Charakterisierung und Analyse vom Reverse and Forward Body Biasing als Durchsatz- und Leistungsoptimierungstechnik für Multi-Core Mikrocontroller

Master Thesis

Fachhochschule Dortmund
Interessengemeinschaft für Mikroelektronik und Eingebettete Systeme
Fachbereich: Informations- und Elektrotechnik

Zur Erlangung des akademischen Grades
Master of Engineering

In Zusammenarbeit mit Infineon Technologies AG - Standort Villach

Jan-Morten Reiners

Prüfer: Prof. Dr. Michael Karagounis
Betreuer: Dr. Frank Prämassing

Eidesstattliche Erklärung

„Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.“

_____ _____

Ort, Datum Unterschrift

Kurzfassung

Diese Masterthesis beschäftigt sich im Rahmen des Testchips TC1.5 der Infineon Austria AG mit der Analyse und Charakterisierung des Reverse und Forward Body Biasing mit differentieller Spannungsskalierung. In einem theoretischen Grundlagenteil werden dem Leser zunächst die Beweggründe sowie die nötigen Informationen der zugrunde liegenden Halbleiter-Technologie vermittelt, um ihn an die Thematik des Body Biasing und der Power Management Einheiten heranzuführen. Es folgt die Beschreibung des AurixPlus-digital EVR and PMS Testchips (Version 1.5) hinsichtlich seiner Struktur und Funktionalität. Den Kern der Arbeit bilden der Aufbau eines teilweise automatisieren Messplatzes, die Entwicklung einer Testsoftware (Python, C#), die Erarbeitung von Test Spezifikationen sowie die Durchführung entsprechender Messungen zur Analyse und Charakterisierung. Die Ergebnisse dieser Messungen werden im Anschluss eingehend besprochen und mit Bezug auf zukünftige Entwicklungen in diesem Bereich bewertet.

Abstract

This master thesis deals with the analysis and characterization of reverse and forward body biasing with differential voltage scaling as part of the test chip TC1.5 of Infineon Austria AG. In a theoretical basic part, the reader is first introduced to the motives and the necessary information of the underlying semiconductor technology in order to introduce him to the topic of body biasing and power management units. The following is a description of the AurixPlus-digital EVR and PMS test chip (version 1.5) regarding its structure and functionality. The main part of the work is the construction of a partially automated measuring station, the development of a test software (Python, C#), the development of test specifications as well as the execution of corresponding measurements for analysis and characterization. The results of these measurements will then be discussed in detail and evaluated regarding future developments in this area.

Inhaltsverzeichnis

1 Einleitung

Die Master Thesis wurde an der Fachhochschule Dortmund in Kooperation mit der Infineon Austria AG in Villach angefertigt. Das Thema dieser Arbeit beschäftigt sich mit der Charakterisierung der Optimierungsmethode Reverse und Forward Body Biasing in Power Management Einheiten in Mikrokontrollern. Ziel dieser Arbeit ist es, durch reale Messergebnisse und die Charakterisierung eines Testchips (TC1.5) fundierte Rückschlüsse bezüglich der Tauglichkeit dieses Optimierungsansatzes ziehen zu können. Für die Erreichung dieses Ziels war die Entwicklung eines Testsystems zur Durchführung der Charakterisierung eines Low-Drop-Out Spannungsreglers (LDO) und einer Charge Pump (CP) sowie der Qualifizierung des Reverse and Forward Body Biasing (RBB und FBB) Features mit differentieller Spannungsskalierung (DVS) notwendig. Dieses Testsystem umfasst eine grafische Benutzeroberfläche (GUI), über die das Messsystem gesteuert und gestartet werden kann. Die Steuerung verschiedener Messinstrumente wurde ebenfalls in die entwickelte Software integriert, welche in der Programmiersprache C# geschrieben wurde. Zusätzlich zu den Messinstrumenten für die Erfassung von Spannungen und Strömen wurden auch Spannungs- und Stromquellen eingebunden, um definierte Größen in den Messaufbau einbringen zu können. Darüber hinaus wurde die Steuerung eines Thermostream in die Software integriert, um die Messungen jeweils bei -40°C, 25°C und 125°C durchführen zu können. Diese Temperaturbereiche sind für die Validierung von Produkten in der Automobilindustrie als Standard definiert und dienen zur Qualitätsüberprüfung und zur Robustheitsbestimmung. In dieser Arbeit werden Aufbau und Ablauf jeder einzelnen Charakterisierungsmessung vorgestellt. Anschließend wird die Messdatenanalyse mittels Python Skript beschrieben. Das Post-Processing erfolgt für jede Messung durch ein eigenes angepasstes Skript und liefert für jede Messung aussagekräftige Plots. Diese Plots werden dann in der Messdatenauswertung vorgestellt, die beobachteten Probleme aufgezeigt und eine Qualifizierung von RBB und FBB durchgeführt.

1.1 Motivation

Ein wesentliches Motiv für die Durchführung dieser Arbeit war die vollständige Automatisierung des Messablaufs, welche die autonome Konfiguration des Testchips und die selbständige Steuerung der Messhardware erlaubt. Für die erfolgreiche Umsetzung dieses Zieles wurde im Vorfeld ein Kurs zur Erlernung der Programmiersprachen C# und Python belegt. In diesem Kurs wurde der allgemeine Umgang mit den zuvor genannten Programmiersprachen vermittelt. Ein weiteres Motiv für die Durchführung dieser Arbeit ist persönlicher Natur und bezieht sich auf den fünf monatigen Auslandsaufenthalt am Standort der Infineon Austria AG in Villach. Durch diesen Aufenthalt war es möglich, die Arbeitsweise eines Unternehmens dieser Größe zu studieren und einen tiefen Einblick in

Themengebiete der industriellen Schaltungsentwicklung zu erlangen, die an einer Hochschule nicht vermittelt werden können.

2 Historische Betrachtung über die Entwicklung in der Chipfertigung

In den letzten Jahrzehnten ist aufgrund gestiegener Design-Komplexität in Mikrocontrollern und Prozessoren das Powermanagement immer wichtiger geworden. Durch den Einsatz von Mikrocontrollern in mobilen Geräten wie Tablets, Smartphones oder Elektrofahrzeugen ist es notwendig, die umgesetzte Energie zu reduzieren. Zu Beginn der Chipfertigung wurden Bipolartransistoren verwendet, die auf Grund der Basisströme höhere statische Leistungsaufnahmen bewirken, als MOS Transistoren die verlustlos gesteuert werden können. Hinzu kommt, dass durch die relativ simplen Anwendungen und die geringe Anzahl an eingesetzten Transistoren mit geringer Taktung das Thema des Powermanagements noch nicht so im Focus stand, wie heutzutage. Der Fokus wurde mit dem Einsatz von Mikrocontrollern in Geräten, die nicht von einem Netz betrieben werden, relevant. Daher wurden im Laufe der Zeit Modelle zur Reduktion der benötigten Energie entworfen. Zum einen wurde intelligente Software zur Reduktion des Rechenaufwands entwickelt, zum anderen wurden Hardwaremechanismen implementiert, die intern das Powermanagement steuern und

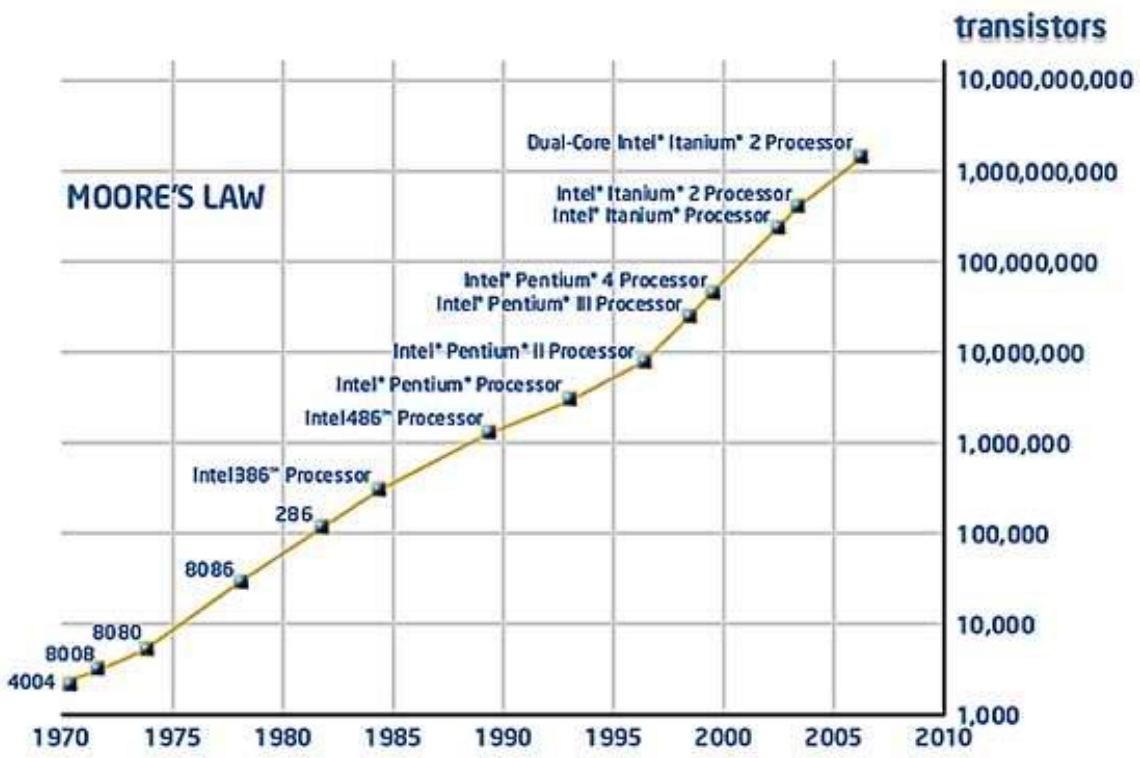


Abbildung 1: Moores Law, Verdoppelung der Transistoren in Prozessoren alle 2 Jahre

[Quelle: http://www.inf.fu-berlin.de/lehre/WS12/PS_PP/slides/Architektur_Parallele_Platformen.pdf , Bild: http://njtechreviews.com/wp-content/uploads/2011/09/varian-moores-law-graph.gif]

dadurch den Stromverbrauch senken. In den 80er Jahren bestand ein Mikrocontroller aus ca. 100.000 Transistoren. Das Mooresche Gesetz, welches auf Grund von Beobachtungen in den 70er Jahren aufgestellt wurde, besagt, dass sich alle zwei Jahre die Anzahl der verbauten Transistoren verdoppelt (Abbildung 1)[23]. So konnten in den letzten Jahrzehnten durch die Entwicklung neuer Fertigungstechnologien, immer mehr Funktionen und Anwendungen auf gleicher Chipfläche implementiert werden, so dass sogar die Schwelle von 1 Milliarde Transistoren 2005 erreicht werden konnte. Dieser Trend setzt sich bis heute fort. Zurzeit werden im aktuellen Prozessor AMD Ryzen7 1800+ ca. 5 Milliarden Transistoren verbaut. In sehr großen FPGAs wie z.B. dem Virtex Ultra Scale+, der in 20nm und in 16nm CMOS-Technologie gefertigt wird, werden bis zu 20 Milliarden Transistoren integriert[2]. Wegen dieser Entwicklung steigen die Anforderungen an das Power Management zur Reduktion der Verlustleistung stetig. Eine Reduktion der Verlustleistung, steigert die Akkulaufzeit von Mobiltelefonen und Tablets oder die Reichweite elektrischer Fahrzeuge. Aus diesem Grund werden heutzutage auch geringe Einsparungspotentiale für die Reduktion des Stromverbrauchs verfolgt und durch Mechanismen im Chip-Design und in der Software integriert (Abbildung 2).

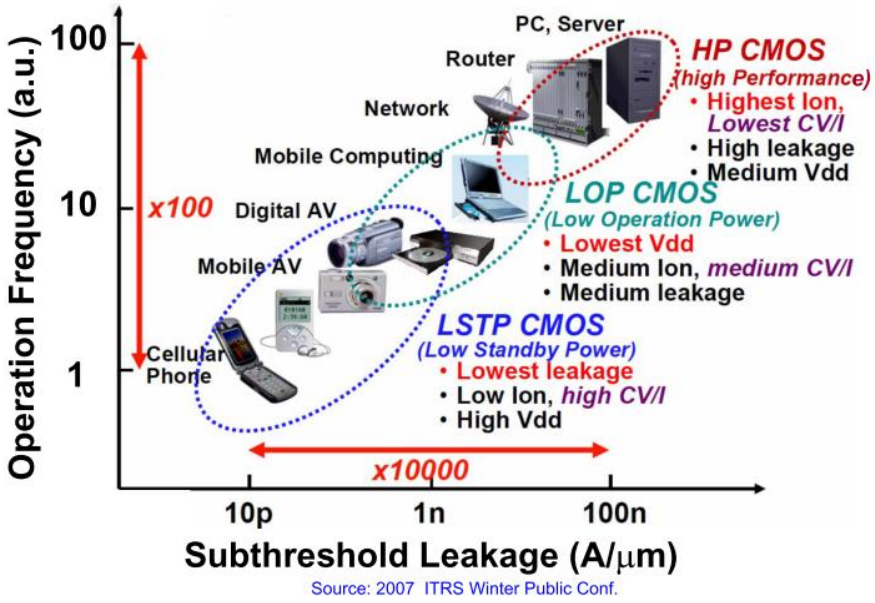


Abbildung 2: Heutige Einsatzmöglichkeiten und Klassifizierung der einzelnen der Anwendungsfälle [34]

2.1 Gründe der steigenden Anforderungen

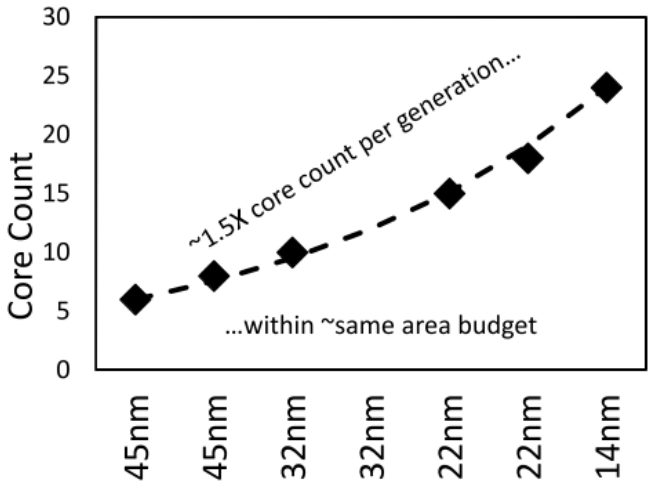


Abbildung 3: Stetige Steigerung der Anzahl der Kerne durch jeden Technologieknoten

Die hohe Anzahl an Transistoren und die stets steigende maximale Taktrate führen dazu, dass eine enorme Menge an elektrischer Energie in Wärme umgesetzt wird. Dadurch stellt sich die Herausforderung, die umgesetzte Leistung zu reduzieren und die Wärme besser abzuleiten. Hinzu treten weitere physikalische Effekte auf, die den Stromverbrauch ansteigen lassen. Die maximale Taktrate eines Prozessors wird durch den Transistor und seine physikalischen Eigenschaften insbesondere seiner maximalen Schaltgeschwindigkeit bestimmt. Die Industrie befindet sich bei jedem eingeführten Technologieknoten (Abbildung 3) an der Grenze des zu diesem Zeitpunkt technisch machbaren, sodass andere Möglichkeiten zur Steigerung der Leistung eines Prozessors gesucht werden müssen. Die einfachste Möglichkeit zur Leistungssteigerung ist es, die Anzahl der Recheneinheiten auf einem Chip zu erhöhen [25], wodurch die Leistung eines Prozessors nun nicht mehr nur an der verwendeten Frequenz, sondern auch an der Anzahl der im Prozessor parallel arbeitenden Kerne ausgemacht werden kann.

Durch die Erhöhung der Rechenkerne steigt auch die entstehende Wärme, die aufgrund der erhöhten Anzahl an Transistoren hervorgerufen wird. Wärme, die innerhalb des Chips entsteht, beeinflusst jeden einzelnen integrierten Transistor. Bei steigender Temperatur entstehen nämlich mehr freie Ladungsträger unter dem Gate, sodass der Kanal-Widerstand zwischen Drain und Source kleiner wird und mehr Strom über diese Strecke fließen kann. Durch diese physikalische Eigenschaft eines Transistors, verbraucht ein aufgewärmter Prozessor bei gleicher Auslastung viel mehr Strom als im kälteren Zustand. Der erhöhte Strom führt wiederum zu einer größeren Wärmeentwicklung auf dem Chip.

