnung durchführen zu können, musste zusätzlich der statische Stromverbrauch im LPM bestimmt werden. Dies wurde durch Messung bei deaktiviertem Takt erreicht. Anschließend war es möglich den tatsächlichen Stromverbrauch pro

Sensormessung zu berechnen. Gleichung (6.1) zeigt zwei Bruchterme. Der erste Term beschreibt dabei den Stromverbrauch pro Sensormessung unter Vernachlässigung des Taktnetzes. Der zweite Term gibt den Anteil des Taktnetzes am Stromverbrauch bezogen auf die aktive Zeit t_a an. Dieser muss nicht nochmals auf eine Sensormessung bezogen werden, da die aktive Zeit bereits diesen Faktor enthält. Der Stromverbrauch im aktiven Betriebszustand wurde dabei bei $n = 40$ Sensormessungen bestimmt. Der Ruhestrom wurde bei $n = 0$ Sensormessungen und der statische Verbrauch bei $n = 0$ mit deaktiviertem Takt gemessen. Der Integrationszeitraum wurde, wie bei allen Messungen, mit 200 ms festgelegt. Aufgrund der Funktionsweise des Testchips war eine Bestimmung des Ruhestromes nicht möglich und diese Berechnungsmethode konnte nur für die MCUs jedoch nicht für den Testchip verwendet werden.

$$
\begin{aligned}
I_\mathrm{SM} &= \frac{I_\mathrm{T} - I_\mathrm{T,LP}}{n} - \frac{(I_\mathrm{T,LP} - I_\mathrm{T,ST}) \cdot t_\mathrm{a}}{T} \\
\text{mit} \quad I_\mathrm{SM} &= \text{Stromverbrauch pro Sensormessung} \\
I_\mathrm{T} &= \text{gemessener Strom im aktiven Betrieb} \\
I_\mathrm{T,LP} &= \text{gemessener Strom im LPM} \\
I_\mathrm{T,ST} &= \text{gemessener statischer Strom im LPM, Takt deaktiviert} \\
n &= \text{Anzahl der Sensormessungen} \\
t_\mathrm{a} &= \text{aktive Zeit des Mikrocontrollers} \\
T &= \text{Messzeitraum, Integrationszeit}
\end{aligned}
\tag{6.1}
$$

Die dritte in Abschnitt 3.6 beschriebene Berechnungsmethode basierte auf einem einfacheren Modell als die erste Methode aber ebenfalls auf linearer Regression. Aufgrund der bereits mehrfach beschriebenen Probleme der MCUs im LPM wurde diese Methode nur beim Testchip eingesetzt.

Der Stromverbrauch pro Sensormessung wurde für die MCUs mit der zweiten Berechnungsmethode bestimmt, für den Testchip mit der dritten Berechnungsmethode. Anschließend wurde der Energieverbrauch pro Sensormessungen für alle Mikrochips nach Gleichung (3.8) berechnet. Im Weiteren wird auf die Ergebnisse der Berechnung des Energieverbrauches pro Sensormessung für jeden untersuchten Chip eingegangen.

6.1.1 ATmega88PA

Abbildung 6.1 zeigt den Energieverbrauch pro Sensormessung abhängig von der Betriebsspannung. Bei gleicher Spannung ist der Energieverbrauch bei höherer Taktfrequenz geringer. Abbildung 6.2 stellt den Energieverbrauch pro Sensormessung abhängig von der Taktfrequenz dar. Der geringste Energieverbrauch lässt sich aus den Grafiken ableiten und ist bei 1,8 V und 4 MHz und beträgt 110,7 nJ.

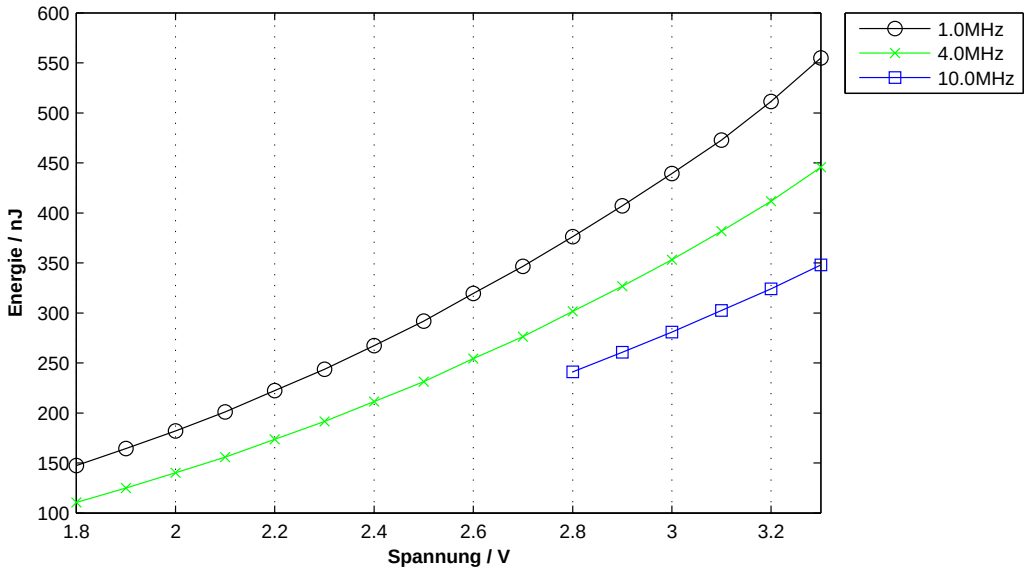


Abbildung 6.1: ATmega88PA: Energieaufnahme pro Sensormessung abhängig von der Spannung

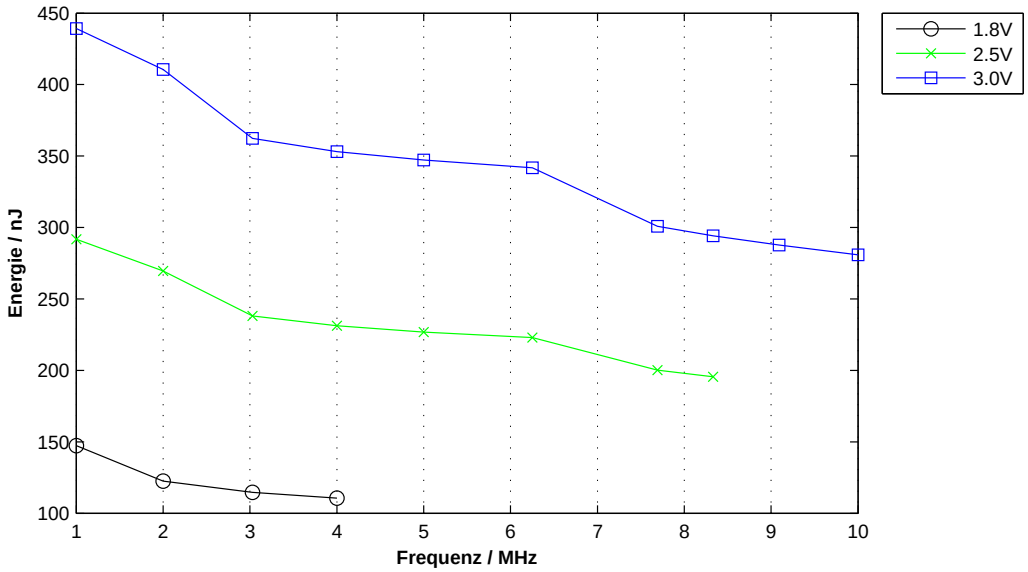


Abbildung 6.2: ATmega88PA: Energieaufnahme pro Sensormessung abhängig von der Frequenz

6.1.2 MSP430F1232

Abbildung 6.3 zeigt den Energieverbrauch pro Sensormessung abhängig von der Betriebsspannung. Aufgrund des Betriebsbereiches des MSP430F1232 fehlt die Kurve für 10 MHz. Auffällig ist der geringe Einfluss der Taktfrequenz auf den Energieverbrauch im Vergleich zum Einfluss der Betriebsspannung. Dies ist darauf zurückzuführen, dass sich der Anstieg des Stromverbrauches pro MHz mit der Reduktion der aktiven Zeit kompensiert. Abbildung 6.4 stellt den Energieverbrauch pro Sensormessung abhängig von der Taktfrequenz dar. Der geringste Energieverbrauch lässt sich aus den Grafiken ableiten und ist bei 1,8 V und 4 MHz und beträgt 105,7 nJ.