Im Folgenden werden Mechanismen, Methoden und Techniken zur Reduktion der Energie und effektivere Ableitungsmechanismen der entstandenen Wärme erläutert, um die Anforderungen an

heutige Mikrocontroller zu erfüllen. Damit die produzierte Wärme effektiv abgeleitet werden kann, werden möglichst große Metallflächen auf der Vorder- und Rückseite des Chips aufgetragen [4], wodurch sich ein besserer Wirkungsgrad bei der Kühlung des Chips einstellt. Der Mikrocontroller ist im Optimalfall mit einer externen Kühlung ausgestattet, wie sie in einem herkömmlichen PC realisiert ist. In diesem Fall sitzt der Lüfter direkt auf dem Chip, welcher die entstandene Wärme effektiv abführt. Dieses Verfahren kann jedoch in der Automobilindustrie aus Kostengründen und der schwierigen Unterbringung eines solchen Systems in mobilen Fahrzeugen nicht in Betracht gezogen werden. Daher wurden andere Konzepte zur Kühlung entwickelt. Eine Möglichkeit besteht darin, die Flip-Chip Technik zum Einbau des Chips in das Gehäuse zu verwenden [25]. Dabei wird auf dem Chip eine komplett geschlossene metallische Fläche über die Passivierung aufgetragen und überkopf in das Gehäuse eingesetzt. Das Gehäuse besitzt auf der Unterseite das sogenannte Exposed-Pad, welches zum einen für die Masse (GND) Verbindung genutzt und zum anderen für die Ableitung der Wärme verwendet wird.

Durch die stetige Weiterentwicklung der Fertigungstechnologie (siehe Abbildung 4) wurde die Core-Spannung des Transistors stetig verringert [27]. Dies führt im Betrieb des Microcontrollers zu einer Reduktion der benötigten Leistung. Außerdem wurde die allgemeine Betriebsspannung (VDD) heruntergesetzt, um dem steigenden Strom, der durch die stetig steigende Anzahl an verbauten Transistoren hervorgerufen wird, entgegen zu wirken. Leckströme bzw. Leackage können in zwei unterschiedliche Arten eingeteilt werden und zwar in statisches und der dynamisches Leakage [28]. Sowohl der statische als auch der dynamische Leakage können dabei durch Mechanismen wie dem Reverse-Body-Biasing und dem Differential Voltage Scaling verringert werden (Abbildung 5).

Year	Technology Node(nm)	Physical Gate(nm)	tox (nm)	Dielec-tric K	Vdd (V)	Vth (V)	Na (/cm^3)	Nd (/cm^3)	xj (nm)
2001	130	90	3.0	3.7	1.2	0.34	1.0e16	1.0e19	67.5
2004	90	53	2.4	3.0	1.1	0.32	1.4e16	1.4e19	46.7
2007	65	32	1.7	2.5	0.9	0.29	2.0e16	2.0e19	33.8
2010	45	22	1.5	2.0	0.8	0.29	2.9e16	2.9e19	23.4
2013	32	16	1.4	1.9	0.7	0.25	4.0e16	4.0e19	16.6
2016	22	11	1.3	1.7	0.6	0.22	5.9e16	5.9e19	11.4

Abbildung 4: Stetige Reduzierung von VDD mit jedem neuen Technologieknoten [5]

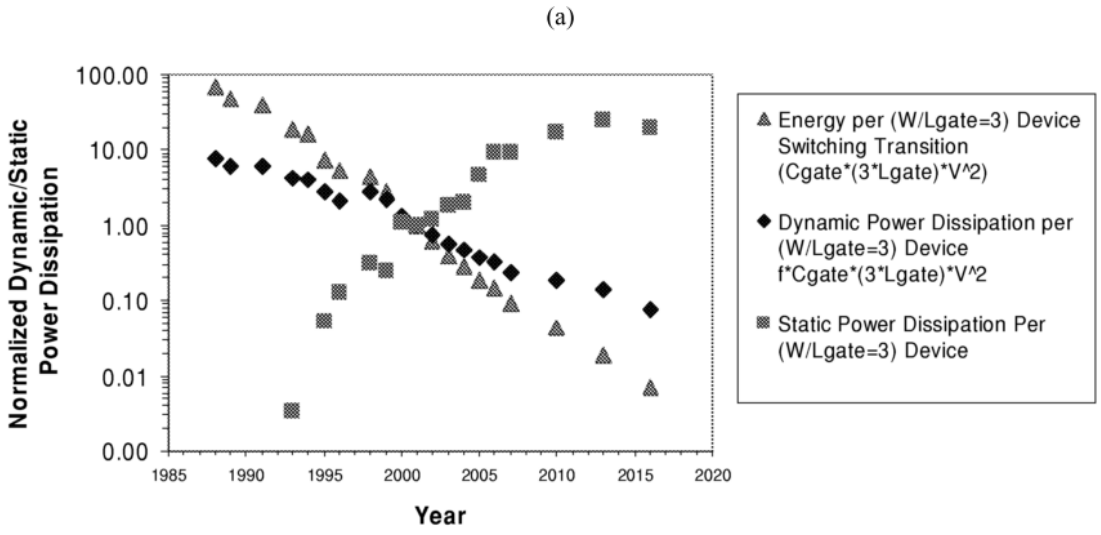


Abbildung 5: Statische und Dynamische Leistung über Technologieknoten [28]

2.2 Statischer Leakage

Statischer Leakage wird von der Anzahl der verbauten Transistoren, der Betriebsspannung, der Core-Spannung, der Temperatur und der Fertigungstechnologie beeinflusst. In Abbildung 6 sind die Mechanismen der statischen Leckströme in heutigen MOSFET-Transistoren gezeigt und in der folgenden Aufzählung benannt:

- PN-Junction Leakage (I1)
- Subthreshold Leakage (I2)
- Tunneleffect in and through Gate Oxid (I3)
- Injection of Hot Carriers from Substrat Gate Oxid (I4)
- Gate Induced Drain Leakage (GIDL) (I5)
- Punchthrough Effekt (I6)

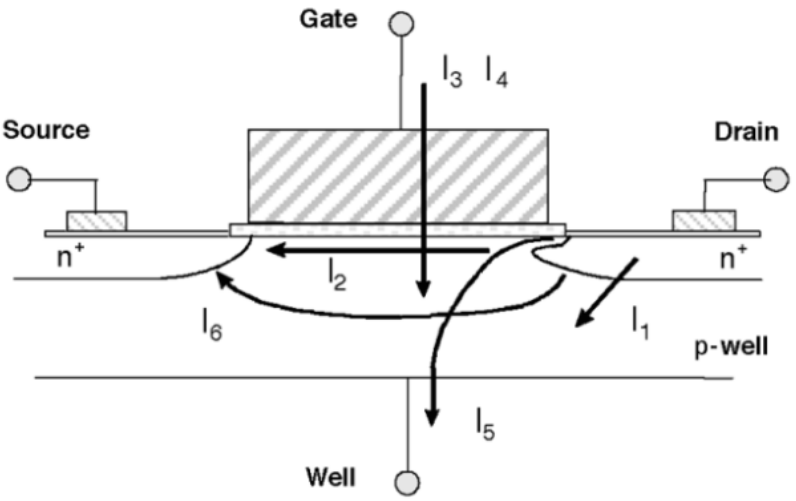


Abbildung 6: Ströme/Verlustströme im Transistor [28]

Dabei sind der „pn-Junction Leakage“, „Subthreshold Leakage“ und „Tunneleffect in and through Gate Oxid“ die dominierenden Mechanismen und verursachen die größten Leckströme innerhalb des Transistors. Wobei der klar überwiegende Faktor der Subthreshold Leakage ist und als nächstens in den Vordergrund gestellt und kurz eingeführt wird. Der Subthreshold Leakage, oder auch schwacher Inversionsstrom genannt, tritt immer dann auf, wenn die Gate-Source Spannung unterhalb der Schwellenspannung V_{th} ist. In diesem Arbeitsbereich des Transistors befindet sich nur eine geringe Anzahl an freien Ladungsträgern in dem Bereich zwischen Drain und Source, da der Kanal nicht völlig ausgeschaltet werden kann. Diese geringe Anzahl an Ladungsträgern bewegt sich durch Diffusion anstatt durch die Gate-Source Spannung hervorgerufene Driftbewegung. Dieser Arbeitsbereich wird auch schwache Inversion genannt. In diesem Bereich ist der Subthreshold Leakage exponentiell von der Schwellenspannung, der Gate-Source Spannung und Temperatur abhängig und kann mit der folgenden Formel näherungsweise dargestellt werden:

$$I_D \sim e^{\frac{q(VGS-VTH)}{k_B * T}}$$

Wenn der Fall einer konstanten Gate-Source Spannung ausgegangen wird, kann durch Erhöhung der Schwellenspannung dieser Art des Leckstroms reduziert werden.

Die weiteren Leakage Arten sind zum einen der pn-Junction Leakage, der durch den Sperrstrom des pn-Übergangs zum Substrat verursacht wird. Zum anderen der Gate Tunneleffect in and through Gate Oxid, der sich durch eine Ladungsübertragung durch ein sehr dünnes Gate-Oxid ergibt. Die Reduzierung der Gate-Oxiddicke führt zu einer Zunahme des Feldes über dem Oxid. Durch dieses stark ausgeprägte Feld und der dünnen Oxid-Schicht unter dem Gate kommt es zu einem Tunneln der Elektronen vom Substrat zum Gate und auch vom Gate zum Substrat. [44, 28]

Die drei übrigen Mechanismen verursachen sehr kleine Ströme und können daher in dieser Betrachtung vernachlässigt werden. Weitere Informationen und detaillierte Analysen zu allen dargestellten Arten des Leakage in einem MOS-Transistor sind der Quelle [28] zu entnehmen.

2.3 Dynamischer Leakage

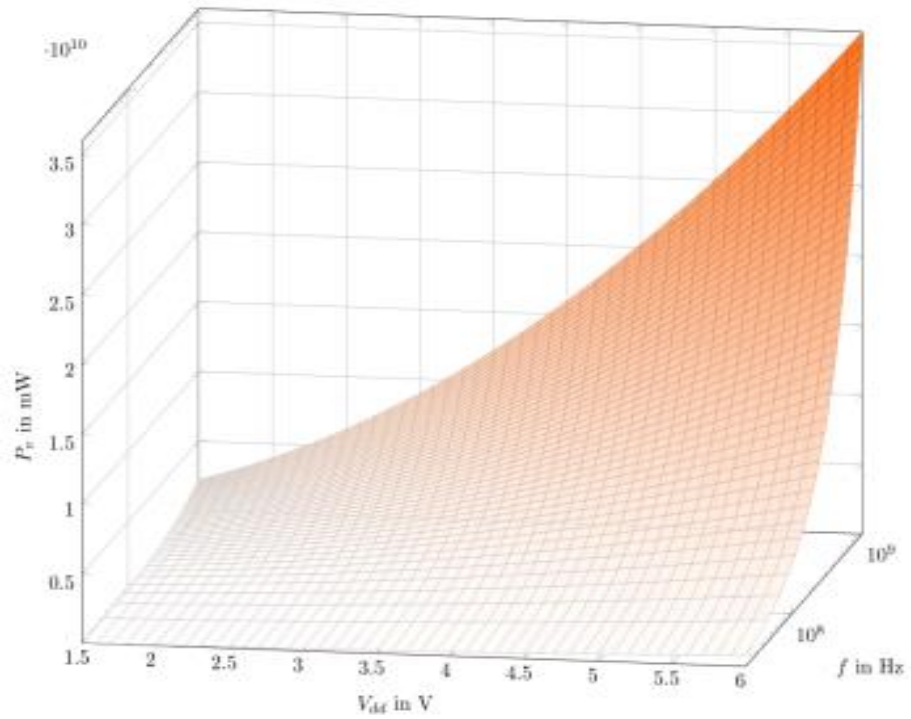


Abbildung 7: Verlustleistung über der steigenden Frequenz und identischem Vdd (z.B. 5V)

Dynamischer Leakage kann durch Hardware Mechanismen und Methoden beeinflusst werden, mit dem Ziel die Leistungsaufnahmen zu reduzieren. Optimierte Software Implementierungen stellen Anfragen an den Prozessor zur Ausführung eines Datensatzes nur zu bestimmten Zeitpunkten, wodurch der Prozessor sich die restliche Zeit im Sleep-Modus befindet. Wenn sich der Prozessor im Sleep-Modus befindet, verbraucht er im Vergleich zur Volllast nur sehr wenig Strom. Dadurch ergibt sich eine entsprechende Reduzierung des dynamischen Leakage (Abbildung 7). Dieser Mechanismus, wird heutzutage direkt auf dem Chip in Form einer Logik implementiert. Diese Logik registriert die Auslastung und schaltet den Prozessor nach Bedarf ab. Auf diese Weise kann zwar der dynamische Leakage reduziert werden, aber es entstehen neue Probleme bei einem dramatischen Anstieg der benötigten Rechenleistung. Wenn zur selben Zeit, alle Kerne ihre volle Leistung abrufen, werden Spannungseinbrüche hervorgerufen [29]. Diese Spannungsdrops müssen abgefangen werden, damit der Prozessor nicht den Spannungsbereich, in dem er zuverlässig arbeitet, verlässt. Verlässt er den zulässigen Spannungsbereich, können ungewollte Reset's ausgelöst werden und Instruktionen verloren gehen oder fehlerhaft ausgeführt werden. Daher wird ein Verfahren eingesetzt, das die Spannungseinbrüche abfangen soll. Diese Methode wird logisch auf dem Chip implementiert und

überwacht die VDDC Spannung. Bei einer drastischen Laständerung wird für die Zeit des Spannungseinbruchs die Frequenz reduziert bis sich die Spannung wieder einpendelt (Abbildung 7).

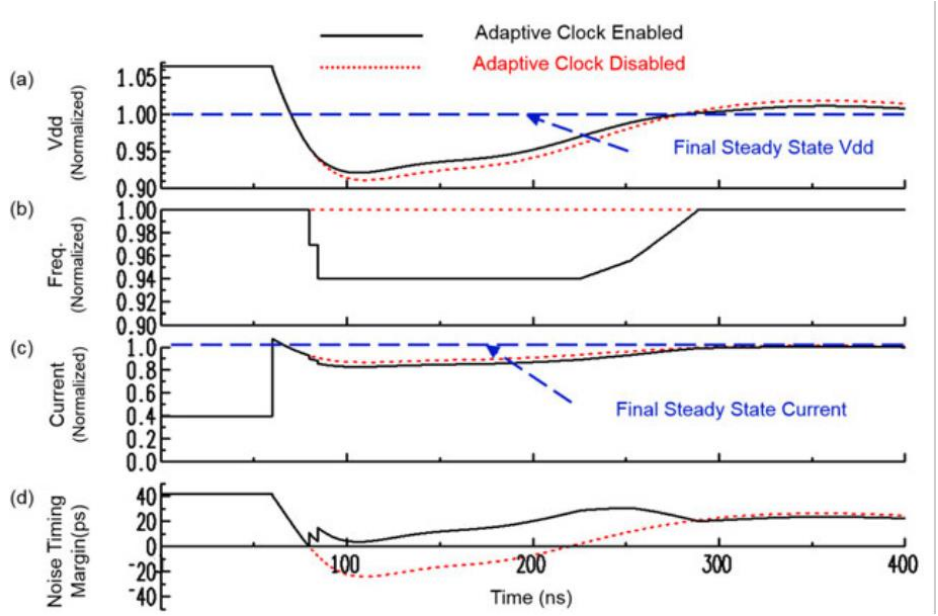


Abbildung 8: Funktionsbeschreibung von Adaptive Clock Skewing. [7]

Es soll dadurch sichergestellt werden, dass in diesem Zeitraum keine Instruktion ausgeführt werden, welche vom derzeitigen Zustand fehlerhaft sein können. Nach dem Abfangen des Spannungseinbruchs wird die Frequenz wieder erhöht und anhand der Last gesteuert. Diese Technik wird Adaptives Clock Skewing genannt und kann parallel zur Dynamischen Frequenz Skalierung (DFS, Dynmamic Frequency Scaling) [30] eingesetzt werden. Anders als beim DFS, wird beim Adaptiven Clock Skewing die Versorgungsspannung überprüft, um Spannungseinbrüche zu erkennen und abzufangen, und so kurzzeitig eine drastische Verringerung der Taktfrequenz hervorzurufen. Hingegen wird bei DFS der aktuelle Lastzustand ermittelt und die Frequenz so beeinflusst, dass der Prozessor die nötige Geschwindigkeit vorweisen kann, um seine Aufgaben schnell genug („just in time“) zu verarbeiten. Dadurch kann im normalen Betrieb, wenn der Prozessor nicht die volle Leistung benötigt, Strom eingespart werden. Diese Technik ist in allen modernen Prozessoren verbaut und wird oft mit der Steuerung der Core-Spannung VDDC kombiniert. Durch eine geringere VDDC Spannung, verbraucht jeder einzelne Transistor weniger Strom. Diese Technik nennt sich Dynamic Voltage and Frequency Scaling (DVFS) [31, 32]. Die im Kontext dieser Arbeit untersuchten Methoden, sind das Reverse und Forward Body Biasing (RBB/FBB) mit Voltage Scaling [33]. Diese Methoden verwenden den Body-Effekt des Transistors, um die Schwellenspannung der Transistoren anzupassen. Hierbei wird das Bulk-Potential und die Core-Spannung so zueinander gesteuert, dass je nach Bedarf eine Leakage Reduktion (RBB) oder eine Steigerung der Schaltgeschwindigkeit (FBB) hervorgerufen wird.

Diese Mechanismen zur Reduktion des Stromverbrauches werden in den nächsten Abschnitten näher betrachtet, wobei das zuletzt genannte Feature RBB/FBB in den Vordergrund gestellt und in Kapitel 5 ausführlich erklärt wird.