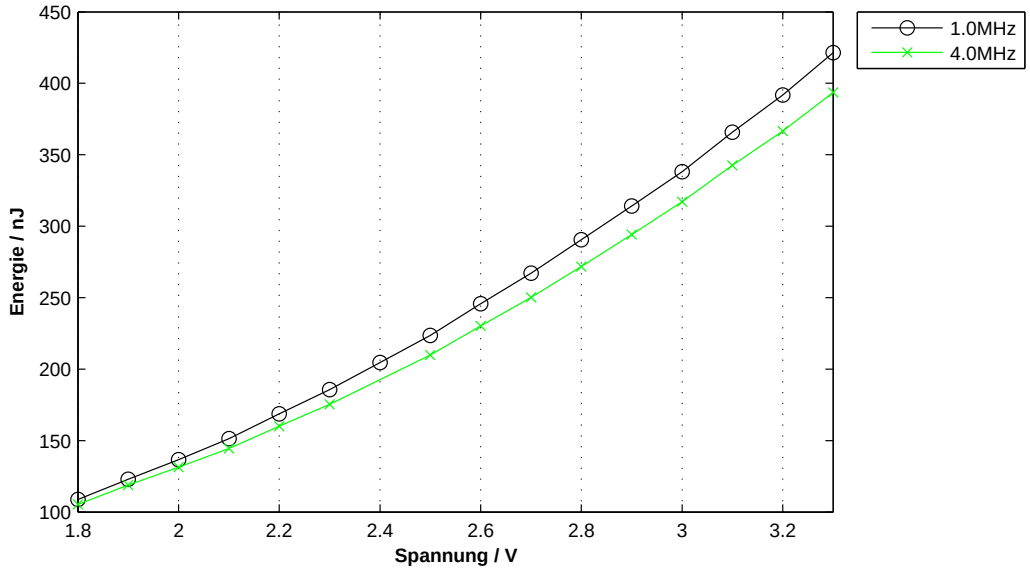


Abbildung 6.3: MSP430F1232: Energieaufnahme pro Sensormessung abhängig von der Spannung

6.1.3 MSP430F2232

Abbildung 6.5 zeigt den Energieverbrauch pro Sensormessung abhängig von der Betriebsspannung. Aufgrund des Betriebsbereiches des MSP430F2232 ist die Kurve für 10 MHz nicht vollständig. Wie beim MSP430F1232 deuten die Kurven auf eine stärkere Abhängigkeit von der Versorgungsspannung als von der Taktfrequenz. Abbildung 6.6 stellt den Energieverbrauch pro Sensormessung abhängig von der Taktfrequenz dar. Der geringste Energieverbrauch lässt sich aus den Grafiken ableiten und ist bei 1,8 V und 4 MHz und beträgt 128,5 nJ.

6.1.4 MSP430F5418A

Abbildung 6.7 zeigt den Energieverbrauch pro Sensormessung abhängig von der Betriebsspannung. Abbildung 6.8 stellt den Energieverbrauch pro Sensormessung abhängig von der Taktfrequenz dar. An zweiter Abbildung fällt auf, dass die Erhöhung der Taktfrequenz im Bereich um 1 MHz bis 4 MHz eine wesentlich größere Verringerung des Energieverbrauches liefert als bei höheren Taktfrequenzen. Der geringste Energieverbrauch lässt sich aus den Grafiken ableiten und ist bei 1,8 V und 10 MHz und beträgt 169,4 nJ.

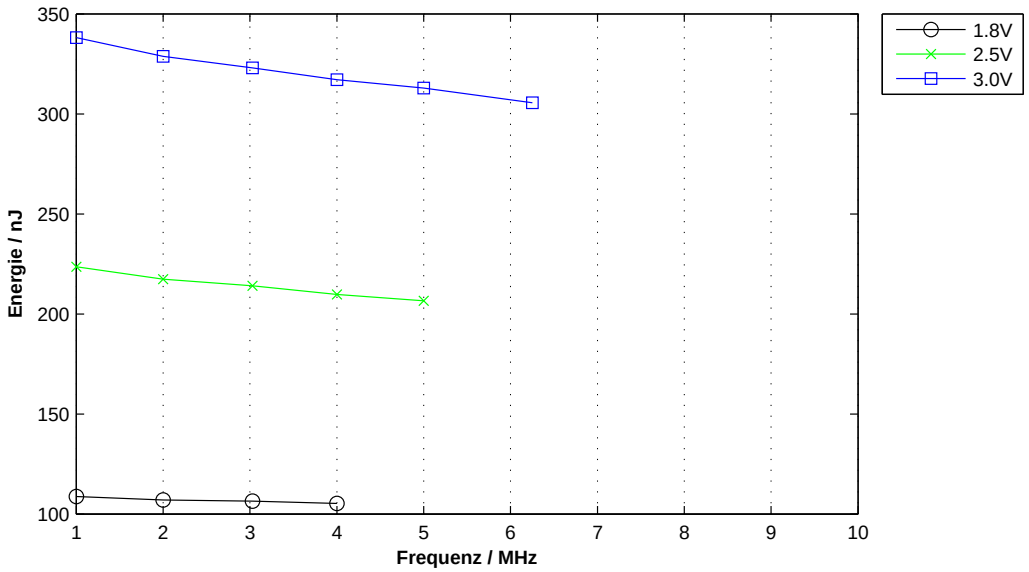


Abbildung 6.4: MSP430F1232: Energieaufnahme pro Sensormessung abhängig von der Frequenz

6.1.5 PIC16LF727

Abbildung 6.9 zeigt den Energieverbrauch pro Sensormessung abhängig von der Betriebsspannung. Abbildung 6.10 stellt den Energieverbrauch pro Sensormessung abhängig von der Taktfrequenz dar. Beim PIC16LF727 ist die Frequenzabhängigkeit noch ausgeprägter als beim zuvor betrachteten MSP430F5418A. Wiederum nähern sich die Kurven bei höherer Taktfrequenz einer gedachten parallelen zur Achse. Der geringste Energieverbrauch lässt sich aus den Grafiken ableiten und ist bei 1,8 V und 10 MHz und beträgt 126,8 nJ.

6.1.6 Testchip

Abbildung 6.11 zeigt den Energieverbrauch pro Sensormessung abhängig von der Betriebsspannung. Im Gegensatz zu den betrachteten MCUs steigert eine höhere Taktfrequenz den Energieverbrauch beim Testchip. Abbildung 6.12 stellt den Energieverbrauch pro Sensormessung abhängig von der Taktfrequenz dar. Der geringste Energieverbrauch lässt sich aus den Grafiken ableiten und ist bei 0,8 V und 100 kHz und beträgt 69,91 pJ.

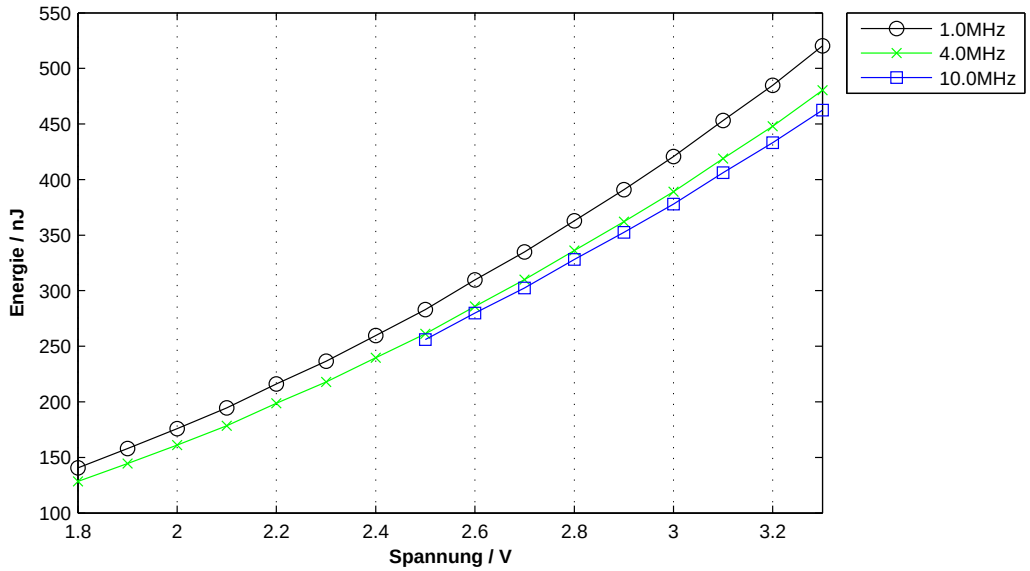


Abbildung 6.5: MSP430F2232: Energieaufnahme pro Sensormessung abhängig von der Spannung

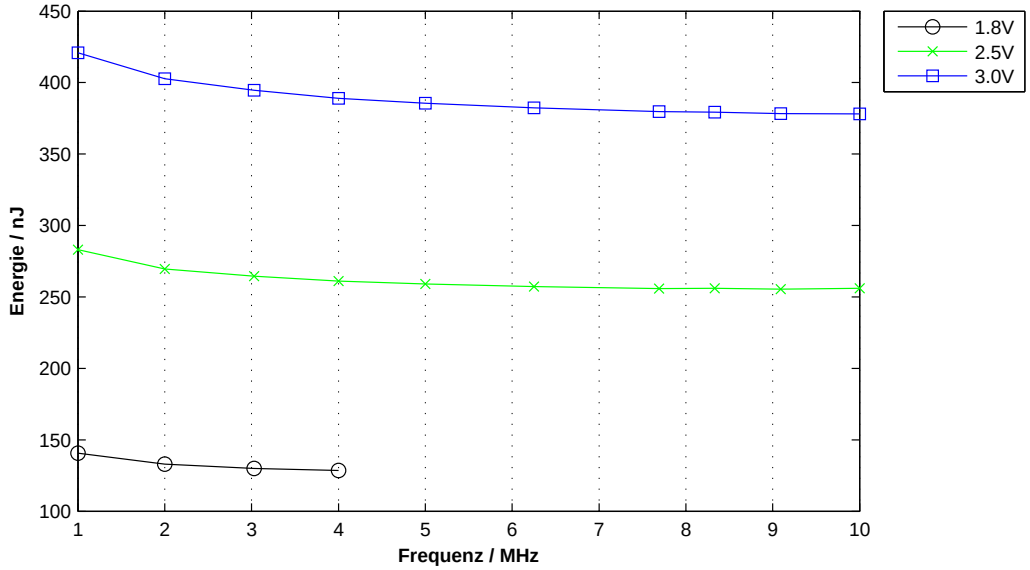


Abbildung 6.6: MSP430F2232: Energieaufnahme pro Sensormessung abhängig von der Frequenz

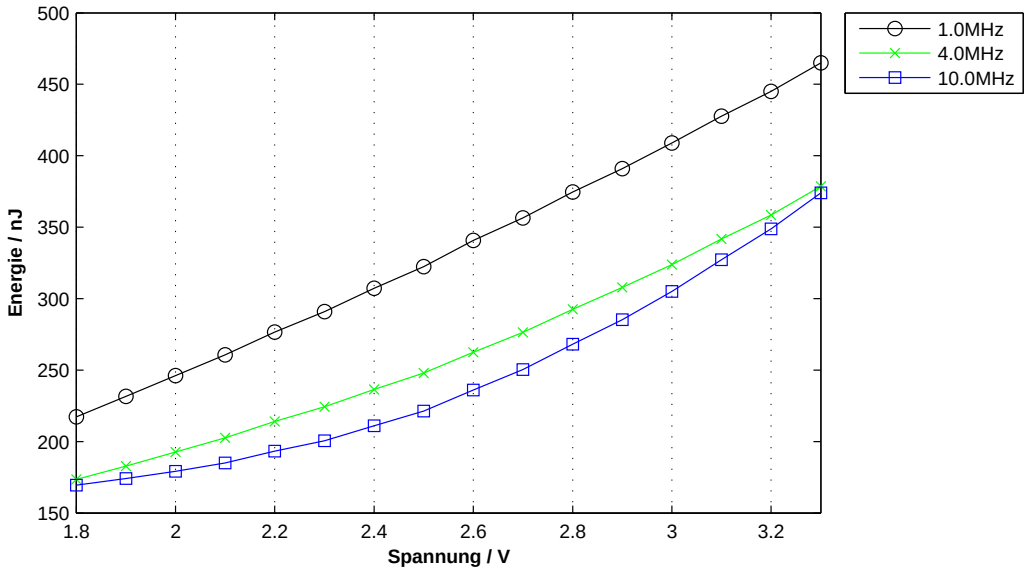


Abbildung 6.7: MSP430F5418A: Energieaufnahme pro Sensormessung abhängig von der Spannung

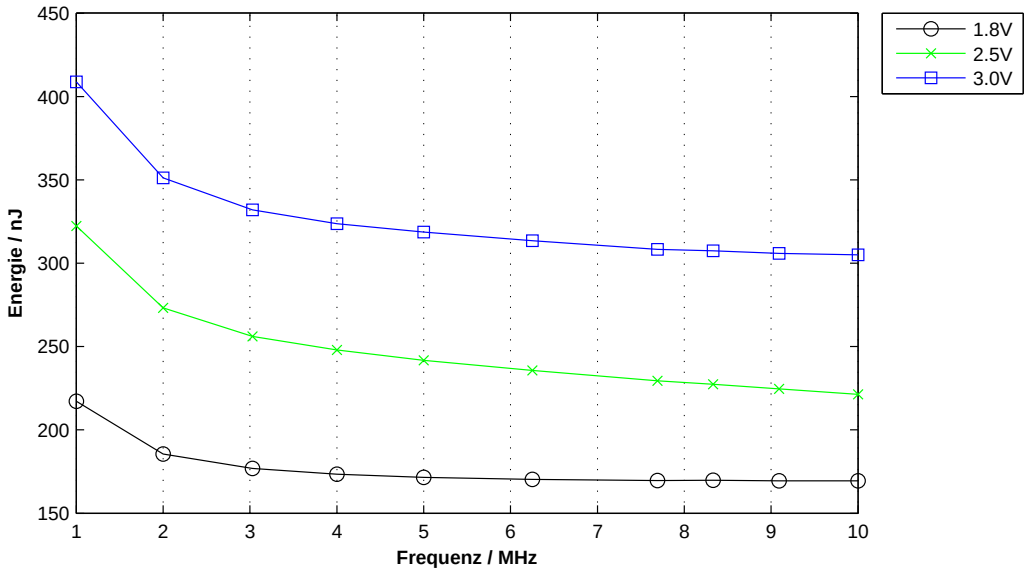


Abbildung 6.8: MSP430F5418A: Energieaufnahme pro Sensormessung abhängig von der Frequenz

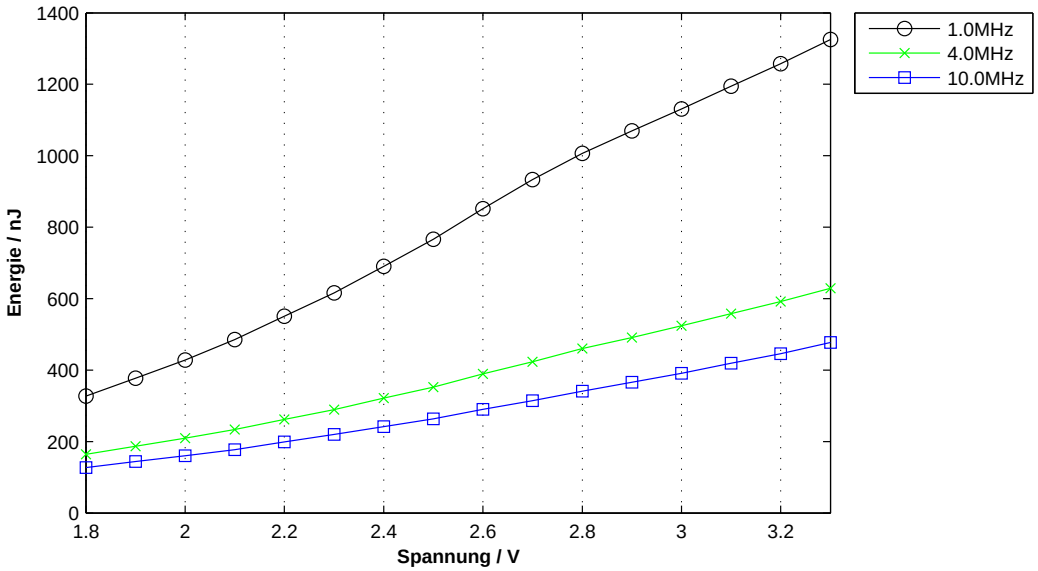


Abbildung 6.9: PIC16LF727: Energieaufnahme pro Sensormessung abhängig von der Spannung