3 Der Transistor

Um die Funktion von Reverse und Forward Body Biasing nachvollziehen zu können ist das Verständnis der Funktionsweise eines MOS Transistors nötig. Aus diesem Grund wird im Folgenden der innere Aufbau des MOSFET Transistor, der Unterschied zwischen NMOS und PMOS, der Einfluss der PN-Übergänge und der Body-Effekt im Detail erklärt.

3.1 Historische Entwicklung

Die historische Betrachtung zeigt zwei parallellaufende Zweige in der Transistorentwicklung. Zum einen wird in den 1920er Jahren durch Julian Edgar Lillienfeld die Idee eines Verstärkenden oder Stromschaltenden Bauelements auf Basis von Halbleitern vorgestellt. Er beschreibt ein Bauteil, dass durch die Anwesenheit eines elektrischen Feldes die Leitfähigkeit eines Materials beeinflussen kann. Dieser vorgestellte physikalische Effekt kann auf den später entdeckten „Feldeffekt“ zurückgeführt werden. Aus diesen Forschungen wurde später der Feldeffekttransistor entwickelt. Am 23. Dezember 1947 konnten u.a. W. Shockley, W.Brattain und J.Bardeen in den Bell Labs durch einen Test mit einem Spitzentransistor am Oszillator eine Spannungsverstärkung von 15V nachweisen. Für diese und weitere Entdeckungen in diesem Gebiet erhielten W. Shockley, W.Brattain und J.Bardeen später den Nobelpreis. [34]

Zum anderen wurde von Gordon Teal und Morgan Sparks in den 1950er Jahren ein wichtiger Fortschritt in der Produktion von Bipolartransistoren gemacht. Durch die Einführung eines Fabrikationsprozesses für Flächentransistoren wurde ein PN-Übergang aus einem gewachsenen Kristall gefertigt. Im Jahr 1954 kamen die ersten Bipolartransistoren aus Silizium auf den Markt. [34]

In den nächsten Jahrzehnten spielten die Feldeffekttransistoren auf Grund von Produktionsschwierigkeiten kaum eine Rolle. Heutzutage sind MOS-FET Transistoren die dominante Technologie, während Bipolartransistoren nur noch in speziellen Nischenprodukten eingesetzt werden.

3.2 Funktion des pn-Übergangs

Die Funktion beider Transistortypen basiert auf dem Prinzip zweier zusammenhängender pn-Übergänge. Der pn-Übergang hat eine gleichrichtende Wirkung und lässt das Halbleitermaterial nur bei bestimmten äußeren Spannungsbedingungen leitend werden.

Ein Halbleiter ist ein Material mit vier Valenzelektronen auf der äußeren Hülle des Atoms. Diese Eigenschaft weisen Silizium und Germanium auf. Mit Hilfe spezieller Techniken kann aus diesem Materialen ein nahezu perfekter Kristall gezüchtet werden. Durch die Anordnung der Atome im Kristall gehen die diskreten Energieniveaus der Elektronen in Energiebänder über. Die obersten beiden Energiebänder werden Valenzband und Leitungsband genannt. Zwischen ihnen liegt eine Bandlücke, wobei die umliegenden Atome sich alle Valenzelektronen teilen und eine Elektronenpaarbindung miteinander eingehen (Abbildung 9). [35]

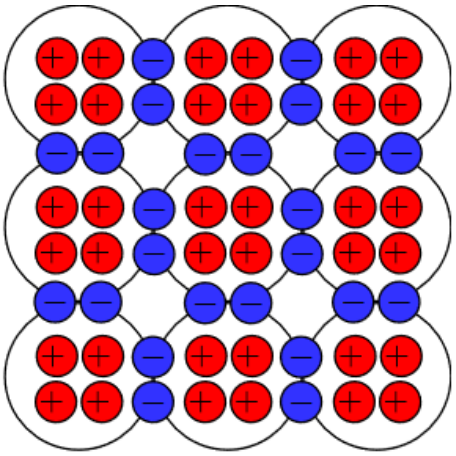


Abbildung 9: Gebildete Kristallgitterstruktur, Quelle: www.onmyphd.com

Alle an den Atombindungen beteiligten Elektronen sind innerhalb des Gitters gepaart, weswegen man auch von Elektronenpaarbindungen spricht. Durch diese Paarung teilen sich benachbarte Atome jeweils zwei Elektronen, wodurch sie die Edelgaskonfiguration mit acht Elektronen auf der äußeren Schale erreichen. Dadurch sind sie elektrisch neutral, da jede negative Ladung eines Elektrons durch eine entsprechende positive Ladung in den Atomrümpfen ausgeglichen wird. Mit zunehmender Temperatur können sich Elektronen vom Atomverband lösen und sich frei im Kristall bewegen. In der Terminologie des Bändermodells erhalten Elektronen durch thermische Bewegung so viel Energie, dass sie die Bandlücke überwinden und vom Valenzband ins Leitungsband übergehen können. Dieser Effekt wird als intrinsische Leitfähigkeit bezeichnet. Die intrinsisch generierte Ladungsträgerdichte ist jedoch relativ gering und sehr stark temperaturabhängig.

Um dem Halbleiter eine hohe Leitfähigkeit zu verleihen, werden Fremdatome in das Kristallgitter eingebracht, welche entweder fünf oder drei Valenzelektronen besitzen. Dadurch können freie positive (p-Typ) oder negative (n-Typ) Ladungsträger in dem Kristallgitter erzeugt werden. Die Implantierung von Fremdatomen wird dotieren genannt. Üblicherweise werden Bor und Phosphor als Dotierstoffe verwendet.

Phosphor ist ein Atom mit fünf Valenzelektronen und erzeugt ein freies Elektron in der Kristallgitterstruktur. Die anderen vier Valenzelektronen gehen eine Elektronenpaarbindung mit den umliegenden vier Valenzelektronen von Silizium ein (siehe Abbildung 10). Bor besitzt drei Valenzelektronen und lässt einen freien Platz in der Kristallgitterstruktur zurück. Dieser Zustand des fehlenden Elektrons in der Elektronenpaarbindung im Kristallgitter wird als Loch oder Defektelektron

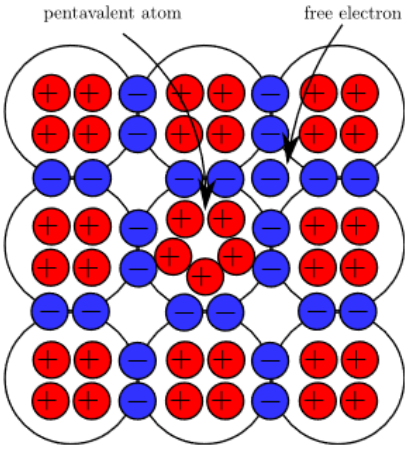


Abbildung 11: N-Dotierung Kristallgitter Quelle: www.onmyphd.com

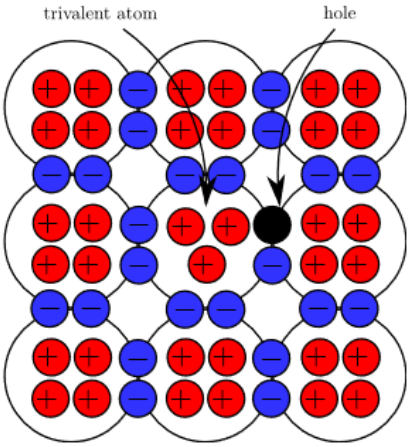


Abbildung 10: P-Dotierung Kristallgitter Quelle: www.onmyphd.com

bezeichnet. Das Loch wirkt auf Elektronen, die in benachbarten Elektronenpaarbindungen stecken, anziehend. Wenn eine Bindung aufreißt und das Elektron sich in das Loch begibt entspricht die aufgebrochene Bindung der neuen Position des Lochs (siehe Abbildung 9).

Wird ein p- und n-dotiertes Material miteinander verbunden, entsteht ein pn-Übergang. Die Elektronen diffundieren aus dem n-dotierten Bereich in das p-Gebiet und rekombinieren dort mit den freien Löchern. Ebenso diffundieren Löcher aus dem p-dotierten Bereich in das p-Gebiet. In der Umgebung des pn-Übergangs entsteht eine Raumladungszone, welche frei von beweglichen

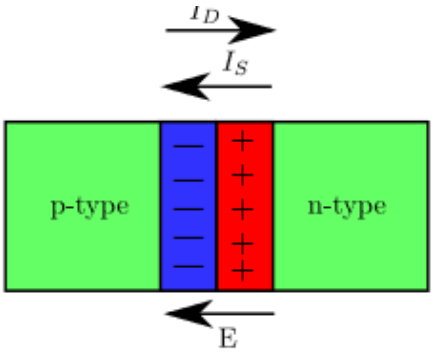


Abbildung 12: PN-Übergang mit dem Diffusionsstrom I_D und Driftstrom I_S, Quelle: www.onmyphd.com

Ladungsträgern ist. Diese Raumladungszone wird auch Verarmungszone oder Sperrschicht genannt. In der n-Zone hinterlassen die abgewanderten Elektronen eine positive ortsgebundene Ladung, während in der p-Zone eine negative ortsgebundene Ladung zurückbleibt (Abbildung 11).

Die Breite der positiven und negativen Gebiete wird durch die Dotierungsdichte bestimmt und kann dementsprechend unterschiedlich sein. Durch die Ladung mit unterschiedlicher Polarität stellt sich ein elektrisches Feld in der Raumladungszone ein, welches wiederum eine Potentialdifferenz zur Folge hat, die Built-In Potential genannt wird. Das elektrische Feld ist so gerichtet, dass es der Diffusionsrichtung der freien Ladungsträger entgegenwirkt. Elektronen werden zurück in das n-Gebiet und Löcher in das p-Gebiet gestoßen. Dieser Ladungsträgerfluss wird Driftstrom I_S genannt. Das Gleichgewicht aus dem Diffusions- und Driftstrom bestimmt die Breite der Raumladungszone. Es gibt zwei Möglichkeiten eine Spannung an den pn-Übergang anzulegen (siehe Abbildung 12).

Reverse Biased:

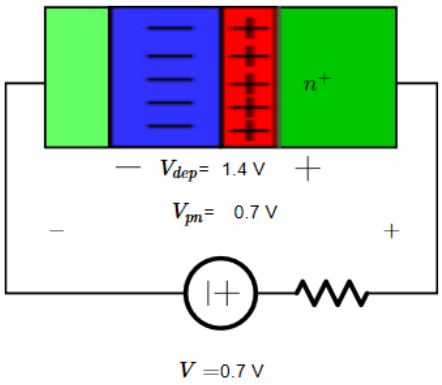


Abbildung 13: PN-Übergang Reverse Biased, RLZ wird größer, Übergang nichtleitend, Quelle: www.onmyphd.com

In diesem Fall wird eine positive Spannung zwischen n und p angelegt. Dadurch verstärkt sich das elektrische Feld, welches der Ladungsträgerdiffusion entgegenwirkt. Es fließt nur ein kleiner Strom, der durch thermisch generierte Minoritätsladungsträger verursacht wird. Elektronen, welche aus dem P-Gebiet ins N-Gebiet abwandern und Löcher, welche aus dem N-Gebiet ins P-Gebiet übergehen. Durch eine steigende Spannung wird die Raumladungszone größer, die Leitfähigkeit nimmt ab und der Widerstand nimmt zu. Der pn-Übergang ist Reverse Biased und nichtleitend (siehe Abbildung 13).

Forward Biased:

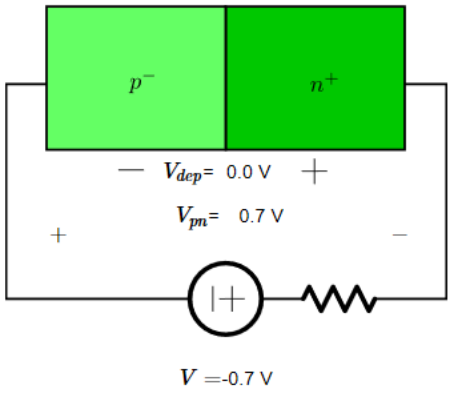


Abbildung 14: PN-Übergang Forward Biased, RLZ wird kleiner, pn-Übergang leitend, Quelle: www.onmyphd.com

In diesem Fall wird eine positive Spannung zwischen p- und n-Gebiet angelegt. Dadurch fließt ein Strom in Richtung des Diffusionsstroms I_D und der Übergang ist Forward Biased. Durch das angelegte Potential wird das n-Gebiet mit Elektronen und das p-Gebiet mit Löchern geflutet. Wenn die Spannung groß genug ist, verschwindet die Raumladungszone (siehe Abbildung 14). Die Majoritätsträger driften in das jeweils andere Dotierungsgebiet und das Silizium wird dadurch leitfähig und weist einen geringen Widerstand auf.

Der pn-Übergang wird in der Elektrotechnik als Diode bezeichnet.

$$V_{pn} < 0 \quad \text{RLZ wird größer, nichtleitend}$$
$$V_{pn} > V_{BI} \quad \text{RLZ verschwindet, leitend}$$
$$V_{BI} = 0{,}7V$$

3.3 Aufbau und Funktion des MOSFET-Transistors

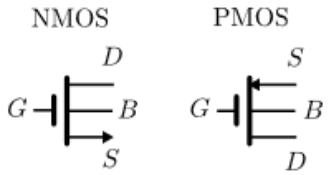


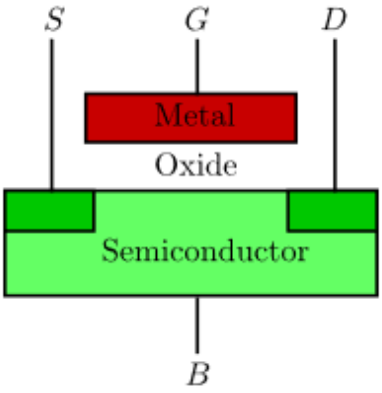
Abbildung 15: Symbole für die Schaltungsentwicklung für PMOS und NMOS Transistor,

Quelle: www.onmyphd.com

Aus dem Namen des MOSFET-Transistors kann zum einen der mechanische Schichtaufbau und zum anderen das Funktionsprinzip entnommen werden. MOS beschreibt die Reihenfolge der jeweiligen Schicht des Transistors, Metal-Oxid-Semiconductor. Während der Namensteil FET (Field Effect

Transistor) sich auf das Funktionsprinzip bezieht. Der MOSFET wird durch ein elektrisches Feld gesteuert, das durch einen Potentialunterschied zwischen Gate und Source hervorgerufen wird. Somit steuert die Spannung am Gate des MOSFET-Transistors den Stromfluss zwischen Drain und Source. Es gibt zwei unterschiedliche Typen von MOS-Transistoren, den NMOS und den PMOS. In Abbildung 15 sind die Schaltzeichen für NMOS und PMOS gezeigt.