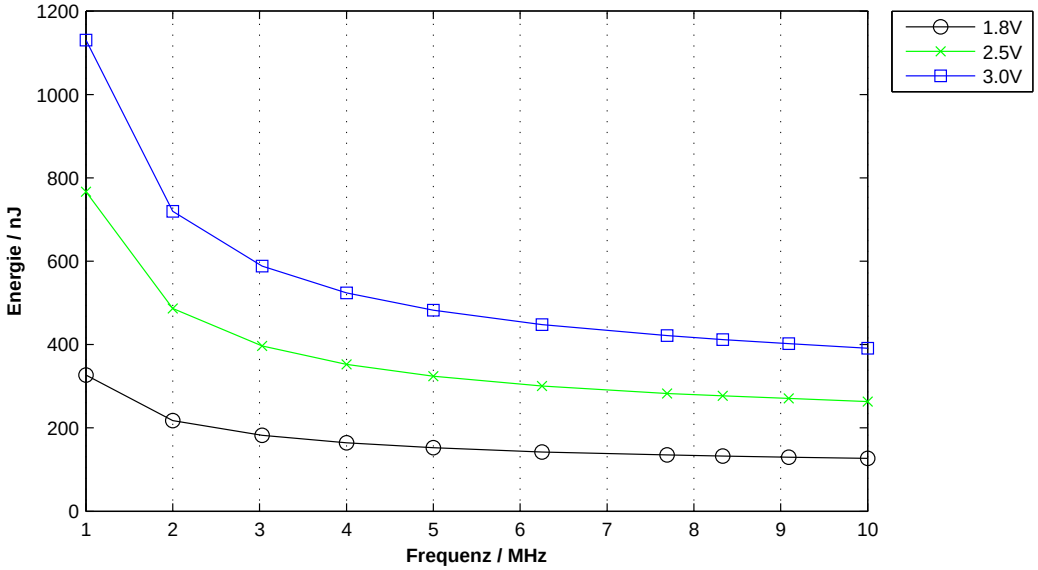


Abbildung 6.10: PIC16LF727: Energieaufnahme pro Sensormessung abhängig von der Frequenz

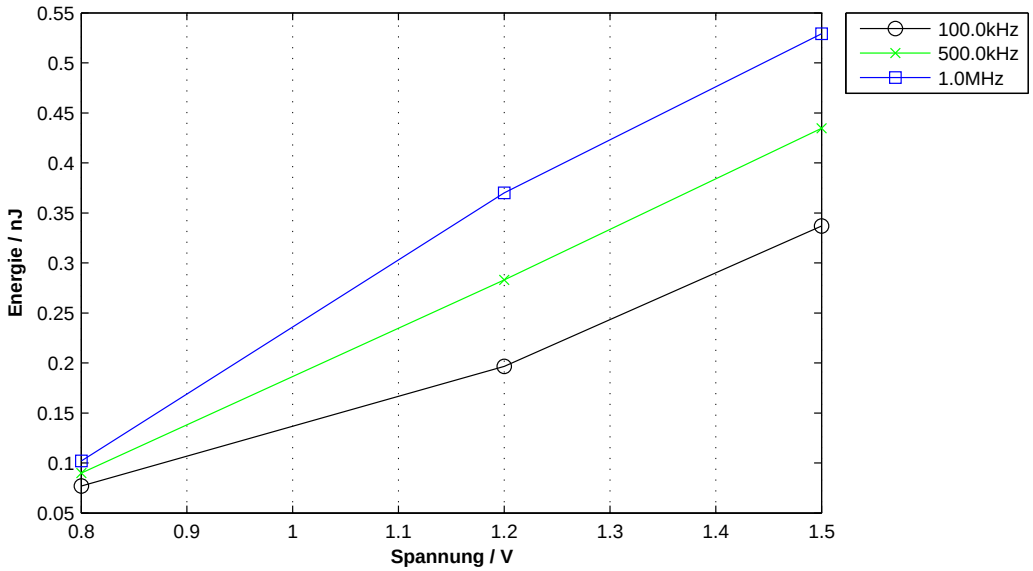


Abbildung 6.11: Testchip: Energieaufnahme pro Sensormessung abhängig von der Spannung

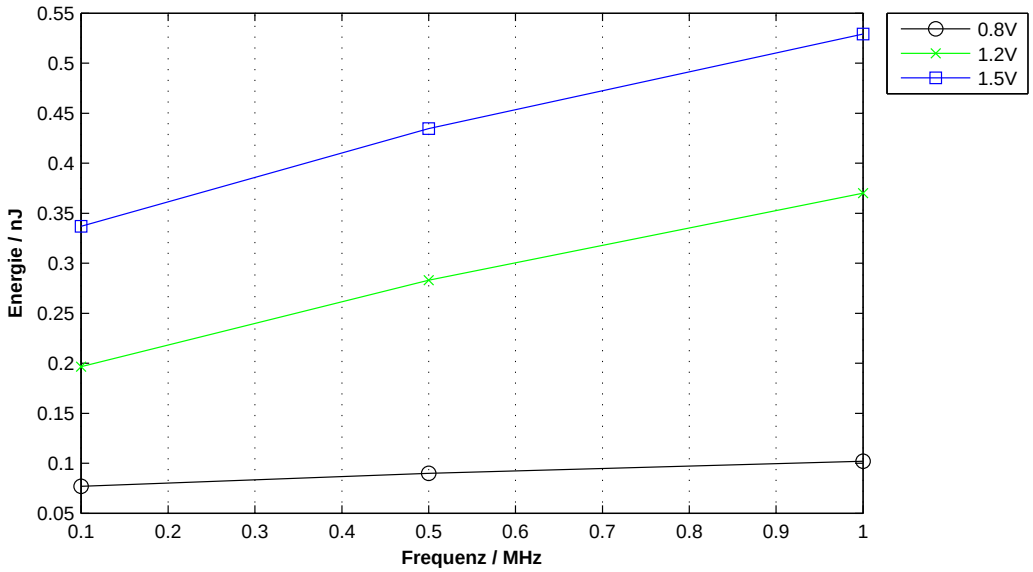


Abbildung 6.12: Testchip: Energieaufnahme pro Sensormessung abhängig von der Frequenz

6.2 Aktive Zeit der Mikrocontroller

Als wesentliche Einflussgröße wird die aktive Zeit der Mikrocontroller während der Sensormessung verglichen. Tabelle 6.1 stellt die in den Abbildungen 5.8, 5.14, 5.20, 5.26 und 5.34 ermittelten Modelle für die aktive Zeit der Mikrocontroller gegenüber.

Der frequenzunabhängige Term ist dabei der Interruptlatenz zuzurechnen, also jener Zeit, die die Mikrocontroller benötigen, um aus dem LPM in den aktiven Betriebszustand überzugehen. Der MSP430F1232 und der MSP430F2232 zeigen dabei im Vergleich größere Werte, was an dem älteren Chipdesign liegt. Der frequenzabhängige Term ist bei den MCUs bis auf den PIC16LF727 in der selben Größenordnung. Der Ausreißer lässt sich auf den Umstand zurückführen, dass der PIC16LF727 für einen internen Zyklus vier Taktzyklen benötigt. Mit dem Faktor vier umgerechnet liegt auch der PIC16LF727 in der gleichen Größenordnung wie die anderen MCUs.

Typ	aktive Zeit
ATmega88PA	$t(f) = 0{,}119\,\mu s + 351{,}8\,\mu s \cdot \frac{1}{f[\mathrm{MHz}]}$
MSP430F1232	$t(f) = 41{,}50\,\mu s + 397{,}6\,\mu s \cdot \frac{1}{f[\mathrm{MHz}]}$
MSP430F2232	$t(f) = 29{,}60\,\mu s + 370{,}3\,\mu s \cdot \frac{1}{f[\mathrm{MHz}]}$
MSP430F5418A	$t(f) = 1{,}857\,\mu s + 381{,}3\,\mu s \cdot \frac{1}{f[\mathrm{MHz}]}$
PIC16LF727	$t(f) = 3{,}394\,\mu s + 1570{,}0\,\mu s \cdot \frac{1}{f[\mathrm{MHz}]}$

Tabelle 6.1: Mikrocontroller: aktive Zeit abhängig von der Frequenz

6.3 Diskussion

In diesem Abschnitt werden die ermittelten Ergebnisse der MCUs und des Testchips gegenübergestellt. Außerdem werden die Kurven der MCUs und des Testchip in einem Diagramm gegenübergestellt. Abschließend erfolgt die Bewertung des Konzeptes der autonomen, rekonfigurbaren Architektur.

Abbildung 6.13 stellt einen Ausschnitt der Abbildung 6.15 dar und zeigt den Energieverbrauch pro Sensormessung abhängig von der Versorgungsspannung bei den MCUs. Als Taktfrequenz wurde jeweils die im vorangegangenen Abschnitt ermittelte Taktfrequenz verwendet, die den geringsten Energieverbrauch zur Folge hat. Die Kurven weisen, bis auf die des MSP430F5418A, sehr ähnliche Steigung auf. Die flachere Kurve des MSP430F5418A kann dem internen Low-Dropout Linear-Regler zugeschrieben werden, welcher bei höheren Betriebsspannungen effektiver ist.