Physikalischer Aufbau:



s

Abbildung 16: Physikalischer Aufbau MOSFET, [Quelle: www.OnmyPHD.com]

Der Aufbau des MOS-Transistors ist in Abbildung 16 dargestellt und zeigt, dass das Gate an einen Metall-Kontakt angeschlossen ist. Zwischen Substrat und Metall befindet sich eine Oxidschicht, die den Halbleiter von der Gate-Kontaktierung trennt. Source und Drain werden an zum Substrat invers dotierte Kontakte angeschlossen während der Bulk oder Body Anschluss in vielen Fällen auf das Source Potential gelegt wird. Zeitweise wurde der Gate-Anschluss durch Polysilizium statt durch Metall implementiert. In modernen CMOS Prozessen wird jedoch aus Gründen der besseren Leitfähigkeit wieder Metall als Kontakt eingesetzt. [35, 36]

Funktion MOSFET:

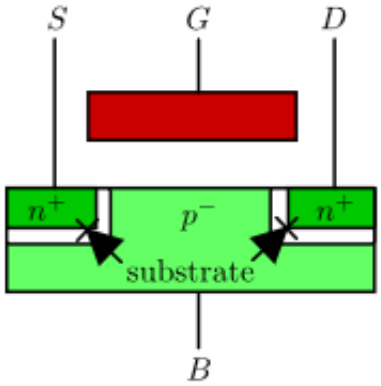


Abbildung 17: Physikalischer Aufbau NMOS, Stark n-Dotierte Source und Drain und schwach dotiertem Substrat

Die Funktion wird am Beispiel eines NMOS Transistors veranschaulicht. Der Aufbau in Abbildung 17 zeigt, dass Source und Drain stark n-dotierte Bereiche sind, die durch das leicht p-dotierte Substrat getrennt sind. Es entstehen dadurch zwei pn-Übergänge. An der Source entsteht die Source-Bulk und am Drain die Drain-Bulk Diode. Diese funktionieren wie die in Abschnitt 4.2 beschriebenen pn-Übergänge. Es entsteht durch Diffusion der Elektronen und Löcher eine nichtleitende Raumladungszone zwischen den stark n-dotierten und dem leicht p-dotierten Substrat. Die beiden Kontakte sind voneinander elektrisch isoliert. Um das Verhalten des MOSFETs zu verstehen, können drei unterschiedliche Betriebsmodi unterschieden werden. [35]

Fall 1) Die Spannung zwischen Gate und Source ist größer als Null ($V_{GS} > 0$) :

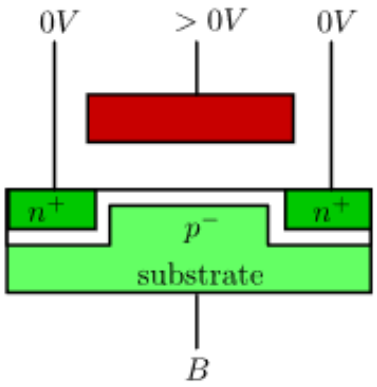


Abbildung 18: Entstandene RLZ unter dem Kanal bei $V_{GS} > 0V$

Wenn die Spannung am Gate erhöht und größer als die Spannung an der Source wird, entsteht aufgrund des Potentialunterschiedes ein elektrisches Feld. Dieses Feld, welches in Abbildung 18 dargestellt ist, schiebt zuerst die Löcher von der Oberfläche des Kanals im Substrat weg und zieht daraufhin thermisch generierte Elektronen oder Elektronen aus den stark n-dotierten Gebieten vom Source und Drain Bereichen näher zum Gate. Wenn die Spannung steigt sammeln sich immer mehr Elektronen unter dem Gate an. Dadurch sinkt der Widerstand bzw. die Leitfähigkeit in diesem Gebiet steigt an. Die Gate-Spannung, bei der sich ein leitender Kanal unter dem Gate ausbildet, wird als

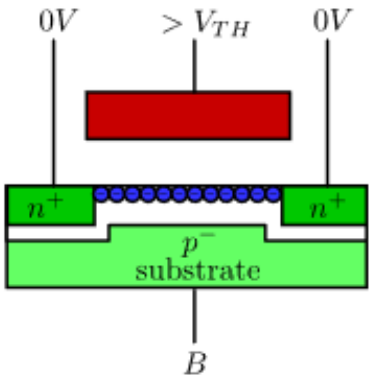


Abbildung 19: Entstandener Kanal unter dem Gate bei $V_{GS} > V_{TH}$

Schwellenspannung (V_{TH}) des Transistors bezeichnet. Bei dieser Gate-Spannung wird das leicht dotierte p-Gebiet zu einem stark dotierten n-Gebiet invertiert (siehe Abbildung 19). [35]

Fall 2) Die Spannung zwischen Drain und Source ist größer als Null ($V_{DS} > 0$) :

Wenn die Spannung V_{DS} den Wert null aufweist (V_{DS} = 0, Abbildung 20), findet keine Elektronenbewegung zwischen Source und Drain statt. Ohne angelegte Drain-Source Spannung ist nur eine ungerichtete thermische Bewegung der Ladungsträger vorhanden. Ist der Kanal invertiert und wird zusätzlich eine Drain-Source Spannung V_{DS} angelegt, setzt ein Stromfluss zwischen Drain und Source ein. Wird die Gate-Source Spannung größer, häufen sich noch mehr Elektronen unter dem Gate an und der Kanal wird leitfähiger und verursacht dadurch eine Zunahme des Stromflusses (siehe Abbildung 19). Durch diesen Zusammenhang ist der Stromfluss bei einem MOSFET proportional zur Gate-Source Spannung. [35]

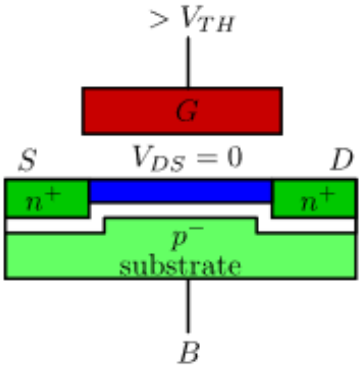


Abbildung 20: Transistor im Betriebszustand V_{DS} = 0V

Fall 3) Die Spannung zwischen Drain und Source ist größer als die Differenz zwischen ($V_{DS} > V_{GS}$-V_{TH}) der Gate-Source- und Schwellenspannung:

Die Leitfähigkeit des Kanals wird von der Stärke des elektrischen Feldes gesteuert, welches von der Potentialdifferenz zwischen Gate und Source abhängt. Da die Potentialdifferenz zwischen Gate und Kanal ausschlaggebend für die Stärke des elektrischen Feldes ist, ist der Kanal in der Nähe der Source stärker ausgebildet als in der Nähe des Drains (siehe Abbildung 21). Wenn die Spannung V_{DS} steigt, nähert sich die Drain Spannung der Gate Spannung an, sodass die Ladungsträgerdichte in der Nähe des Drains unter dem Gate kleiner wird. Dies führt zu einer geringeren Leitfähigkeit und steigendem Widerstand im Inversionslayer. Wenn die Spannung zwischen Drain und Source soweit ansteigt, dass $V_{DS} = V_{GS}$-V_{TH} entspricht, erreicht der Inversionslayer nicht mehr den Drain-Kontakt und der Kanal ist nicht mehr komplett ausgebildet. Der Transistor leitet den Strom, aber die Abhängigkeit von der Drain-Source Spannung wird reduziert. Der Transistor ist dann gesättigt. [35] [36]

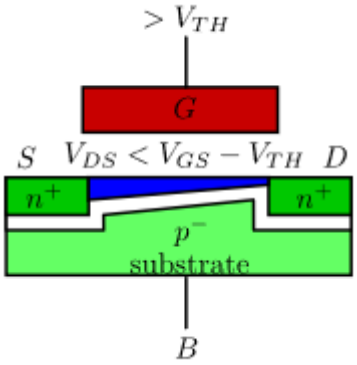


Abbildung 21: PN-Übergang Reversed Biased, RLZ wird größer, Übergang nicht leitend.

Dieser Effekt wird auch als Pinched-Off-Effekt bezeichnet (siehe Abbildung 22). Die Kanalausbildung hängt nur von der Gate-Source Spannung ab und muss die Bedingung $V_{GS} > V_{DS} + V_{TH}$ erfüllen. In diesem Fall ist der Transistor in Sättigung. [35] [36]

$$V_{DS} < V_{GS} - V_{TH} \; - \text{Kanal ausgebildet, linearer Bereich}$$
$$V_{DS} = V_{GS} - V_{TH} - \text{Kanal ausgebildet, Abschnürrpunkt}$$
$$V_{DS} > V_{GS} - V_{TH} \; - \text{Kanal ausgebildt, Sättigung}$$

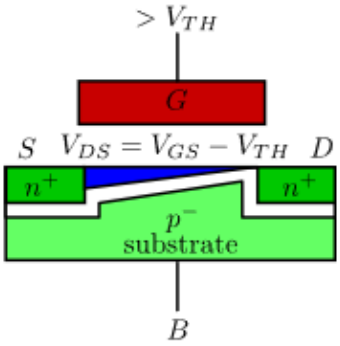


Abbildung 22: Pinched-Off Effekt. Kanal erreicht nicht den Drain, kein Stromfluss in diesem Zustand [14]

3.4 Transistorkennlinien MOS-Transistor

Die Transistorkennlinien geben Aufschluss über die unterschiedlichen Betriebsmodi und das Verhaltensmodell des MOSFETs. Von besonderer Bedeutung sind hierbei die Eingangskennlinie $I_{DS} = f(V_{GS})$ und die Ausgangskennlinie $I_{DS} = f(V_{DS})$. Die Betriebsbereiche werden je nach der entsprechenden Anforderung unterschieden. Abhängig von den Bias-Bedingungen der Transistoren gibt es unterschiedliche Zusammenhänge zwischen Spannung und Strom am Transistor. Bezogen auf die Ausgangskennlinie des Transistors unterscheidet man zwischen zwei Arbeitsbereichen, nämlich dem Trioden- und dem Sättigungsbereich. Der Strom steigt in jedem Arbeitsbereich mit steigender Gate-Source Spannung. Der Strom im Trioden-Bereich kann mit Formel (1) näherungsweise errechnet werden.

$$I_{DS} = \mu C_{ox} \frac{W}{L}((V_{GS} - V_{TH})V_{DS} - \frac{1}{2}V_{DS}^2) \ (1)$$

Wobei μ (μ_n oder μ_p) die Ladungsträgerbeweglichkeit, C_{OX} die Oxidkapazität und W/L das Breiten Längenverhältnis des Transistors darstellt. Wie Formel 1 zu entnehmen ist, steigt im Triodenbereich der Strom mit der Gate-Source Spannung V_{GS}. Darüber hinaus hat der Strom eine quadratische Abhängigkeit von der Drain-Source Spannung. Auf Grund des negativen Vorzeichens des quadratischen Terms ähnelt die Kennlinie einer nach unten geöffneten Parabel. Für kleine Drain-Source Spannungen ist der Einfluss des quadratischen Terms vernachlässigbar und es stellt sich ein nahezu linearer Zusammenhang zwischen der Drain-Source Spannung und dem Kanalstrom ein. Aus diesem Grund wird der Triodenbereich oft auch linearer Arbeitsbereich genannt. Die Sättigung lässt sich mithilfe von Formel (2) näherungsweise berechnen.

$$I_{DS} = \mu C_{ox} \frac{W}{L}(V_{GS} - V_{TH})^2 \ (2)$$

Im Folgenden werden die Transistorkennlinien visualisiert. Die Eingangskennlinie d.h. der Kanalstrom als Funktion der Gate-Source Spannung wird in Abbildung 22 dargestellt.

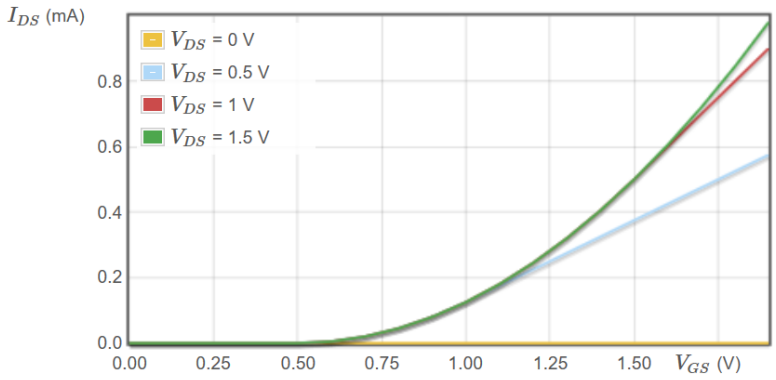


Abbildung 23: Eingangskennlinie Kennlinie für I_{DS} über V_{GS} mit unterschiedlichen V_{DS} Spannung mit Scharrparameter [14]

Die Schwellenspannung dieses Transistors beträgt 0,5V. In Abbildung 23 ist ab dieser Spannung ein rapider Anstieg des Drain Stroms zu erkennen. Dieser Anstieg kann auf die ausgebildete Inversionszone zurückgeführt werden. Außerdem kann durch das Erhöhen der Drain-Source Spannung die Stromaufnahme des Transistors erhöht werden. Im Diagramm ist auch der Graph für eine Drain-Source Spannung von $V_{DS} = 0$ eingezeichnet. Unterhalb der Schwellenspannung befindet sich der Transistor im ausgeschalteten Zustand. Der Übergang zwischen linearen und gesättigten Bereich hängt von der Sättigungsspannung ab. [35] [36]

$$V_{GS} < V_{DS} + V_{TH}, \qquad \text{Saturation} \qquad (3)$$

$$V_{GS} > V_{DS} + V_{TH}, \qquad \text{Triode} \qquad (4)$$

Die zuvor beschriebenen Bedingungen für die verschiedenen Arbeitsbereiche des Transistors treffen natürlich auch auf die Kennlinie des Drain Stroms über die Drain-Source Spannung zu. Es ist zu

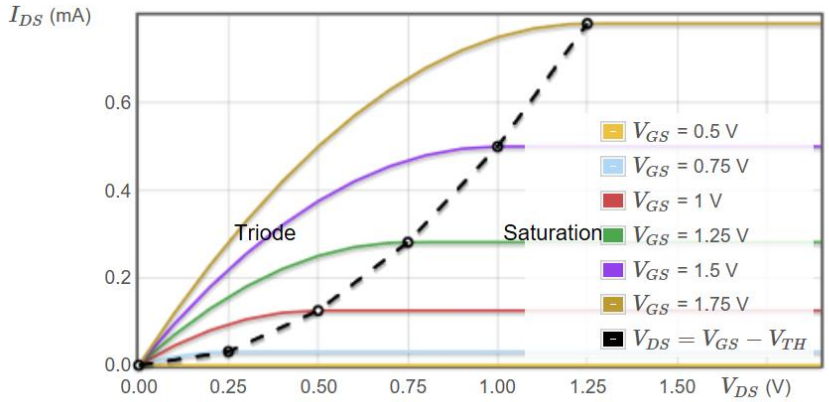


Abbildung 24: Ausgangskennlinie I_{DS} über V_{DS} mit markiertem Trioden- und Saturationsbereich [14]

erkennen, dass solange $V_{GS} < V_{TH}$ gilt, kein Stromfluss zwischen Drain und Source stattfindet (Kennlinie V_{GS}=0,5V; siehe Abbildung 24). Es lässt sich außerdem der Trioden Bereich von dem Bereich der Saturation des Transistors gut unterscheiden. Der Strom steigt stark mit der Drain-Source Spannung an, solange der Transistor sich im Trioden Bereich befindet. Wenn die Saturation des Transistors erreicht wurde, erreicht auch der Strom sein Maximum. Die Höhe des Drain Stroms hängt somit allein von der Gate-Source Spannung ab. [35] [36]

3.5 Unterschied PMOS und NMOS

In diesem Abschnitt werden die Fabrikationsunterscheide zwischen PMOS und NMOS Transistor erläutert. Die beiden Transistortypen unterscheiden sich im physikalischen Aufbau und in der Berechnungsweise der Schwellenspannung.

NMOS:

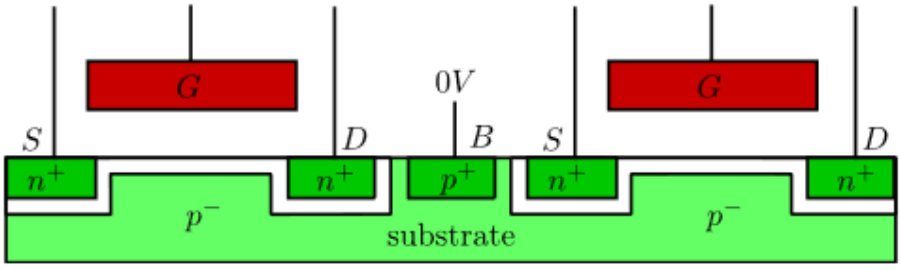


Abbildung 25: Physikalischer Aufbau NMOS Transistor mit dem Shared-Bulk Konzept [36]

Der NMOS teilt sich in dieser Fertigungsform einen Bulk Anschluss mit den umliegenden Transistoren. Es gibt aber auch NMOS Typen mit isolierten p-Wannen. Dabei ist die p-Wanne des betreffenden Transistors von einer tiefen n-Wanne umgeben, welche die p-Wanne vom globalen p-Substrat isoliert. In Abbildung 25 ist die Bulk Elektrode, welche zwischen den beiden Transistoren liegt, zu erkennen.

Beim NMOS muss das Bulk-Potential kleiner oder gleich dem Source Potential sein. Das kann dadurch erreicht werden, dass Bulks mit dem niedrigsten Potential der Schaltung verbunden werden, welches in den meisten Fällen 0V entspricht.

Die Schwellenspannung wird durch die nachfolgende Formel berechnet:

$$V_{TH} = V_{TH0n} + \gamma\left(\sqrt{(2\phi} + V_{SB}\right) - \sqrt{(2\phi})$$

PMOS:

Der PMOS ist im Gegensatz zum NMOS etwas komplexer aufgebaut. Da sich das Grundsubstrat nicht ändert und weiterhin p-Dotiert ist, müssen sogenannte „wells“ bzw. Wannen in das Substrat implantiert werden. Diese Nwells weisen eine schwache n-Dotierung auf, wodurch ein pn-Übergang zum Substrat mit entsprechender Isolation entsteht. Beim PMOS wird das Nwell sehr häufig mit dem Source-Kontakt kurzgeschlossen (siehe Abbildung 26). Bei der späteren Charakterisierung von RBB und FBB wird anstatt des Kurzschlusses eine Spannungsquelle verwendet, um eine Potentialdifferenz zwischen Source und Bulk einzuführen. [36]

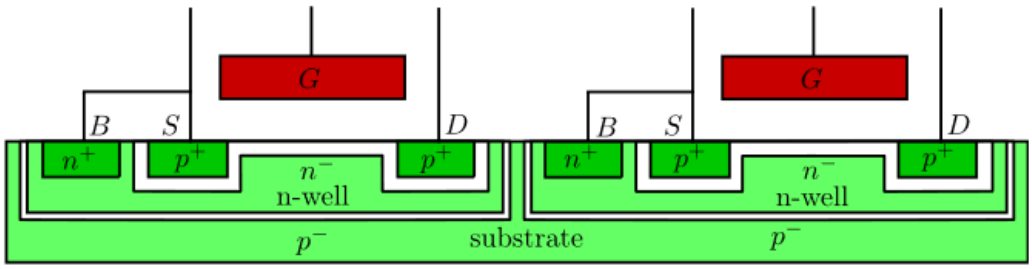


Abbildung 26: Physikalischer Aufbau PMOS Transistor [36]

Mit der nachfolgenden Formel wird die Schwellenspannung für einen PMOS berechnet.

$$V_{TH} = V_{TH0p} + \gamma\left(\sqrt{(2\phi} + V_{SB}\right) - \sqrt{(2\phi})$$

3.6 Der Body Effekt

In der Funktionsbeschreibung des Transistors wurde bisher der Bulk Kontakt vernachlässigt. Die Funktion des Body-Effekts wird am Beispiel eines NMOS Transistors erklärt. Über den Bulk des Transistors kann eine Veränderung der Schwellenspannung hervorgerufen werden. Bislang wurde davon ausgegangen, dass beim NMOS oder PMOS der Bulk immer mit der Source kurzgeschlossen ist. Über den Bulk-Effekt kann jedoch die Schwellenspannung sowohl vergrößert als auch verringert werden. Dieser sogenannte Body-Effekt wird in diesem Projekt zur Reduktion des statischen und dynamischen Verluststroms genutzt und wird im folgenden Abschnitt erklärt. Ähnlich wie bei einem

pn-Übergang existieren zwei Betriebszustände nämlich das Reverse- und das Forward-Body-Biasing, die beide auf dem Prinzip des Body-Effektes basieren. [33] [35] [36]

3.6.1 Reverse Body Biasing

Sowohl die Namensgebung als auch die Funktionsweise des Reverse-Body-Biasing steht im engen Zusammenhang mit der Funktion eines pn-Übergangs. Die Source-Bulk oder die Drain-Bulk Diode müssen in Sperrrichtung betrieben werden, damit kein Verluststrom über die Diode zum Substrat fließt. Dementsprechend muss an der Source ein gleich großes oder ein größeres Potential gegenüber dem Bulk Potential angelegt werden. Wenn die Source Spannung größer ist als die Bulk Spannung, vergrößert sich die Raumladungszone unterhalb des Kanals, der leitend mit der Source Elektrode verbunden ist. Auf Grund der größeren Ausdehnung der Raumladungszone, erhöht sich die ortsgebundene negative Ladung, welche durch Ionisation der Dotierungsatome in der Raumladungszone entsteht (siehe Abbildung 27). Aufgrund der Ladungssymmetrie muss die zusätzliche negative Ladung durch positive Gate Ladung gespiegelt werden. In Folge stehen weniger positive Gate-Ladungen für die Invertierung des Kanals durch Attraktion von Elektronen zur Verfügung. Damit der Kanal dieselbe Ladungsträgerdichte wie bei einer Source-Bulk Spannung von 0V aufweist, und somit derselbe Strom zwischen Drain und Source fließen kann, muss die Gate-Source Spannung angehoben werden. Die Schwellenspannung steigt, da eine größere Gate-Source Spannung benötigt wird, um dieselbe Leitfähigkeit des Kanals zu erreichen (siehe Abbildung 27). Die Zunahme der

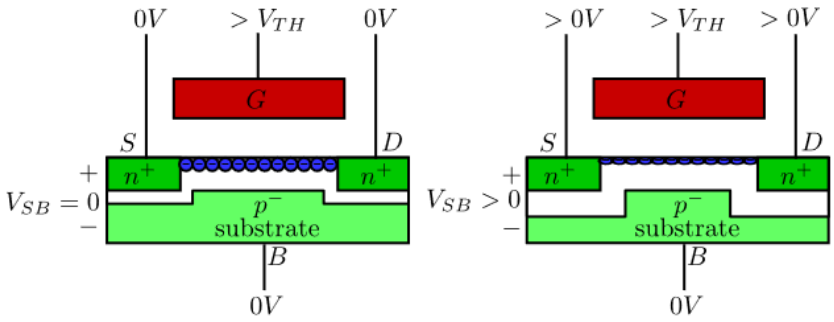


Abbildung 27: Der Body-Effekt mit RBB, Kanal wird kleiner und die RLZ unter dem Gate wird größer. Vergrößerung der Schwellenspannung. [36]

Schwellenspannung wird durch den Bodyfaktor Gamma γ beschrieben. Wird die Gate-Source Spannung bei erhöhter Schwellenspannung nicht angehoben, so führt der Kanal einen geringeren Drain Strom als zuvor.

Es ist festzustellen das bei RBB die Schwellenspannung steigt und der transportierte Drain Strom bei konstanter Gate-Source Spannung sinkt. Für die Implementierung des RBB Features ist nur wenig Schaltungsaufwand notwendig. Der Kurzschluss zwischen den Bulk und Source Elektroden muss durch

eine separate Verdrahtung ersetzt werden. Darüber hinaus wird eine Spannungsquelle benötigt, die bei einem NMOS eine negative Spannung zwischen Source und Bulk einprägt. [33] [35] [36]

Aufgrund der Tatsache, dass RBB die Schwellenspannung erhöht kann der Subthreshold Leakage im Bereich der schwachen Inversion reduziert werden. Dieser ist, wie in Kapitel 2.2 gezeigt, exponentiell von der Schwellenspannung abhängig und kann durch eine Erhöhung der Schwellenspannung reduziert werden. Daher bietet der Einsatz von RBB den Vorteil, dass RBB auch im statischen Betrieb einer CMOS Schaltung die Stand-By Ströme reduziert.

3.6.2 Forward Body Biasing

Beim Forward-Body-Biasing (FBB) wird die Schwellenspannung eines NMOS Transistor verringert, indem der Bulk gegenüber dem Source Potential angehoben wird. Die Raumladungszone wird durch den Potentialunterschied zwischen der Gate-Source Spannung und dem Bulk Potential kleiner bzw. wird komplett abgebaut. Aus diesem Grund sollte es vermieden werden, die Source-Bulk Spannung größer als eine Diodenspannung zu wählen. In diesem Fall würde die Bulk-Diode komplett öffnen und sich ein großer Stromfluss über die Dioden zum Substrat einstellen. In diesem Zustand ist der Transistor nicht funktionsfähig. FBB führt zu einer geringen Schwellenspannung, wodurch auch mehr Strom zwischen Drain und Source fließen kann.

Das Potential vom RBB und FBB werden in Abbildung 29 und Abbildung 28 grafisch dargestellt. Mit

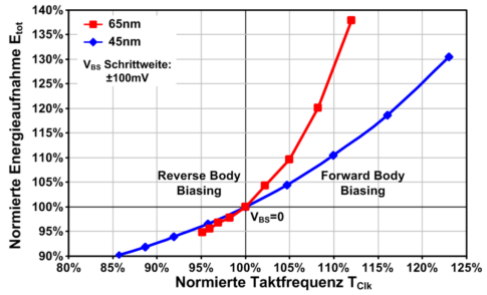


Abbildung 29: Taktfrequenz über Energieaufnahme in Bezug auf das Potential von RBB/FBB [11]

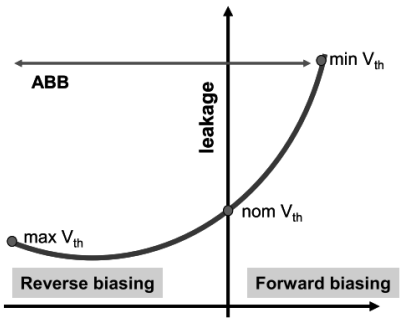


Abbildung 28: Einsparpotential von RBB/FBB mit Spannungsskalierung [11]

FBB kann durch wenig Aufwand eine höhere Schaltgeschwindigkeit im Betrieb eingestellt werden, wobei bei RBB der Leakage im Betrieb reduziert werden kann. Durch die geringe Änderung im Schaltungsaufwand ist das Potential hinter diesen beiden Features enorm, da nur die Bulk-Spannungen verändert werden müssen, um die genannten Vorteile zu nutzen. [33] [35] [36]

4 Power Management Einheiten

In diesem Abschnitt wird die Aufgabe und Funktion von Spannungswandlern erläutert. Dabei werden verschiedene Wandler Architekturen vorgestellt. Grob lassen sich Spannungswandler in zwei Klassen kategorisieren und zwar in Linearspannungsregler und Schaltregler. Die einfachste Form der Linearspannungsregler sind sogenannte Festspannungsregler, welche eine zuvor definierte und nicht veränderbare Ausgangsspannung liefern. Festspannungsregler werden über einen Spannungsteiler eingestellt. Die Switched-Mode Regler unterscheiden sich in Ihrer Funktion dahingehend, dass sie über einen Takt gesteuert und für die Generierung der Ausgangsspannung Energiespeicher verwendet werden.

Aufgrund der Tatsache, dass im Projekt nur die Nutzung von Gleichspannungen vorgesehen ist, werden in diesem Bericht auch ausschließlich nur Gleichspannungswandler erläutert.

4.1 Wandler Architekturen

Die beiden Grundarten der Regler/Wandler Architekturen unterscheiden sich in Ihrer Funktion und ihrem Anforderungsprofil drastisch. Switched-Mode-Regler benutzen Energiespeicher, die mittels eines Taktes auf- und entladen werden, um die Spannung zu generieren. Obwohl sie mit einem Takt gesteuert werden, liefern Schaltspannungsregler eine konstante geregelte Gleichspannung am Ausgang. Bei Linearreglern wird eine skalierte Darstellung der aktuellen Ausgangsspannung zur Justierung mit einer Referenzspannung verglichen und anhand der erkannten Abweichung nachgeregelt. Im Folgenden Abschnitt werden die Topologien der Linearspannungswandler anhand eines Low-Drop-Out-Reglers und die Switched-Mode-Regler anhand einer Ladungspumpe mit positiver und negativer Ausgangsspannung erläutert. In der folgenden Auflistung sind einige Typen der jeweiligen Architekturen aufgelistet

- Linear Regler:
    - Low Drop Out Regler (LDO)
    - Shunt Regler
    - Zener Diode Shunt Regler
- Switched-Mode-Regler:
    - Charge Pump (CP, Ladungspumpe)
    - Buck-Converter (BuC, Abwärtswandler)
    - Boost-Converter (BoC, Aufwärtswandler)
    - Buck-Boost-Converter (BBC, Split-Pi-Wandler)

Bei der Auswahl, des eingesetzten Wandlers, spielt das Anforderungsprofil eine ausschlaggebende Rolle. Da bei einem Linearregler der Ausgangsstrom abhängig vom Eingangsstrom ist, wird dieser Wandler eingesetzt, wenn der Eingangsstrom in etwa dem Ausgangsstrom entsprechen soll.

$$I_{out} \approx I_{in}$$

Switched-Mode-Regler werden eingesetzt, wenn die Eingangsleistung in etwa der Ausgangsleistung entspricht. Dabei wird die Energie aus dem Eingangszweig so gewandelt, dass die passende Ausgangsleistung generiert wird. Dabei kann auch eine höhere Ausgangsspannung bei gleichzeitig kleinerem Ausgangsstrom oder eine kleinere Ausgangsspannung bei gleichzeitig höherem Ausgangsstrom erreicht werden. Gerade der letztere Fall wird of genutzt, um eine Reduktion des Stromflusses auf den Versorgungsleitungen zu erzielen, die mit geringeren ohmschen Verlusten einhergeht und damit zu einer Verbesserung der Versorgungseffizienz führt. In allen besprochenen Wandlungsfällen gilt:

$$P_{out} \approx P_{in}$$

Diese beiden Kriterien sind für die Auswahl der geeigneten Wandler Architektur zu beachten. Der Schaltungsaufwand ist bei Schlatreglern deutlich höher, als bei Linearreglern. Das wird in den nachfolgenden Kapiteln deutlich, da für jede Ausgangsspannung einer Ladungspumpe eine Kaskade aus Transistor und Kondensator benötigt wird. [41]

4.2 Shunt-Regler

Ein Shunt-Regler besteht aus einem NPN oder NMOS Transistor, einem Operationsverstärker, einem Spannungsteiler und einer Referenzspannungsquelle(Abbildung 30). Der Feedback Zweig wird an den positiven Eingang des Operationsverstärkers angelegt. [37]

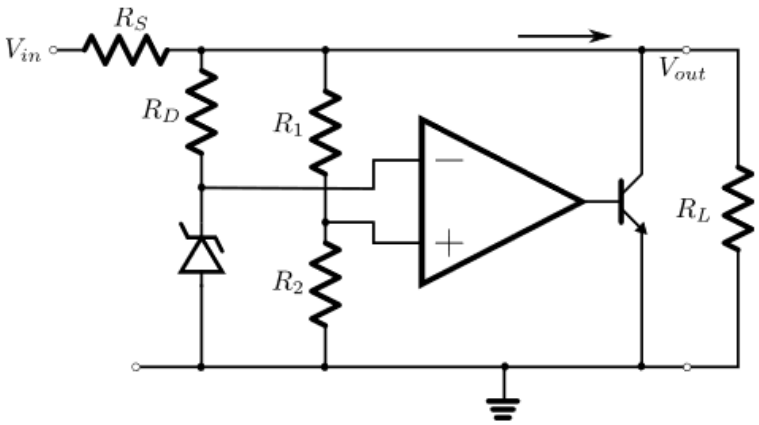


Abbildung 30: Shunt-Regler mit Zener-Diode und PNP Bipolartransistor [5]

Der Spannungsteiler (R_1, R_2) dient zur skalierten Wiedergabe der aktuellen Ausgangsspannung. Der negative Eingang des Operationsverstärkers wird über eine Zener-Diode mit einer Referenzspannung versorgt, welche alternativ über eine Bandgap-Schaltung erzeugt werden kann. Dieser Spannungsteiler muss so eingestellt werden, dass die gewünschte Ausgangsspannung nach der Teilung exakt dem Spannungspegel der Referenzspannung entspricht. Bei einem Operationsverstärker mit negativem Feedback gilt das Prinzip des virtuellen Kurzschlusses. Das System regelt sich so ein, dass Spannung am nichtinvertierenden Eingang exakt der Spannung am invertierenden Eingang entspricht. Die Ausgangsspannung regelt sich dementsprechend so ein, dass die runtergeteilte Spannung am Spannungsteiler genau der Referenzspannung der Zenerdiode entspricht. [41]

Fall 1) Die Spannungsdifferenz am Operationsverstärker steigt:

Dieser Fall tritt bei einer zu hohen Ausgangsspannung ein. Bei diesem Fall steigt die Ausgangsspannung des Operationsverstärkers, wodurch der Transistor mit einer größeren Basis-Emitter-Spannung U_{BE} angesteuert wird. Der Transistor zieht dadurch mehr Strom über den Kollektor und der Spannungsabfall über R_S wird größer. Dadurch sinkt die Ausgangsspannung wieder und wird stabilisiert.

Fall 2) Die Eingangsspannung am Operationsverstärker sinkt:

Dieser Fall tritt bei einer zu niedrigen Ausgangsspannung auf. In Folge sinkt die Ausgangsspannung des Operationsverstärkers, wodurch die Basis-Emitter-Spannung U_{BE} kleiner wird. Dadurch sinkt auch der Kollektorstrom I_C wodurch die Spannung angehoben und stabilisiert wird.

Mit diesem Regler wird jede Störung der Eingangsspannung und unerwartete Laständerung abgefangen und eine stabilisierte Ausgangsspannung gewährleistet. Die Rückkopplungsschleife des Reglers reagiert ausreichend schnell auf die neuen Lastbedingungen.

4.3 Low Drop Out Regler

Der LDO wird eingesetzt, um eine empfindliche Last von einer gestörten Versorgungsspannung zu isolieren. Er gehört zu den Linearreglern und zeichnet sich durch seine geringe Differenz zwischen Ein- und Ausgangsspannung aus. Für eine hohe Versorgungseffizienz muss diese Spannungsdifferenz sehr gering sein und darf in einigen Anwendungsfällen 200 mV nicht überschreiten. Damit aus einem herkömmlichen Linearregler ein LDO wird, muss ein sogenanntes Durchgangselement (Pass-Device) verbaut werden. Dies kann sowohl ein PMOS als auch ein PNP-Transistor sein. Je nach Anwendungsfall wird das passende Durchgangselement ausgewählt. MOS-Transistoren werden gewählt, wenn durch

die Änderung des Kanalstroms die Ausgangsspannung angepasst werden soll. Die Drop-Out Spannung beschreibt die Spannungsdifferenz zwischen Ein- und Ausgangsseite und sollte dementsprechend klein sein. Diese Reglerarchitektur reagiert langsam auf plötzliche Lastwechsel, was in bestimmten Anwendungsfällen ein großer Nachteil sein kann. Das Pass Device befindet sich in Reihe zur Ausgangsspannung, wodurch bei einem plötzlichen Lastwechsel, kurzzeitig kein zusätzlicher Strom über das Pass Device bereitgestellt werden kann. Aus diesem Grund wird parallel zur Last ein Kondensator C_{Out} zur Glättung und Stabilisierung der regulierten Ausgangsspannung geschaltet, um für einen kurzen Moment der Last den fehlenden Strom zu liefern. Dadurch fällt die Ausgangsspannung nur minimal ab, und wird anschließend durch die Regelung wieder nachjustiert. Wenn die Regelung sich wieder eingependelt hat, wird auch der Kondensator wieder aufgeladen.