Abbildung 6.14 zeigt den Ausschnitt der MCUs aus Abbildung 6.16 und stellt den Energieverbrauch pro Sensormessung abhängig von der Taktfrequenz dar. Die Versorgungsspannung wurde für alle MCUs mit 1,8 V für den geringsten Energieverbrauch gewählt. Auffallend ist die unterschiedliche Frequenzabhängigkeit der MCUs. Beim MSP430F1232 wird der Energieverbrauch pro MHz durch die kürzere aktive Zeit des Mikrocontrollers aufgewogen. Ebenfalls nur eine geringe Verringerung des Energieverbrauches mit steigender Taktfrequenz zeigt der MSP430F2232. Dem Gegenüber stehen der MSP430F5418A und der PIC16LF727 mit ausgeprägter Frequenzabhängigkeit. Während der MSP430F5418A bei Taktfrequenzen über 6 MHz nur eine geringe Verringerung des Energieverbrauches zeigt, profitiert der PIC16LF727 auch noch von höheren Taktfrequenzen.

Abbildung 6.15 stellt den Energieverbrauch pro Sensormessung abhängig von der Versorgungsspannung für MCUs und Testchip dar. Als Taktfrequenz wurden wiederum jene verwendet, die den geringsten Energieverbrauch zur Folge haben. Wesentliche Anzumerken ist, dass der Energieverbrauch auf der Y-Achse im logarithmischen Maßstab aufgetragen ist. Aus der Grafik ist ersichtlich, dass der Testchip einen um 3 Zehnerpotenzen geringeren Energieverbrauch aufweist als die Mikrocontroller.

Abbildung 6.16 zeigt den Energieverbrauch pro Sensormessung abhängig von der Taktfrequenz bei der jeweils geringsten Versorgungsspannung. Der Energieverbrauch ist auf der Y-Achse ebenfalls im logarithmischen Maßstab aufgetragen. Die unterschiedlichen Betriebsbereiche der MCUs und des Testchip sind genauso erkennbar wie der um 3 Zehnerpotenzen geringer Energieverbrauch des Testchip.

In Tabelle 6.2 sind die Zahlenwerte für den Energieverbrauch pro Sensormessung im optimalen Betriebspunkt für die untersuchten Mikrocontroller und den Testchip aufgelistet. Wie in Abschnitt 3.7 definiert, wird der Betriebspunkt bei welchem der Prüfling den geringsten Energieverbrauch pro Sensormessung aufweist als optimal bezeichnet. Die Versorgungsspannung konnte als wesentlicher Faktor und die geringste mögliche Versorgungsspannung als optimal bestätigt werden. Ebenfalls wesentlichen Einfluss auf das Ergebnis hat die Taktfrequenz wobei in diesem Fall zwischen MCU und Testchip unterschieden werden muss. Bei den Mikrocontrollern wird die aktive Zeit direkt von der Taktfrequenz bestimmt. Eine im Verhältnis schnellere Taktfrequenz bewirkt eine kürzere aktive Phase und damit einen geringeren Energieverbrauch. Dadurch, dass der Testchip permanent läuft, steigt hingegen der Energieverbrauch mit steigender Taktfrequenz und die geringste Taktfrequenz ist optimal für den Testchip.

Typ	Spannung	Frequenz	Energieverbrauch
ATmega88PA	1,8 V	4 MHz	110,7 nJ
MSP430F1232	1,8 V	4 MHz	105,7 nJ
MSP430F2232	1,8 V	4 MHz	128,5 nJ
MSP430F5418A	1,8 V	10 MHz	169,4 nJ
PICLF727	1,8 V	10 MHz	126,8 nJ
Testchip	0,8 V	100 kHz	69,91 pJ

Tabelle 6.2: Energieverbrauch pro Sensormessung bei optimalem Betriebspunkt

Den geringsten Energieverbrauch pro Sensormessung der MCUs beim untersuchtem Szenario hat der MSP430F1232 mit 105,7 nJ. Im Vergleich mit dem Testchip der 69,91 pJ benötigt, ist der Energieverbrauch um den Faktor 1512 größer. Die MCU mit dem höchsten Energieverbrauch im optimalen Betriebspunkt ist mit 169,4 nJ der MSP430F5418A und hat einen um Faktor 2423 größeren Energieverbrauch als der Testchip.

Aufgrund der Erkenntnisse der Messung und anschließenden Auswertung kann die in der Einleitung gestellte Frage zur Evaluierung des Ansatzes mit autonomen, rekonfigurierbaren Modulen beantwortet werden. Im betrachteten Szenario benötigt der Testchip 1512 mal weniger Energie für eine Sensormessung als der beste Mikrocontroller. Damit ist der Ansatz mit autonomen, rekonfigurierbarem Modul besser als eine reine Umsetzung mit Mikrocontroller.

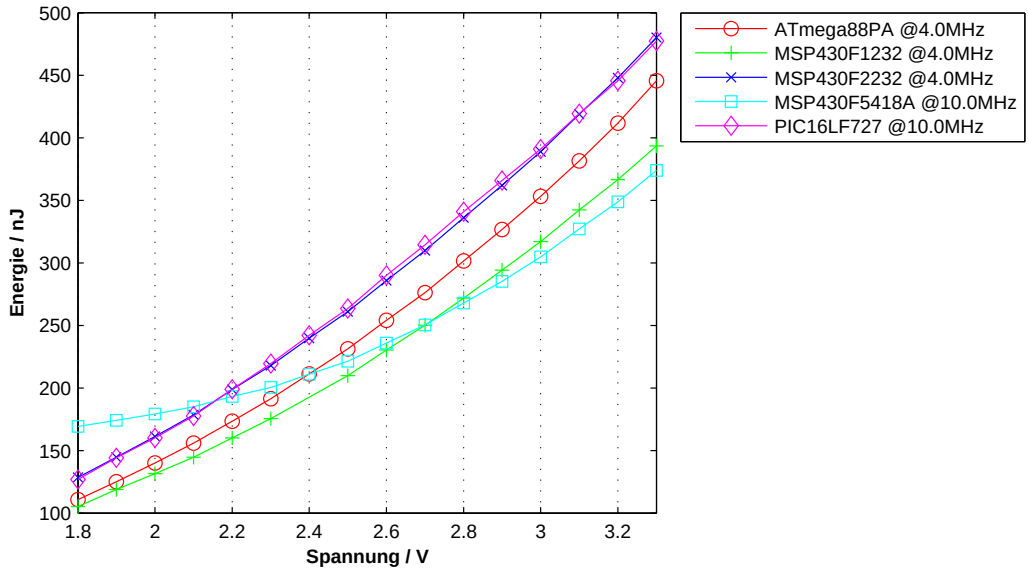


Abbildung 6.13: Mikrocontroller: Energieaufnahme pro Sensormessung abhängig von der Spannung bei optimaler Frequenz

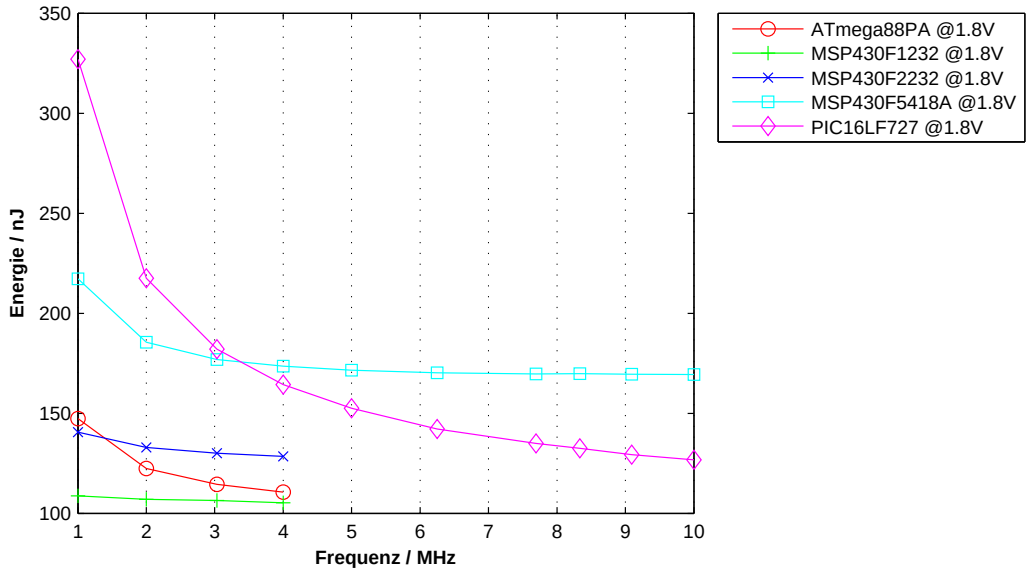


Abbildung 6.14: Mikrocontroller: Energieaufnahme pro Sensormessung abhängig von der Frequenz bei optimaler Spannung

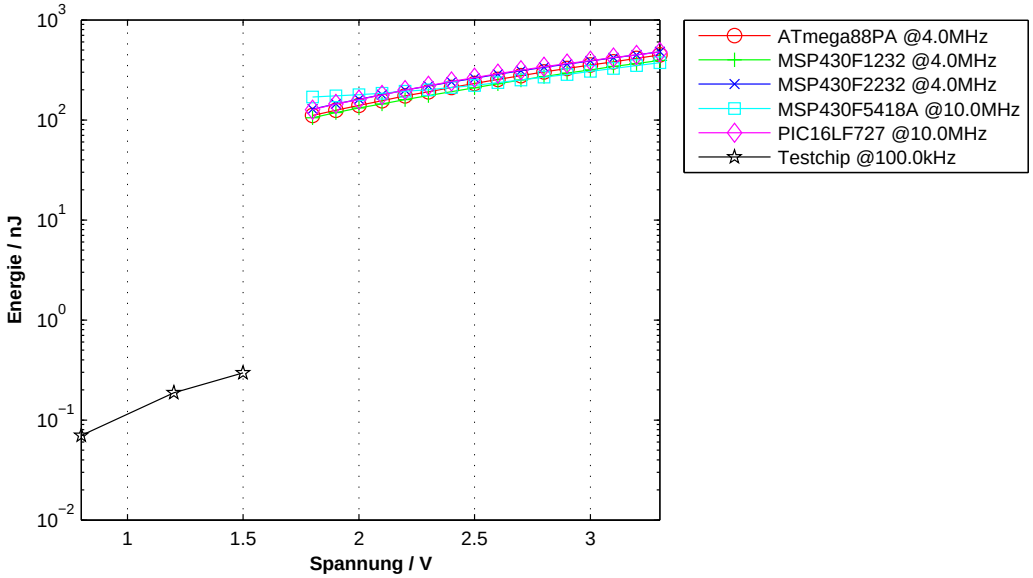


Abbildung 6.15: Mikrocontroller und Testchip: Energieaufnahme pro Sensormessung abhängig von der Spannung bei optimaler Frequenz

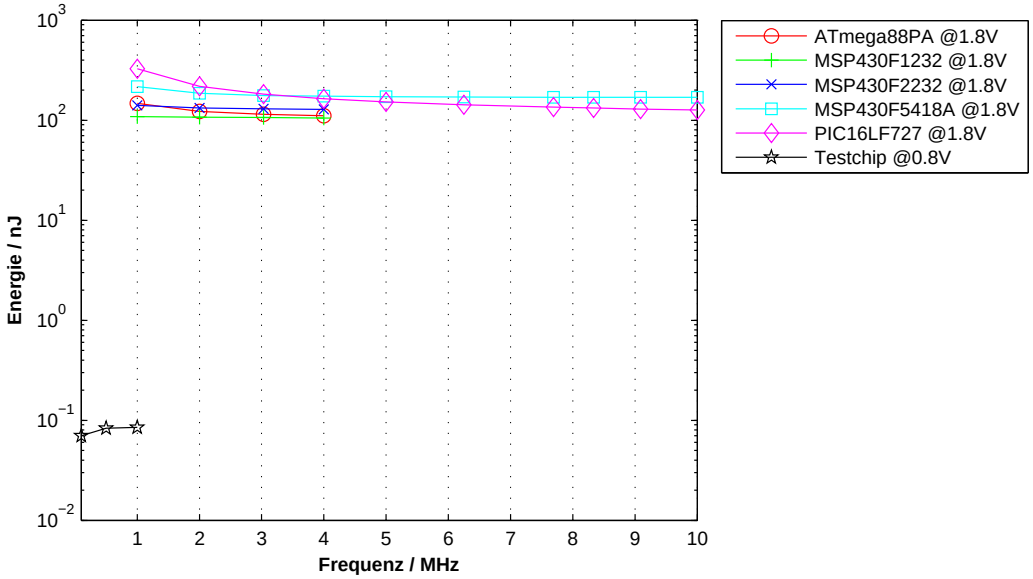


Abbildung 6.16: Mikrocontroller und Testchip: Energieaufnahme pro Sensormessung abhängig von der Frequenz bei optimaler Spannung

7 Zusammenfassung

In diesem Kapitel werden die wesentlichen Punkte der vorhergehenden Kapitel nochmals zusammengefasst. Außerdem wird auf die Herausforderung der praktischen Arbeit mit der Entwicklung der Mikrocontroller-Module eingegangen. Den Abschluss bilden Überlegungen zu weiteren Ergänzungen und Punkten die in dieser Arbeit nicht realisiert werden konnten.

Am Beginn von Kapitel 1 wurde die Motivation der Arbeit eingehend erläutert. Dabei wurde festgehalten, dass die Gründe, die für eine Reduktion des Energieverbrauches sprechen vielfältig sind. Unter anderem sind wirtschaftliche Interessen und die technische Notwendigkeit die Hauptmotive. Das Themengebiet der Arbeit wurde mit den unterschiedlichen Einsatzgebieten der WSNs definiert und viele einzelne Anwendungsfälle wie Umweltüberwachung oder Gebäudeautomatisierung präsentiert.

In Abschnitt 1.2 wurde das Ziel der Arbeit vorgestellt, welches die Evaluierung des rekonfigurierbaren Ansatzes in Hinblick auf den Energieverbrauch ist. Als Grundlage der Evaluierung wurde eine universelle und erweiterbare Messumgebung definiert, die in dieser Arbeit entwickelt wurde. Die Evaluierung sollte anhand von Ultra-Low-Power Mikrocontrollern als Referenz durchgeführt werden und abschließend eine Gegenüberstellung mit den Ergebnissen des rekonfigurierbaren Ansatzes beinhalten.

In Kapitel 2 wurde der aktuelle Stand der Technik im Bereich der integrierten Schaltungen erfasst. Dabei wurde besonderes Augenmerk auf die Reduktion des Energieverbrauches gelegt und eine Kategorisierung nach Abstraktionslevel eingeführt. Beginnend bei der eingesetzten Halbleitertechnologie mit grundlegenden Prinzipien wie Voltage Scaling. Bei den Optimierungsmöglichkeiten wurde jeweils zwischen Einfluss auf den statischen oder dynamischen Energieverbrauch unterschieden. Als nächsthöheres Abstraktionslevel wurde das Schaltungsdesign auf Transistor-Ebene und RTL betrachtet, wo Techniken wie Clock-Gating großes Einsparungspotential versprechen. Das letzte und höchste betrachtete Abstraktionslevel stellte das Schaltungsdesign auf Architektur-Ebene dar. Dort wurde eine Unterscheidung zwischen Datenpfad-orientierten und steuerungs-zentrierten Architekturen eingeführt. Dabei wurde festgestellt, dass zahlreiche Ansätze zur Optimierung der Datenpfad-orientierten Architektur existieren, aber für Steuerungsaufgaben nach wie vor Mikrocontroller eingesetzt werden. Abschließend wurde in Abschnitt 2.4 der Ansatz mit autonomen, rekonfigurierbaren Modulen eingehend betrachtet, welcher in dieser Arbeit evaluiert wurde.

Die wissenschaftliche Herleitung der Evaluierung erfolgte in Kapitel 3. Für den Vergleich wurde die Realisierung mit Mikrocontroller als Referenz gewählt, da sie dem aktuellen Stand der Technik entsprach. Der Ansatz mit autonomen, rekonfigurierbaren Modulen wurde durch den Testchip

als physische Umsetzung des Konzeptes repräsentiert. Als Bewertungskriterium wurde der Energieverbrauch pro Sensormessung definiert, weil dieses Kriterium einen besseren Schluss auf die Veränderung auf architektureller Ebene zuließ. Eine wesentliche Randbedingung war, dass beide Ansätze die gleiche Aufgabe zu erfüllen haben. Für diese Aufgabe wurde als Anwendungsfall ein SPI-Temperatursensor definiert, welcher durch einen Sensor-Emulator umgesetzt wurde, um immer gleiche Sensor-Ergebnisse zu bekommen.