In Abbildung 31 ist die Schaltung eines LDO mit PMOS Pass-Device gezeigt. Der negative Eingang des Operationsverstärkers ist mit der Referenzspannungsquelle verbunden. Der Ausgang des Operationsverstärkers ist mit dem Gate von M1 verbunden. Der Drain von Transistor M1 an die Versorgungsspannung V_{IN} angeschlossen. Der negative Eingang des Operationsverstärkers A wird mit einer skalierten Spannung, die über den Spannungsteiler R_1 und R_2 erzeugt wird, versorgt. Der Widerstand R_1 ist an den Drain des Transistors M1 angeschlossen.

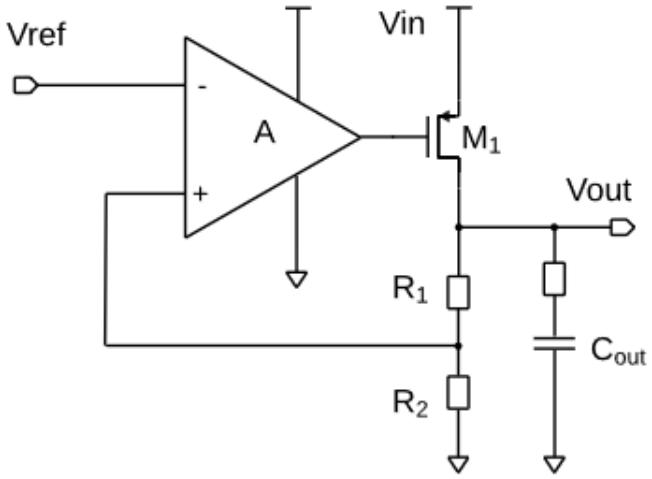


Abbildung 31: LDO mit PMOS Pass-Device

Mit dem Operationsverstärker wird eine Differenz zwischen der eingestellten Spannung am Spannungsteiler und der Referenzspannung gebildet. Schwankt die Ausgangsspannung und das Potential am Drain des Transistors M1 sinkt, entsteht eine Differenz an den Eingängen des Operationsverstärkers. In diesem Fall wird das Gate des Transistors M1 mit einer höheren Spannung angesteuert, wodurch der leitende Kanal entsprechend vergrößert und der Kondensator wieder

geladen wird. Dies geschieht bis der Spannungsabfall ausreguliert ist und das System wieder in den eingeschwungenen Zustand übergeht [37]. Steigt nun die Ausgangsspannung über das eingestellte Potenzial an, so sinkt nahezu gleichzeitig die Spannung am nicht-invertierenden Eingang des Operationsverstärkers. Dieser senkt die Source-Gate Spannung an M1, wodurch der Kanal verkleinert wird und weniger Strom zum laden des Kondensators zur Verfügung steht. Dadurch sinkt die Ausgangsspannung solange, bis der eingestellte Ausgangsspannungswert erreicht wird.

Die Ausgangsspannung lässt sich näherungsweise über die folgende Formel einstellen:

$$V_{OUT} = V_{In} * (1 + \frac{R1}{R2})$$

Ein LDO der mit einem PMOS Pass-Device betrieben hat im Vergleich zu einem Linear Regler mit NMOS Pass Device den Vorteil, dass die abfallende Spannung über den Transistor eine Schwellenspannung höher liegt, als beim NMOS Pass Device. Dadurch erreicht ein LDO mit PMOS als Pass Device einen Spannungsverlust von ca. 200mV. Durch diesen geringeren Spannungsabfall entsteht der Name Low Drop Out Spannungsregler. [43]

4.4 Ladungspumpe

In manchen Anwendungsfällen werden negative oder positive Spannungen benötigt, die nicht zwischen Masse und V_{DD} liegen. Ein Beispiel stellt ein Flash-Speicher dar, der zur Löschung seines gespeicherten Zustands einen negativen Spannungsimpuls und zum Schreiben und Speichern eine sehr große positive Spannung, die oberhalb von V_{DD} liegt, benötigt. Dazu werden sogenannte Ladungspumpen (Charge Pump) eingesetzt. In diesem Abschnitt werden die Ladungspumpen beschrieben und unterschiedliche Schaltungstopologien aufgezeigt.

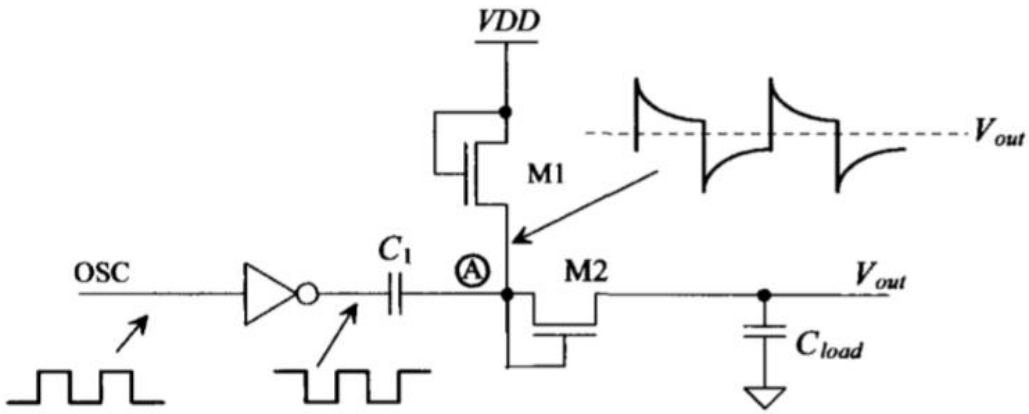


Abbildung 32: Aufbau einer positiven Ladungspumpe[40]

In Abbildung 32 ist ein einfacher Aufbau einer Ladungspumpe dargestellt. Sie besteht aus zwei Kondensatoren, zwei MOSFETs und einem Inverter. Der Inverter ist an ein oszillierendes Steuersignal

angeschlossen und bewirkt, dass C1 ständig geladen und entladen wird. Die Funktion der Ladungspumpe setzt voraus, dass M1 und M2 wie Dioden arbeiten. M1 wird verwendet um den Punkt A auf eine Spannung von V_{DD}-V_{THN} (V_{THN}, Schwellenspannung NMOS) zu ziehen. Dadurch wird C_1 geladen, wenn sich das Oszillatorsignal in einer Low Phase befindet. Zu Beginn des Einschwingvorgangs ist der Kondensator C_{out} noch entladen und die Ausgangsspannung V_{out} sehr niedrig. Aus diesem Grund stellt sich eine Spannungsdifferenz über den Transistor M2 ein, die größer als die Schwellenspannung des Transistors ist. Daher wird zu Beginn des Einschwingvorgangs der Kondensator C_2 ebenfalls aufgeladen, wenn sich das Oszillatorsignal in einer Low-Phase befindet. Später wenn der Kondensator C_2 bereits teilweise aufgeladen ist und die Ausgangsspannung V_{out} steigt, wird die Spannung über M2 in dieser Ansteuerungsphase unterhalb der Schwellenspannung liegen, wodurch sich der Transistor M2 öffnet und der Kondensator C_2 vom Potential des Kondensators C_1 entkoppelt wird.

Geht der Oszillator nun in die High Phase, dann wird die Elektrode des Kondensators, die zuvor auf Masse lag, auf V_{DD} angehoben. Da der Kondensator zuvor auf eine Spannung V_{DD}-V_{THN} aufgeladen wurde, steigt das Potential am Punkt A auf 2*V_{DD}-V_{THN} an. In dieser Phase ist der Transistor M2 leitend und es findet ein Ladungsausgleich zwischen den Kondensatoren C_1 und C_{load} statt. Im eingeschwungenen Zustand lädt sich der Kondensator C_{load} bis 2*(V_{DD}-V_{THN}) auf. Die zusätzliche Schwellenspannung fällt über den Transistor M2 ab. Für den korrekten Betrieb muss sichergestellt werden, dass C_1 um ein Vielfaches größer ist als C_{load}. Die Oszillator Frequenz muss so gewählt werden, dass der Kondensator komplett geladen oder entladen wird, bevor der Oszillator seinen Zustand ändert. In den meisten praktischen Anwendungen sind C_{load} und C_1 nahezu gleich groß und die Ladungspumpe mit einer DC-Last belastet. Idealerweise sollte die Ausgangsspannung konstant sein und der Kondensator am Ausgang nicht vollständig entladen werden. Ein Nachteil, dieser Regler Architektur besteht allerdings in ihrer Zeitabhängigkeit. Eine Ladungspumpe erreicht nicht sofort ihre

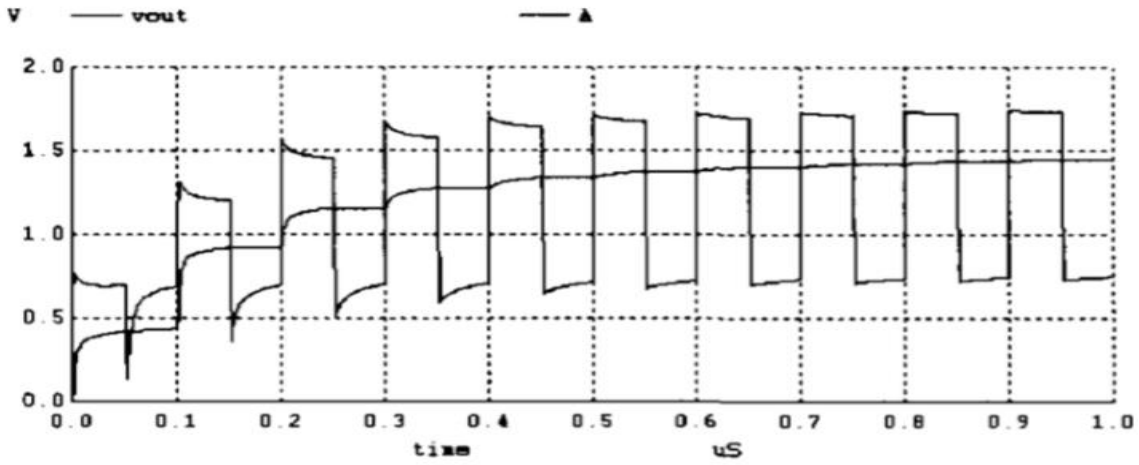


Abbildung 33: Zeitabhängige Ausgangsspannung der Ladungspumpe[4]

eingestellte Ausgangsspannung, sondern benötigt hierfür einige Taktzyklen, was in Abbildung 33 gut zu erkennen ist. Die Welligkeit der Ausgangsspannung ist stark abhängig von der Größe der Ausgangslast. Je größer die Last am Ausgang, desto schneller wird der Kondensator C_{load} entladen und umso welliger wird die erzeugte Ausgangsspannung. Für den gezeigten Simulationsausschnitt (Abbildung 33) liegen folgende Parameter vor:

- NMOS Skalierungsfaktor 10/1
- $C_{load} = C_1 = 1$ pF
- $f_{osc} = 10$ MHz

In diesem Beispiel wurde die Eingangsspannung durch die Ladungspumpe fast verdoppelt.[38] [40]

$$V_{out} = 2 * (V_{DD} - V_{THN})$$

4.5 Ladungspumpe mit negative Ausgangsspannung

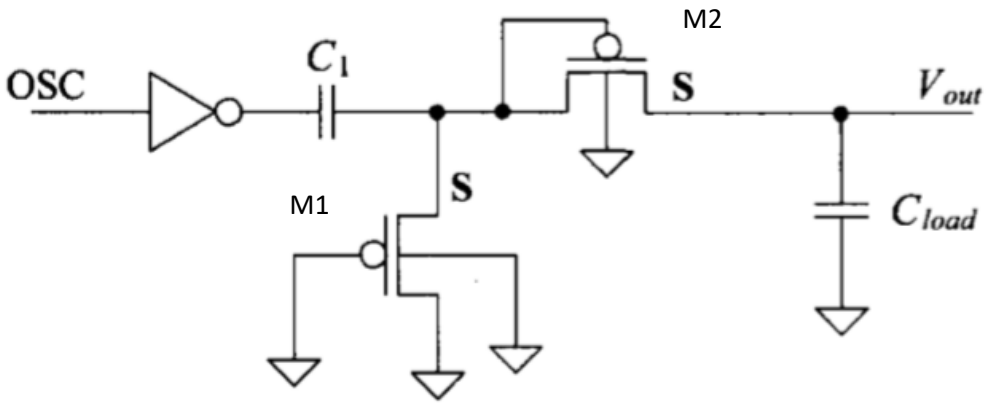


Abbildung 34: Ladungspumpe mit negativer Ausgangsspannung [40]

Anders als bei der Landungspumpe für positive Ausgangsspannungen werden bei negativen Ausgangsspannungen PMOS Transistoren eingesetzt. Der Grund dafür liegt im physikalischen Aufbau des PMOS. In Abbildung 34 ist ein weiterer Unterschied zur positiven Ladungspumpe zu erkennen. Die Bulk-Kontakte werden mit Masse verbunden und nicht mit der Source des PMOS. Wären die Source-Anschlüsse mit dem Bulk verbunden, so würde, wenn die Ausgangsspannung in den negativen Bereich fällt, auch das Nwell negativ und somit die Bulk-Nwell Diode leitend. Ist das Nwell (Bulk) mit Masse verbunden, so wird die Bulk-Nwell Diode nicht in Flussrichtung vorgespannt und es kommt zu keinem Stromfluss zwischen Nwell und Bulk. Der Body-Effekt, der sich auf Grund des Potentialunterschiedes zwischen Bulk und Source einstellt, wird hierbei in Kauf genommen.

Die negative Ausgangsspannung wird wie folgt generiert. Die in der Abbildung 34 nach links ausgerichtete Elektrode des Kondensator C_1 wird bei einer Low Phase des oszillierenden Signals auf das Potential der Eingangsspannung V_{DD} aufgeladen. Da über dem Transistor M1 mindestens eine Schwellenspannung abfallen muss, damit der Ladestrom fließen kann, lädt sich der Kondensator auf eine Spannung von V_{DD}-V_{THP} auf. Die rechte Elektrode des Kondensators lädt sich dann in etwa auf das V_{THP} Potential auf. Wird während der High Phase des Oszillators die linke Elektrode auf Masse gelegt, bleibt die Ladung im Kondensator erst mal konstant und damit der Spannungsabfall am Kondensator unverändert. Wenn die linke Elektrode des Kondensators von V_{DD} auf Masse absinkt, muss die rechte Elektrode ebenfalls um V_{DD} absinken, damit der Spannungsabfall am Kondensator konstant bleiben kann. Das Potential an der rechten Elektrode des Kondensators wird damit negativ und sinkt auf V_{THP}-V_{DD}. Zu diesem Zeitpunkt sperrt der untere Transistor M1 während der Transistor M2 leitend wird. Der Kondensator C_1 entlädt sich dann über M2 und lädt dabei C_{load} mit einer negativen Spannung auf. Im eingeschwungenen Zustand stellt sich eine negative Spannung ein. Die generierte Spannung wird im Betrag noch um die Schwellenspannung des Transistors M2 reduziert, so dass sich am Ausgang folgende Spannung ergibt. [40]

$$\mathrm{V_{out}} = -2 * \mathrm{V_{THP}} - \mathrm{V_{DD}}$$

4.6 Dickson Ladungspumpe

Die vorherigen Ladungspumpen sind nur in der Lage eine Ausgangsspannung von V_{out} = 2*(V_{DD} - V_{THN}) bis -2*V_{DD} zu generieren. Um einen Betrag größerer Spannungen zu generieren, ist es notwendig die Struktur der gezeigten Ladungspumpen zu kaskadieren. Dabei gibt es unterschiedliche Schaltungsarchitekturen, wobei hier die Dickson Ladungspumpe näher erläutert wird. Mit der Dickson Ladungspumpe ist es möglich ein positives Vielfaches der Ausgangsspannung zu generieren. Die Höhe

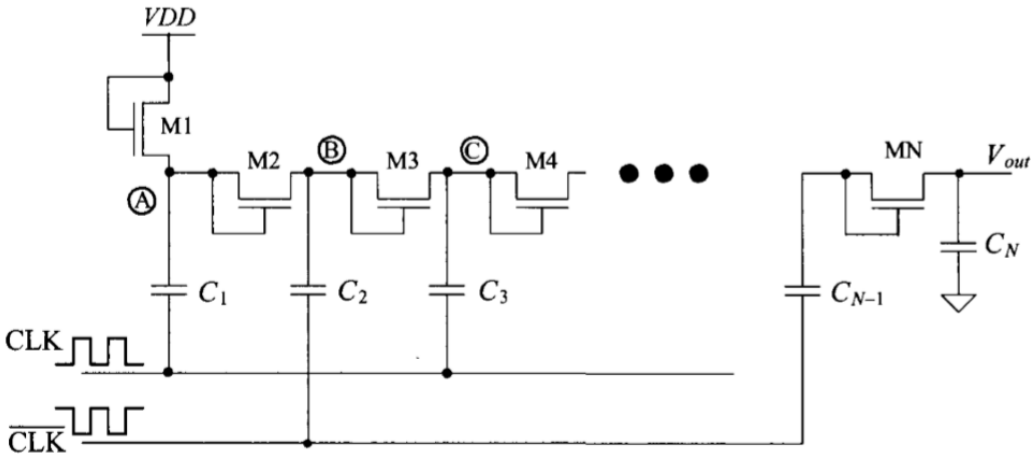


Abbildung 35: Kaskadierte Ladungspumpe nach Dickson[40]

der Ausgangsspannungen ist abhängig von den kaskadierten Stufen und lässt sich allgemein in der nachfolgenden Gleichung darstellen.

$$V_{out} = (N + 1) * V_{DD} - N * V_{THN}$$

In Abbildung 35 ist eine Dickson Ladungspumpe dargestellt. Die MOSFETs agieren in diesem Schaltungskonstrukt, wie bei den vorherigen Schaltungen, als Diode. [40]

In der Funktionsbeschreibung der Dickson Ladungspumpe wird davon ausgegangen, dass die Ladungspumpe eingeschwungen ist, und sich im Betriebszustand nach mehreren Taktzyklen befindet. [40]

5 AurixPlus-digital EVR and PMS - Testchip Version 1.5

Dieses Kapitel unterliegt der Gehimhaltung.