Der Abschnitt 3.4 zeigte eine ausführliche Analyse der variablen Parameter und deren Einfluss auf den Strom- und Energieverbrauch. Außerdem wurden die tatsächlich zu messenden Parameter definiert und eine Selektion der Testfälle zur Reduktion der Messdauer vorgeschlagen. Eine wesentliche Grundlage zur Berechnung des Stromverbrauches pro Sensormessung und des Energieverbrauches pro Sensormessung wurde mit der Herleitung der Berechnung in Abschnitt 3.6 gelegt. Den Abschluss des Kapitels bildeten Überlegungen, in welchem Betriebspunkt ein fairer Vergleich möglich ist. Dazu wurde der optimale Betriebspunkt, also der Betriebspunkt mit dem geringsten Energieverbrauch ausgewählt.

Die Messumgebung in Kapitel 4 stellt den praktischen Teil der Arbeit dar. Aufgabe der Messumgebung war die automatisierte Messung der notwendigen Stromwerte, aus welchen der Energieverbrauch pro Sensormessung berechnet wurde. Die Funktionen der Messumgebung wurden drei Themenbereichen zugeordnet, der Steuerung des Messsystems, den Prüflingen und der eigentlichen Messung.

- Die Steuerung wurde wiederum aus mehreren Teilfunktionen erstellt. Die Ansteuerung erledigte ein Laptop mit MATLAB. Als Spannungsversorgung wurde ein Labornetzteil eingesetzt. Die Takterzeugung für die Prüflinge und die Sensor Emulation wurden mit einem SmartFusion FPGA realisiert.
- Die Prüflinge wurden in die jeweils passenden Sockel des SNOPS-Eval-Boards gesteckt. Dazu wurde für jeden Mikrocontroller ein entsprechendes Modul entwickelt, für den Testchip existierte bereits eine Trägerplatine.
- Die eigentliche Messung wurde mit Labor-Messgeräten durchgeführt. Zur Bestimmung der aktiven Zeit des Mikrocontrollers wurde ein Oszilloskop eingesetzt.

Die große Herausforderung bei der Entwicklung der Messumgebung war, die Erweiterbarkeit bei der großen Anzahl Komponenten beizubehalten. Insbesondere die Inbetriebnahme der Messumgebung wurde durch Abhängigkeiten der Komponenten untereinander erschwert.

Die Messergebnisse wurden in Kapitel 5 vorgestellt. Die Ergebnisse wurden ebenfalls automatisiert ausgewertet und grafisch aufbereitet. Dabei wurden Diagramme vom Stromverbrauch abhängig von den verschiedenen betrachten Parametern erstellt. Zu den Parametern zählten die Versorgungsspannung, die Taktfrequenz, die Anzahl der Sensormessungen im Messzeitraum, die Anzahl der Schwellwertüberschreitungen und der SPI-Busteiler. Des Weiteren wurden Diagramme vom statischen Stromverbrauch und der aktiven Zeit der Mikrocontroller abhängig von der Frequenz erstellt.

Auf Basis der Messdaten wurde die Evaluierung in Kapitel 6 durchgeführt. Dazu wurde für jeden untersuchten Mikrochip der Energieverbrauch pro Sensormessung bestimmt. Dafür musste bei den Mikrocontrollern und dem Testchip auf jeweils andere Berechnungsmethoden zurückgegriffen werden. Des Weiteren wurde die aktive Zeit der Mikrocontroller als wesentliche Einflussgröße auf den Energieverbrauch pro Sensormessung besonders hervorgehoben. Den Abschluss des Kapitels

bildeten die Gegenüberstellung der Ergebnisse der Mikrocontroller und des Testchips. Dabei wurde festgestellt, dass der beste Mikrocontroller im optimalen Arbeitspunkt 1512 mal mehr Energie pro Sensormessung benötigt als der Testchip.

Mit dieser Erkenntnis wurde die Aufgabenstellung erfüllt und es konnte bestätigt werden, dass der Ansatz mit autonomen, rekonfigurierbaren Modulen weniger Energie pro Sensormessung verbraucht als der beste untersuchte Mikrocontroller bei der gleichen Aufgabe.

Während vor dieser Arbeit nur theoretische Analysen über das Energiesparpotential des Ansatzes mit autonomen, rekonfigurierbaren Modulen existierten [GHDG09], wurde mit dieser Arbeit das Ergebnis in einer praktischen Messung bestätigt. Einschränkend muss dazu erwähnt werden, dass die Untersuchung nur bei einem Anwendungsfall durchgeführt wurde. Aufgrund der geschaffenen automatisierten Messumgebung ließen sich jedoch mit geringerem Aufwand weitere Anwendungsfälle untersuchen.

Während der Durchführung der praktischen Arbeit traten immer wieder neue Herausforderungen auf. Insbesondere die fünf untersuchten Mikrocontroller führten zu zahlreichen Verzögerungen. Die Zielsetzung alle Mikrocontroller mit einer Art HAL zu vereinheitlichen führte aufgrund der unterschiedlichen Definition der Register der einzelnen Typen zu Problemen. Während erwartungsgemäß die unterschiedlichen Mikrocontroller-Familien (ATmega, MSP430, PIC) unterschiedliche Register verwendeten, gab es auch innerhalb der MSP430-Familie unterschiedliche Interpretationen der Registerflags. Als Beispiel sei hier das SPI-Transmission Complete Flag erwähnt.

Aber auch die Messumgebung musste mit jedem neuen, in betrieb genommenen Mikrocontroller angepasst werden. Bei einigen Mikrocontrollern konnten die Konfigurationsroutinen bei zu geringer Spannung nicht durchgeführt werden und verursachten darüber hinaus Folgefehler beim unmittelbar folgenden Messpunkt. Gelöst wurde das Problem mit einer Mindestspannung während der Konfiguration der Mikrocontroller von 2 V.

Durch den großen Aufwand den die Entwicklung und Inbetriebnahme der Mikrocontroller verursacht hat, war es in der gegebenen Zeit nicht möglich die Evaluierung noch umfassender anzulegen. Zwei Punkte stehen dabei auf der Liste der Open Issues. Einerseits wäre es interessant neuere Mikrocontroller zu untersuchen, da einige der untersuchten Mikrocontroller seit über zehn Jahren am Markt sind und das Design bei bestehenden Produkten üblicherweise nicht angepasst wird. Somit sind einige der untersuchten Architekturen und Prozesstechnologien ebenfalls über zehn Jahre alt. Andererseits wäre auch eine Erweiterung um weitere Sensoren denkbar. Dabei sollte die Auswahl nicht nur auf SPI-Sensoren beschränkt bleiben, da der Testchip wie in Abschnitt 2.4 erwähnt auch weitere Schnittstellen wie I^2C und 1-Wire unterstützt. Dadurch wird die Anzahl an möglichen Sensoren und Testfällen um große Palette erweitert.

Eine weiterführende Untersuchung der in dieser Arbeit erzielten Ergebnisse sowie weitere Messungen auf Basis der entwickelten Messumgebung wurden in der Dissertation von Johann Glaser durchgeführt [Gla15].

Literatur

[Agi12a] Agilent Technologies, Inc. (Hrsg.): *Agilent 34410A/11A 6 1/2 Digit Multimeter, User's Guide*. Fifth Edition. Santa Clara, CA: Agilent Technologies, Inc., 2012

[Agi12b] Agilent Technologies, Inc. (Hrsg.): *Agilent E3631A Triple Output DC Power Supply, User's Guide*. Santa Clara, CA: Agilent Technologies, Inc., April 2012

[Ana09] Analog Devices, Inc. (Hrsg.): *ADT7310: ±0.5°C Accurate, 16-Bit Digital SPI Temperature Sensor*. One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A: Analog Devices, Inc., 2009. – Rev. 0

[Atm11] Atmel Corporation (Hrsg.): *8-bit Atmel Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash: ATmega48A, ATmega48PA, ATmega88A, ATmega88PA, ATmega168A, ATmega168PA, ATmega328, ATmega328P*. 8271D-AVR-05/11. San Jose, CA: Atmel Corporation, Mai 2011

[BMM01] Benini, Luca ; Micheli, Giovanni D. ; Macii, Enrico: Designing Low-Power Circuits: Practical Recipes. In: *IEEE Circuits and Systems Magazine* 1 (2001), First Quarter, Nr. 1, 6-25. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=928306

[BRR08] Basha, Elizabeth A. ; Ravela, Sai ; Rus, Daniela: Model-based monitoring for early warning flood detection. In: *Proceedings of the 6th ACM conference on Embedded network sensor systems* ACM, 2008, S. 295–308

[CCMM04] Carvalho, Ewerson ; Calazans, Ney ; Moraes, Fernando ; Mesquita, Daniel: Reconfiguration Control for Dynamically Reconfigurable Systems. In: *DCIS'04: 19th International Conference on Design of Circuits and Integrated Systems*, 2004, 405-410

[CLC04] Ciccarelli, L. ; Lodi, A. ; Canegallo, R.: Low leakage circuit design for FPGAs. In: *Proceedings of the IEEE 2004 Custom Integrated Circuits Conference, 2004*, 2004, 715-718

[CLCC09] Cao, Huasong ; Leung, Victor ; Chow, Cupid ; Chan, Henry: Enabling technologies for wireless body area networks: A survey and outlook. In: *Communications Magazine, IEEE* 47 (2009), Nr. 12, S. 84–93

[Das09] Dasgupta, Ranjan: Low Power MCU Selection Criteria and Sleep Mode Implementation using Hardware/Software CoDesign Technique / Innovation Lab, Tata Consultancy Services Limited, Kolkata. Version: 2009. http://www.techonline.com/article/pdf/showPDF.jhtml?id=2186001281. 2009. – Forschungsbericht

[Dav08] Davies, John H.: *MSP430 Microcontroller Basics*. Elsevier, 2008

[DM95] Devadas, Srinivas ; Malik, Sharad: A Survey of Optimization Techniques Targe-

ting Low Power VLSI Circuits. In: *32nd Conference on Design Automation, 1995. DAC '95*, 1995, 242-247

[DRTAMA12] Díaz-Ramírez, Arnoldo ; Tafoya, Luis A. ; Atempa, Jorge A. ; Mejía-Alvarez, Pedro: Wireless sensor networks and fusion information methods for forest fire detection. In: *Procedia Technology* 3 (2012), S. 69–79

[Fre07] Freescale Semiconductor, Inc. (Hrsg.): *M68HC11 Reference Manual*. Rev. 6.1. Denver, CO: Freescale Semiconductor, Inc., 2007

[GBW+11] Glaser, J. ; Brame, F. ; Wenninger, J. ; Haase, J. ; Herndl, T. ; Vienna University of Technology, Infineon Technologies Austria AG (Hrsg.): *A Novel Reconfigurable Architecture for Control and Computation*. Rev. 1.0. Vienna: Vienna University of Technology, Infineon Technologies Austria AG, 2011

[GDHG11] Glaser, Johann ; Damm, Markus ; Haase, Jan ; Grimm, Christoph: TR-FSM: Transition-based Reconfigurable Finite State Machine. In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 4 (2011), August, Nr. 3, S. 23:1–23:14

[GGHG11] Glaser, Johann ; Gravogl, Klaus ; Haase, Jan ; Grimm, Christoph: A Reconfigurable Architecture for Ultra-Low Power Wireless Sensors. In: *The Mediterranean Journal of Electronics and Communications (MEDJEC)* 7 (2011), Nr. 3, S. 255–266

[GHDG09] Glaser, Johann ; Haase, Jan ; Damm, Markus ; Grimm, Christoph: Investigating Power-Reduction for a Reconfigurable Sensor Interface. In: *Proceedings of Austrochip 2009*. Graz, Austria, 7. Oktober 2009

[Gla15] Glaser, Johann: *Design Methodology for Custom Reconfigurable Logic Architectures*, TU Wien, Diss., 2015

[GZR99] George, Varghese ; Zhang, Hui ; Rabaey, Jan: The Design of a Low Energy FPGA. In: *International Symposium on Low Power Electronics and Design*, 1999, S. 188–193

[Har01] Hartenstein, Reiner: Coarse Grain Reconfigurable Architecture. In: *Proceedings of the Asia South Pacific Design Automation Conference*. Yokohama, Japan, 2001, 564-570

[Hit07] Hitzelberger, Christoph: *LowPower Design Methoden für VLSI CMOS Digitalschaltungen*. Saarbrücken, Universität des Saarlandes, Diss., 2007

[HVH99] Hwang, Enoch ; Vahid, Frank ; Hsu, Yu-Chin: FSMD Functional Partitioning for Low Power. In: *Design, Automation and Test in Europe Conference and Exhibition 1999. Proceedings*. Munich, Germany, 9.–12. März 1999, 22-28

[IK11] Ivanov, Ivan ; Kazmierski, Tom: Extremely Low-Power Circuits Based on Pass-Transistor Logic in Sub-Threshold Region. In: *Proceedings of the Virtual Worldwide Forum for PhD Researchers in Electronic Design Automation VW-FEDA*. Southampton, UK, 28. November – 2. Dezember 2011

[Ive11] Ivey, Brant: *Low-Power Design Guide*. DS01416A. Chandler, AZ : Microchip Technology Inc., 2011. – Application Note 1416

[KHZS07] Kansal, Aman ; Hsu, Jason ; Zahedi, Sadaf ; Srivastava, Mani B.: Power management in energy harvesting sensor networks. In: *ACM Transactions on Embedded Computing Systems (TECS)* 6 (2007), Nr. 4, S. 32

[KPM10] Khedo, Kavi K. ; Perseedoss, Rajiv ; Mungur, Avinash: A wireless sensor network air pollution monitoring system. In: *arXiv preprint arXiv:1005.1737* (2010)

[Mic09] Microchip Technology Inc. (Hrsg.): *PIC16F72X/PIC16LF72X Data Sheet*. DS41341E. Chandler, AZ: Microchip Technology Inc., März 2009

[Mic13] Microsemi Corporation (Hrsg.): *SmartFusion Customizable System-on-Chip (cSoC), Datasheet*. Revision 12. Aliso Viejo, CA: Microsemi Corporation, 2013

[MKC02] Mattos, Júlio C. B. ; Kreutz, Márcio ; Carro, Luigi: Low-power control architecture for embedded processors. In: *15th Symposium on Integrated Circuits and Systems Design, 2002. Proceedings.*, 2002, 221-226

[OMCH+07] O'Flynn, B ; Martinez-Català, R ; Harte, S ; O'Mathuna, C ; Cleary, J ; Slater, C ; Regan, F ; Diamond, D ; Murphy, H: SmartCoast: a wireless sensor network for water quality monitoring. In: *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on* Ieee, 2007, S. 815–816

[ÖPR+07] Österlind, Fredrik ; Pramsten, Erik ; Roberthson, Daniel ; Eriksson, Joakim ; Finne, Niclas ; Voigt, Thiemo: Integrating building automation systems and wireless sensor networks. In: *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on* IEEE, 2007, S. 1376–1379

[ORT14] International Technology Roadmap for Semiconductors: *Summary 2013 Overall Roadmap Technology Characteristics Technology Trend Targets*. April 2014

[RCN03] Rabaey, Jan M. ; Chandrakasan, Anantha ; Nikolic, Borivoje: *Digital Integrated Circuits*. Upper Saddle River : Prentice Hall, 2003

[RN00] Rao, M. S. ; Nandy, S. K.: Power Minimization Using Control Generated Clocks. In: *37th Design Automation Conference, Proceedings*, 2000, 794–799

[SNBN06] Sydow, T. von ; Neumann, B. ; Blume, H. ; Noll, T. G.: Quantitative Analysis of Embedded FPGA-Architectures for Arithmetic. In: *International Conference on Application-specific Systems, Architectures and Processors, 2006. ASAP '06.*, 2006, 125-131

[Tex04] Texas Instruments (Hrsg.): *MSP430x11x2, MSP430x12x2 Mixed Signal Microcontroller*. SLAS361D. Dallas, Texas: Texas Instruments, August 2004

[Tex10] Texas Instruments (Hrsg.): *MSP430F543xA, MSP430F541xA Mixed Signal Microcontroller*. SLAS655B. Dallas, Texas: Texas Instruments, Oktober 2010

[Tex11] Texas Instruments (Hrsg.): *MSP430F22x2, MSP430F22x4 Mixed Signal Microcontroller*. SLAS504F. Dallas, Texas: Texas Instruments, Juli 2011

[Tex13] Texas Instruments (Hrsg.): *MSP430x2xx Family User's Guide*. SLAU144J. Dallas, Texas: Texas Instruments, Juli 2013

[TRD+07] Tuan, Tim ; Rahman, Arif ; Das, Satyaki ; Trimberger, Steve ; Kao, Sean: A 90-nm Low-Power FPGA for Battery-Powered Applications. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26 (2007), Februar, Nr. 2, 296-300. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4068935

[TS02] Tietze, Ulrich ; Schenk, Christoph: *Halbleiter-Schaltungstechnik*. 12. Auflage. Springer-Verlag, 2002

[TVF08] Tavares, Jorge ; Velez, Fernando ; Ferro, João: Application of wireless sensor networks to automobiles. In: *Measurement Science Review* 8 (2008), Nr. 3, S. 65–70

[Xil07] Xilinx, Inc. (Hrsg.): *XC9572XL High Performance CPLD*. v. 2.0. San Jose, CA: Xilinx, Inc., April 2007