6 Programmiersprachen

In diesem Kapitel werden die eingesetzten Programmiersprachen vorgestellt. Sie wurden benötigt, um die grafische Oberfläche, die Steuerung der Messinstrumente und die Durchführung der Testabläufe zu implementieren. Diese Softwaremodule für die Steuerung der Testabläufe und die Erfassung der Messdaten wurden in C# geschrieben, während die Messdatenanalyse und die Generierung von Diagrammen aus den erfassten Messdaten mit Hilfe der Scriptsprache Python durchgeführt wurden. In diesem Kapitel werden die Programmiersprachen C# und Python näher erläutert.

6.1 C#

Die Programmiersprache C# (C-Sharp) wurde von Anders Hejlsberg im Auftrag von Microsoft entwickelt und 2001 in der Version 1 eingeführt. C# ist eine objektorientierte Programmiersprache, die sehr leistungsstark ist. Sie ist plattformunabhängig und die Hauptsprache der .NET-Umgebung von Microsoft. Plattformunabhängig bedeutet in diesem Kontext, dass sie Maschinencode für alle Plattformen anbietet, die C++ Maschinencode unterstützen. Somit ist es möglich, mit der „.Net-Core“ Umgebung und dem Open Source Mono-Projekt die entwickelte Software nicht nur unter Windows, sondern zusätzlich auch unter Linux und MacOS auszuführen. C# vereint Konzepte der Programmiersprachen Java, C, C++, Haskell und Delphi. Sie unterstützt funktionale, prozedurale, generische, objektorientierte und komponentenorientierte Programmierdisziplinen. Als Designziele der Sprachentwicklung wurden Software-Robustheit, Langlebigkeit und Programmierer-Produktivität formuliert. Ihr Einsatzgebiet ist weit gefächert und beinhaltet die Entwicklung von Konsolen-, GUI-, Web-Anwendungen sowohl für Desktop PCs als auch für eingebettete Systeme.

Die .NET Common Language Runtime ist eine komponentenbasierte Laufzeitumgebung, die Code ausführt und Dienste für den Entwicklungsprozess bereitstellt. Compiler und Tools machen es in dieser Umgebung möglich, Vorteile einer verwalteten Ausführungsumgebung zu nutzen. Unter „verwaltetem“ Code wird eine sprachübergreifende Inspiration, sprachübergreifende Ausnahmebehandlung, erhöhte Sicherheit, Unterstützung bei der Versionserstellung und Bereitstellung, ein vereinfachtes Modell für die Interaktion von Komponenten, sowie Debug- und Profilerstellungsdienste verstanden. Hinzu kommt, dass Compiler und Tools in der Lage sind, eine nutzbare Ausgabe zu erzeugen, die sowohl das Typsystem als auch das Format der Metadaten und die Laufzeitumgebung umfasst. All diese Ausgaben sind für die Common Language Infrastruktur [1] durch einen öffentlichen Standard definiert worden, den ECMA-Standard [2]. Die Sprache C# in der Version 4 ist seit April 2006 als ISO/IEC 23270:2006 veröffentlicht.

Durch das Konzept der komponentenbasierten Umgebung wurde die Erstellung von Komponenten in C# sehr vereinfacht. Alle Objekte werden als Komponenten geschrieben und bilden den Mittelpunkt

des Geschehens. Eigenschaften, Methoden und Ereignisse stellen die Kernelemente der Sprache und der zugrundeliegenden Laufzeitumgebung dar. Attribute können auf Komponenten angewendet werden, um andere Bestandteile des Systems wie Entwurfszeit- und Laufzeitinformationen einer Komponente zu liefern. C#-Objekte erfordern weder einen Header noch eine IDL-Datei der Typenbibliothek. Die erstellten Komponenten sind vollständig selbstbeschreibend und können ohne vorherige Registrierung eingesetzt werden.

C# zählt außerdem zu den typensicheren Programmiersprachen. Der Begriff Typensicherheit kommt aus der Informatik und bedeutet, dass Datentypen gemäß Ihrer Typdefinition verwendet werden müssen und keine Typverletzungen auftreten dürfen. Falls eine Datentypverletzung auftritt, kann der Compiler sie nicht verarbeiten und gibt einen Error aus. Programmiersprachen, die keine Typensicherheit aufweisen, sind zum Beispiel C und Java. Durch die fehlende Typenkontrolle kann ein fehlerhafter oder leicht angreifbarer Code erzeugt werden. Aufgrund dieser Typensicherheit ist C# eine sichere Programmiersprache.

6.2 Python

Python ist eine universelle und üblicherweise interpretierte höhere Programmiersprache [3], die auf schnellen Erfolg und einen syntaktischen einfachen Umgang ausgelegt ist. Der Focus bei der Entwicklung von Python lag vor allem auf Einfachheit und Übersichtlichkeit. Durch die geringen Syntax-Anforderungen ist ihr Anspruch, einen gut lesbaren und knappen Programmierstiel zu fördern, gelungen. Unter einfacher syntaktischer Umgebung wird verstanden, dass Programm-Blöcke nicht durch geschweifte Klammern, sondern durch Einrücken des Programmcodes durchgeführt werden. Dadurch wird der Code für Einsteiger nicht durch Trennzeichen unnötig verkompliziert. Des Weiteren werden wenige Schlüsselwörter verwendet, die den Umgang mit der Sprache noch ein weiteres Stück einfacher machen.

Python wurde Anfang der 1990er Jahre von Guido van Rossum am Centrum Wiskunde & Informatica in Amsterdam als Nachfolger für die Programmier-Lehrsprache ABC entwickelt. Eingesetzt werden sollte die Sprache für das verteilte Betriebssystem Amoeba. Die Namensgebung stammt nicht von der gleichnamigen Schlange, sondern bezog sich auf die Komikertruppe von Monty Python. Die erste Vollversion, Python 1.0, wurde im Januar 1994 veröffentlicht und wird bis heute weiterentwickelt. Aktuell werden zwei Versionen parallel zueinander gepflegt. Die Python Version 2.0 wurde im Oktober 2000 veröffentlicht, wird bis heute gepflegt und befindet sich aktuell in der Version 2.7. Der zweite Zweig wurde im Dezember 2008 veröffentlich. Python 3.0 beinhaltet tiefgründige Änderungen in der Benutzung und ist teilweise inkompatibel zu vorherigen Versionen. Aus diesem Grund werden die Versionen 2.X und 3.X parallel weitergeführt. Die aktuellste Python Version 3.7 wurde im Juni 2018 veröffentlicht. Python setzt auf ein gemeinschaftsbasiertes und offenes Entwicklungsmodell. Die

Pflege der Referenzumgebung „CPython“ wird durch die gemeinnützige Organisation der „Python Software Foundation“ durchgeführt. [4][5]

Python unterstützt mehrere Konzepte der modernen Informatik, darunter fallen die Objektorientierung, die funktionale- und die aspektorientierte Programmierung. Aufgrund dieser Vorteile wird Sie häufig als Skriptsprache eingesetzt. Bei der Entwicklung von Python wurde aus den Schwächen des Vorgängers ABC gelernt. ABC hatte keine überschaubaren, leicht anwendbaren und erweiterbaren Standardbibliotheken. Daher wurde bei der Entwicklung von Python darauf großer Wert gelegt. Python-Skripte sind in Ihrer Länge deutlich knapper und effektiver im Vergleich zu anderen Skriptsprachen. Der Ruhm der letzten Jahre lässt sich aus einem offiziellen Ranking der beliebtesten Programmiersprachen ablesen. Dabei ist Python schon auf Platz 4 (2018) der beliebtesten Programmiersprachen und hat mittlerweile C# auf den Platz 5 verwiesen. [6]

7 Hardware und Setup

In diesem Kapitel werden die eingesetzten Messinstrumente und die zusätzlichen Hardwarekomponenten aufgezählt, die für die Messungen nötig sind. Darunter fallen Messgeräte und Quellen von unterschiedlichen Herstellern wie Rhode & Schwarz, Agilent und Keithley. Außerdem wird die Funktionsweise des Thermostreams zum Erreichen der einzelnen Messtemperaturen erklärt. Die Kommunikation zwischen dem PC bzw. der Software und allen verwendeten Geräten wird über die GPIB-Schnittstelle (General Purpose Interface Bus) hergestellt. Der Befehlssatz zur Steuerung der Geräte basiert auf den SCPI-Standard (Standard Commands for Programmable Instruments). Im folgenden Kapitel wird die Funktionsweise näher betrachtet und erklärt. Jeder Instrumentenhersteller definiert für seine Geräte spezielle SCPI- Schlüsselwörter und Kommandos. Daher ist es nötig, die implementierten Kommandos für alle Instrumente zu erläutern. Außerdem werden die Maßnahmen, die für die Inbetriebnahme des Messplatzes durchgeführt werden müssen, vorgestellt.

7.1 Treiberinstallation

Für die Etablierung der Kommunikation zwischen PC und den Instrumenten, muss zunächst der GPIB-Treiber auf dem PC installiert werden. Verwendet wird hierbei der IVI-VISA-Treiber (Interchangeable Virtual Instruments, Virtual Instrument Software Architecture) der IVI Foundation (Interchangeable-Virtual-Instruments Stiftung). Dieser Treiber stellt für unterschiedliche Programmiersprachen Standardbibliotheken für die Kommunikation mit Messgeräten zur Verfügung. Darunter fallen auch COM-Elemente, die innerhalb der C#-Programmierung eingesetzt werden können. Der Treiber kann von der offiziellen Homepage www.ivifoundation.org heruntergeladen werden. Die IVI Foundation wurde für die Pflege und Bereitstellung der Treiber gegründet. Die IVI Stiftung übernimmt diese Aufgabe seit 2002. Die Aufgaben der IVI sind Entwicklung und Verabschiedung und Förderung von Standards für die Programmierung von Messinstrumenten. Dazu werden die vorhandenen Standardbibliotheken stetig weiterentwickelt und dabei die Herstellerunabhängigkeit gewahrt. Diese Vorgehensweise erlaubt den Einsatz des Treibers für Geräte aller bekannten Hersteller. Das vereinfacht den Umgang mit Instrumenten verschiedener Hersteller, da die Installation von unterschiedlichen Treibern nicht mehr notwendig ist. Die IVI baut auf bestehende Industriestandards auf, um Spezifikationen zu erstellen und dem Anwender bessere Leistungen und vereinfachte Wartung zu bieten.

Da heutige PC-Systeme über keinen Parallel-Port-Anschluss mehr verfügen, muss ein USB zu GPIB Adapter eingesetzt werden. In diesem Projekt wurde der 82357 USB/GPIB Interface High-Speed USB 2.0 Dongle von der Firma Keysight eingesetzt. Die Produkt-Homepage enthält alle benötigten Informationen über diesen Interface-Adapter. Dieser Adapter funktioniert nach einem Plug & Play Prinzip und benötigt keinen zusätzlichen Treiber. [7]

7.2 GPIB und der Befehlssatz SCPI

7.2.1 GPIB

Abbildung 36: Stapelbarer GPIB Anschluss Stecker

Der General Purpose Interface Bus ist ein paralleler 8-Bit Multi-Master Interface Bus, der im IEEE 488 spezifiziert wurde. Er wurde anfänglich von Hewlett & Packard unter dem Namen HP-IB entwickelt und ist heutzutage auch bekannt unter dem Namen GPIB. Die Entwicklung dieses Busses wurde in den späten 1960er Jahren abgeschlossen und im Jahr 1975 unter der Referenz 488.1975 als IEEE Standard veröffentlicht. Seitdem wurde der Standard stetig weiterentwickelt, und 1990 durch die

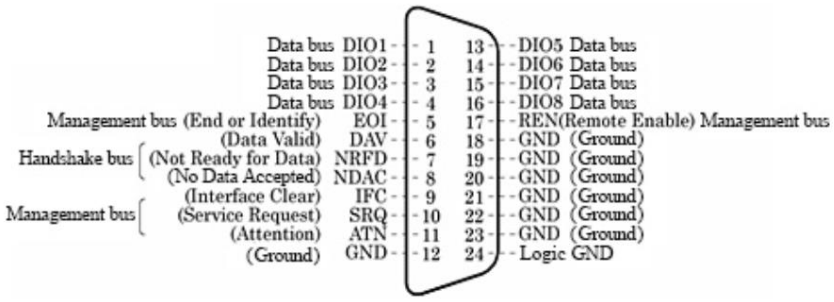


Abbildung 37: Pinout vom GPIB Stecker [8][9]

Kommandosprache SCPI erweitert (IEEE 488.2). Dieser Befehlssatz wird im nächsten Abschnitt näher erläutert. Die GPIB-Schnittstelle wurde entwickelt, um eine Automatisierung und Kopplung von Messinstrumenten durchzuführen. Mit GPIB ist es möglich bis zu 15 Geräte in Reihe miteinander zu verbinden. Die Verbindung zwischen dem PC und den einzelnen Instrumenten geschieht über einen stapelbaren Anschlussstecker. Dieser besitzt 16 Signalleitungen und acht Masseleitungen. In Abbildung 36 ist der Stecker abgebildet und in Abbildung 37 ist die Kontaktbelegung des GPIB-Steckers zu sehen.

Die 16 Signalleitungen setzen sich wie folgt zusammen und haben folgende Aufgaben: [10]

- 8x Datenleitungen
    - DOI1-DOI8
- 5 x Management Bus Leitungen
    - EOI (End of Identify)
        - Signalisiert Ende des Datenwortes
    - SRQ (Service Request)
        - Vergleichbar mit Interrupt Leitung im Microcontroller
    - ATN (Attention)
        - Datenleitung enthält Kommando-Byte
    - REN (Remote Enable)
        - Remote-Modus für Busteilnehmer freigeben
    - IFC (Interface Clear)
        - System-Controller kann Bus zurücksetzen
- 3x Handshake Bus Leitungen
    - DAV (Data Valid)
        - Datensatz gültig
    - NRFD (Not Ready For Data)
        - Verarbeitung der Daten noch nicht beendet
    - NDAC (No Data Accepted)
        - Datenwort auf DOI nicht akzeptiert

Die Kommunikation wird über ein Talker-, Controller- und Listener-Konzept aufgenommen. Wobei der Controller festlegt, welche Geräte ihre Sendeschnittstelle aktivieren dürfen. Jedem Gerät wird ein 5-Bit-BCD-Code zugewiesen, der für das Instrument eindeutig sein muss. Dadurch kann der Listener entscheiden, ob das ankommende Datenwort für ihn bestimmt ist. Wenn die Adressierung von Gerät und Datenwort nicht übereinstimmt, wird der Befehl auch nicht ausgeführt. Daher erhält jedes Instrument jedes Datenwort, wobei nur bei übereinstimmender Adresse der Befehl ausgeführt wird. Außerdem darf zu einem gewissen Zeitpunkt nur ein Talker seine Daten zum Controller oder Listener senden. Auf Grund des parallelen Kommunikationsansatzes ist GPIB ein sehr langsames Übertragungsprotokoll. In Abbildung 38 ist das Konzept der Kommunikation grafisch dargestellt.

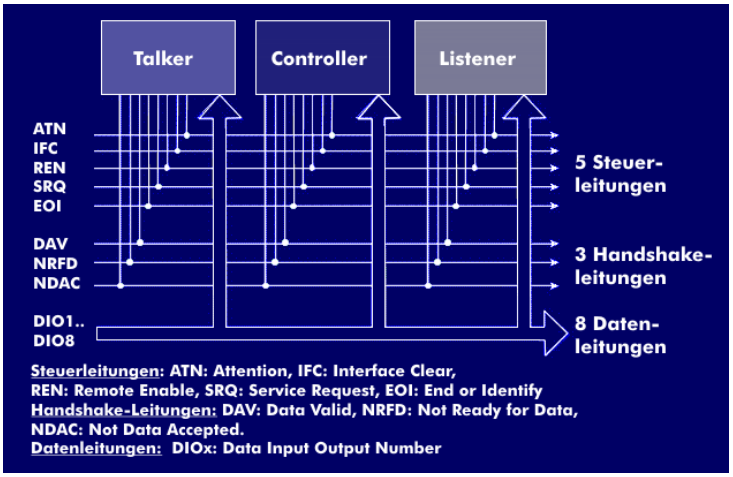


Abbildung 38: Talker, Controller, Listener Konzept der GPIB Kommunikation [11]

Der Handshakeprozess läuft in drei Phasen ab, wobei das langsamste Gerät die Übertragungsgeschwindigkeit auf dem Bus bestimmt. In der ersten Phase ist das Instrument bereit, Daten zu empfangen. Dieser Zustand wird durch ein „low“ auf der NRFD-Leitung signalisiert. In der zweiten Phase wird überprüft, ob die empfangenen Daten gültig sind, was über ein „high“ am DAV-Bit signalisiert wird. Die dritte Phase, in der geprüft wird, ob die empfangenen Daten akzeptiert werden, wird über ein „low“ auf dem NDAC-Bit eingeleitet. Diese drei Phasen werden bei jeder Verarbeitung eines Datenwortes durchlaufen. GPIB definiert nur wie ein Datenwort zu einem Instrument übertragen wird und enthält keine Kommandos zur Steuerung dieser Instrumente. Damit diese Lücke geschlossen wird, wurde der Befehlssatz SCPI entwickelt.

7.2.2 Kommunikationsprotokoll SCPI

SCPI (Standard Commands for Programmable Instruments) ist ein Befehlssatz, welcher grundlegende als auch einige spezielle Kommandos für Messgeräte aller Art definiert. Die Befehle werden in ASCII-Text-Form übertragen, was den Einsatz in jeder Programmiersprache und in jeder Entwicklungsumgebung erlaubt. Der Standard zielt auf die Vereinheitlichung der Geräteschnittstellen unterschiedlicher Hersteller von Messgeräten ab, um die Einarbeitung, den Entwicklungsaufwand und den Gerätewechsel zu vereinfachen. Mittlerweile benutzt jeder Hersteller von programmierbaren Messinstrumenten den SCPI Standardbefehlssatz. Dieser Befehlssatz ist in Abbildung 39 gezeigt.

*CLS	Clear Status Command
*ESE	Standard Event Status Enable Command
*ESE?	Standard Event Status Enable Query
*ESR?	Standard Event Status Register Query
*IDN?	Identification Query
*OPC	Operation Complete Command
*OPC?	Operation Complete Query
*RST	Reset Command
*SRE	Service Request Enable Command
*SRE?	Service Request Enable Query
*STB?	Read Status Byte Query
*TST?	Self-Test Query
*WAI	Wait-to-Continue Command

Abbildung 39: Standard Kommandos die jeder Hersteller aus dem SCPI implementiert hat [12]

Bei Kommandos bei denen ein „?“ am Ende des Befehls steht, wird ein Rückgabewert mit Informationen vom Instrument zum Controller zurückgesendet.

Ein programmierbares Messinstrument verwendet die Formatierung des Befehlssatzes, der ebenfalls im SCPI festgelegt wurde. Es gibt zwar immer noch instrumentenspezifische Befehle, die sich von Hersteller zu Hersteller unterscheiden, aber auch diese speziellen Befehle halten sich dann an die standardisierte Formatierung der übertragenen Befehle. Einige Unterschiede werden im folgenden Abschnitt erläutert, in dem die verwendeten Instrumente aufgezählt und erklärt werden. Die Formatierung der Befehlssätze setzt sich wie folgt zusammen und wird an einem Beispiel für das Setzen und Abfragen der BAUD-Rate (Übertragungsgeschwindigkeit der RS232-Schnittstelle) gezeigt.

- **SYST**em:**COMM**unicate:**SER**ial:**BAUD** 2400
    - Mit diesem Kommando wird die BAUD Rate für das Instrument auf 2400 Bit/s gesetzt
    - Der Befehl ist auch in dieser Form gültig: **SYST:COMM:SER:BAUD** 2400
- **SYST:COMM:SER:BAUD?**
    - Mit dem „?“ am Ende wird ein Query signalisiert, und der Controller erwartet einen Rückgabewert
    - In diesem Fall würde der Rückgabewert 2400 Bit/s sein

Der Standard definiert zwei Formen von Befehlssätzen. In der obigen Aufzählung müssen nur die Großbuchstaben übertragen werden, damit das Instrument den Befehl ausführt. Daher wird das Übertragene Datenwort kleiner und bietet einen geringeren Aufwand zur Implementierung in die Software.

7.3 Eingesetzte Instrumente und Hardware

7.3.1 Keithley 2000 Multimeter

Abbildung 40: Keithley 2000 Digital Multimeter in der Frontansicht [13]

Das Keithley 2000 Digital Multimeter ist ein 6 ½ Stellen Messgerät mit einer hohen Genauigkeit und schnellen Messgeschwindigkeit (Abbildung 40). Durch die hohe Messgeschwindigkeit lassen sich auch dynamische Messungen mit den internen Funktionen realisieren, was die Flexibilität des Messgerätes erhöht. Dieses Multimeter ist in der Industrie sehr weiter verbreitet, da es durch seine Schnelligkeit einen hohen Durchsatz bei Messungen erreicht. Ein Vorteil des Gerätes besteht darin, dass derselbe A/D Wandler wie bei den hochpreisigen 7 ½ und 8 ½ Messgeräten eingesetzt wird und dadurch eine hohe Genauigkeit erzielt wird. Die Gleichspannungsempfindlichkeit liegt bei nur 100 nV und besitzt eine Grundgenauigkeit von 0,002%. Die weiteren elektrischen Charakteristika sind der Abbildung 41

DC Characteristics

Conditions:	MED (1 PLC)[1] or SLOW (10 PLC) or MED (1 PLC) with filter of 10				Accuracy: ±(ppm of reading + ppm of range) (ppm = parts per million) (e.g., 10ppm = 0.001%)			
Function	Range	Resolution	Test Current or Burden Voltage (±5%)	Input Resistance	24 Hour[14] 23°C ± 1°	90 Day 23°C ± 5°	1 Year 23°C ± 5°	Temperature Coefficient 0°–18°C and 28°–50°C
Voltage	100.0000 mV	0.1 µV		> 10 GΩ	30 + 30	40 + 35	50 + 35	2 + 6
	1.000000 V	1.0 µV		> 10 GΩ	15 + 6	25 + 7	30 + 7	2 + 1
	10.00000 V	10 µV		> 10 GΩ	15 + 4	20 + 5	30 + 5	2 + 1
	100.0000 V	100 µV		10 MΩ ±1%	15 + 6	30 + 6	45 + 6	5 + 1
	1000.000 V[9]	1 mV		10 MΩ ±1%	20 + 6	35 + 6	45 + 6	5 + 1
Resistance[15]	100.0000 Ω	100 µΩ	1 mA		30 + 30	80 + 40	100 + 40	8 + 6
	1.000000 kΩ	1 mΩ	1 mA		20 + 6	80 + 10	100 + 10	8 + 1
	10.00000 kΩ	10 mΩ	100 µA		20 + 6	80 + 10	100 + 10	8 + 1
	100.0000 kΩ	100 mΩ	7 µA		20 + 6	80 + 10	100 + 10	8 + 1
	1.000000 MΩ[16]	1 Ω	7 µA		20 + 6	80 + 10	100 + 10	8 + 1
	10.00000 MΩ[11, 16]	10 Ω	700 nA // 10MΩ		150 + 6	200 + 10	400 + 10	95 + 1
	100.0000 MΩ[11, 16]	100 Ω	700 nA // 10MΩ		800 + 30	1500 + 30	1500 + 30	900 + 1
Current	10.00000 mA	10 nA	< 0.15 V		60 + 30	300 + 80	500 + 80	50 + 5
	100.0000 mA	100 nA	< 0.03 V		100 + 300	300 + 800	500 + 800	50 + 50
	1.000000 A	1 µA	< 0.3 V		200 + 30	500 + 80	800 + 80	50 + 5
	3.00000 A	10 µA	< 1 V		1000 + 15	1200 + 40	1200 + 40	50 + 5
Continuity 2W	1 kΩ	100 mΩ	1 mA		40 + 100	100 + 100	120 + 100	8 + 1
Diode Test	3.00000 V	10 µV	1 mA		20 + 6	30 + 7	40 + 7	8 + 1
	10.00000 V	10 µV	70 µA		20 + 6	30 + 7	40 + 7	8 + 1
	10.00000 V	10 µV	7 µA		20 + 6	30 + 7	40 + 7	8 + 1

Abbildung 41: DC-Characteristics Keithley 2000 [13]

zu entnehmen, welche einem Auszug aus dem Datenblatt entspricht. Eingesetzt wird das Keithley für jede getätigte Spannungsmessung, um reproduzierbare Ergebnisse und eine Vergleichbarkeit der Resultate zu gewährleisten. [13]

7.3.2 Keithley 2450 Source Meter SMU

Abbildung 42: Keithley 2450 Source Meter SMU Front Ansicht [14]

Die 2450 Source Meter Unit (SMU) von Keithley, dargestellt in Abbildung 42, bietet eine Vier Quadranten-Spannungs- und Stromquelle mit Messfunktion. Die SMU ist in der Lage gleichzeitig eine Spannung oder einen Strom zu erzeugen und gleichzeitig zu messen. Dabei sind an der Front Sense-Anschlüsse zum Messen und Force-Anschlüsse zum Erzeugen von Spannung oder Strom platziert. Die

Voltage Specifications [1, 2]

	Source			Measure [3]		
Range	Resolution	Accuracy (23° ± 5°C) 1 Year ±(% setting + volts)	Noise (RMS) (<10 Hz)	Resolution	Input Resistance	Accuracy (23° ± 5°C) 1 Year ±(% rdg. + volts)
20.00000 mV	500 nV	0.100% + 200 µV	1 µV	10 nV	>10 G□	0.100% + 150 µV
200.0000 mV	5 µV	0.015% + 200 µV	1 µV	100 nV	>10 G□	0.012% + 200 µV
2.000000 V	50 µV	0.020% + 300 µV	10 µV	1 µV	>10 G□	0.012% + 300 µV
20.00000 V	500 µV	0.015% + 2.4 mV	100 µV	10 µV	>10 G□	0.015% + 1 mV
200.0000 V	5 mV	0.015% + 24 mV	1 mV	100 µV	>10 G□	0.015% + 10 mV

Current Specifications [1, 2]

	Source			Measure [3]		
Range	Resolution	Accuracy (23° ±5°C) [4] 1 Year ±(% setting + amps)	Noise (RMS) (<10 Hz)	Resolution	Voltage Burden	Accuracy (23° ±5°C) 1 Year ±(% rdg. + amps)
10.00000 nA [5]	500 fA	0.100% + 100 pA	500 fA	10 fA	<100 µV	0.100% + 50 pA
100.0000 nA [5]	5 pA	0.060% + 150 pA	500 fA	100 fA	<100 µV	0.060% + 100 pA
1.000000 µA	50 pA	0.025% + 400 pA	5 pA	1 pA	<100 µV	0.025% + 300 pA
10.00000 µA	500 pA	0.025% + 1.5 nA	40 pA	10 pA	<100 µV	0.025% + 700 pA
100.0000 µA	5 nA	0.020% + 15 nA	400 pA	100 pA	<100 µV	0.020% + 6 nA
1.000000 mA	50 nA	0.020% + 150 nA	5 nA	1 nA	<100 µV	0.020% + 60 nA
10.00000 mA	500 nA	0.020% + 1.5 µA	40 nA	10 nA	<100 µV	0.020% + 600 nA
100.0000 mA	5 µA	0.025% + 15 µA	100 nA	100 nA	<100 µV	0.025% + 6 µA
1.000000 A	50 µA	0.067% + 900 µA	3 µA	1 µA	<100 µV	0.030% + 500 µA

Temperature Coefficient
(0°–18°C and 28°–50°C) ±(0.15 × accuracy specification)/°C.

Abbildung 43: Keithley 2450 Source Meter SMU Spannungs- und Stromspezifikationen [15]

SMU hat einen Spannungsbereich von 100nV bis 200V bei einer Gesamtleistung von 100 W (DC). Die weiteren Spannungs- und Stromspezifikationen sind in Abbildung 43 als Auszug aus dem Datenblatt, gezeigt [15].

7.3.3 Rhode und Schwarz Power Supply HMC8043

Abbildung 44: Power Supply HMC8043 von Rhode und Schwarz in der Front Ansicht [16]

Die eingesetzte Spannungsquelle HMC8043 von Rhode und Schwarz, dargestellt in Abbildung 44, hat 3 Kanäle und eine maximale Ausgangslast von 100W, die auf alle 3 Kanäle gleichmäßig verteilt werden. In diesem Fall hat somit jeder einzelne Kanal eine Leistung von 33 W. Sie verfügt zur Kommunikation und Steuerung über LAN-, USB-, RS232- und GPIB-Schnittstellen und ist somit in Bezug auf den gewählten Kommunikationsweg sehr flexibel [17].

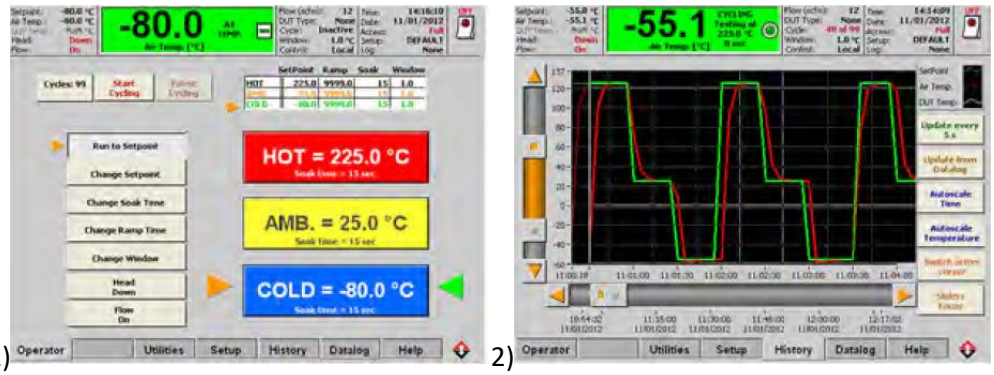
7.3.4 Temptronic ATS 710-M Thermostream

(*)

Abbildung 45: Thermostream von Temptronic ATS 710-M [18]

Der ATS 710-M (Advanced Tepmerature Source), welcher in Abbildung 45 gezeigt ist, stellt ein Temperaturgerät dar, das in Messplätzen eingesetzt wird, um die Temperaturbereiche von -80°C bis 225°C zu erreichen. Die Einstellung der gewünschten Temperatur wird dabei ohne die Zufuhr von flüssigem Stickstoff erreicht. Die Temperatur wird über einen Wärmetauscher und Druckluft mit hohem anliegendem Druck erreicht. Außerdem läuft die Temperierung frostfrei ab, da die auf das Werkstück geleitete Luft gleichzeitig getrocknet wird. Unter Laborbedingungen entsteht daher kein Eis am Werkstück, wodurch die Auswerteelektronik geschont wird. Gesteuert werden kann der Thermostream über das GPIB-Interface und über Ethernet (LAN). Die Temperaturgenauigkeit liegt bei +/- 0,1°C und ist damit ausreichend für die Qualifizierung von Automotive Produkten. Der Druck der ausströmenden Luft sollte im Bereich zwischen 6,2 und 7,6 Bar liegen, damit eine einwandfreie Funktion gewährleistet werden kann. In diesem Projekt wird der Thermostream für die Temperaturbereiche -40°C, 25°C und 125°C eingesetzt, was den üblichen Temperaturbereichen für Automotive Produkte entspricht. Neben den Fernsteuerungsmöglichkeiten besitzt der Thermostream auch noch einen Touchscreen zur lokalen Steuerung. Diese Oberfläche ist in Abbildung 46 gezeigt und dient zur manuellen Kontrolle der geloggten Daten und zum Einstellen einer Temperaturrampe. Damit die Substrattemperatur gemessen werden kann, wird noch ein externer Temperaturfühler in Form eines NTCs benötigt, der mit dem Substrat verbunden wird. Dadurch ist es möglich, während einer Messung die erreichte Temperatur zu erfassen. Damit die genannten Temperaturen erreicht werden,

wird der Test Head, markiert mit (*) in Abbildung 45, auf das Substrat/PCB abgelassen, wodurch ein geschlossener Raum mit kleinem Volumen entsteht, der die gewünschten Temperaturen schnell annimmt. Die Glocke besitzt an der Unterseite eine Dichtung und schließt dadurch sehr gut mit dem Werkstück ab.



1) 2)

Abbildung 46: 1) Operator Screen zur lokalen Steuerung, 2) Datalog Screen zur visuellen Darstellung der Temperaturrampen

Quelle: https://www.intestthermal.com/pdfs/Thermal_ds/TemperatureForcing_ATS710M.pdf

8 GUI und Testsoftware

In diesem Abschnitt geht es um die Entwicklung der Testsoftware zur Charakterisierung von Modulen und Features im Microcontroller. Es werden die grafische Oberfläche, die Ansteuerung der Messinstrumente und die Testablaufsteuerung als drei zentrale Themenbereiche vorgestellt. Außerdem wird ein kurzer Blick auf die IDE gelegt, die eingesetzt wird, um die C# Testsoftware zu entwickeln.

8.1 IDE - Visual Studio 2015 Professional

Visual Studio wurde von Microsoft entwickelt und vereint diverse Hochsprachen in einer integrierten Entwicklungsumgebung. Es wurde 1997 erstmalig für Windows veröffentlicht. Aktuell befindet sich die

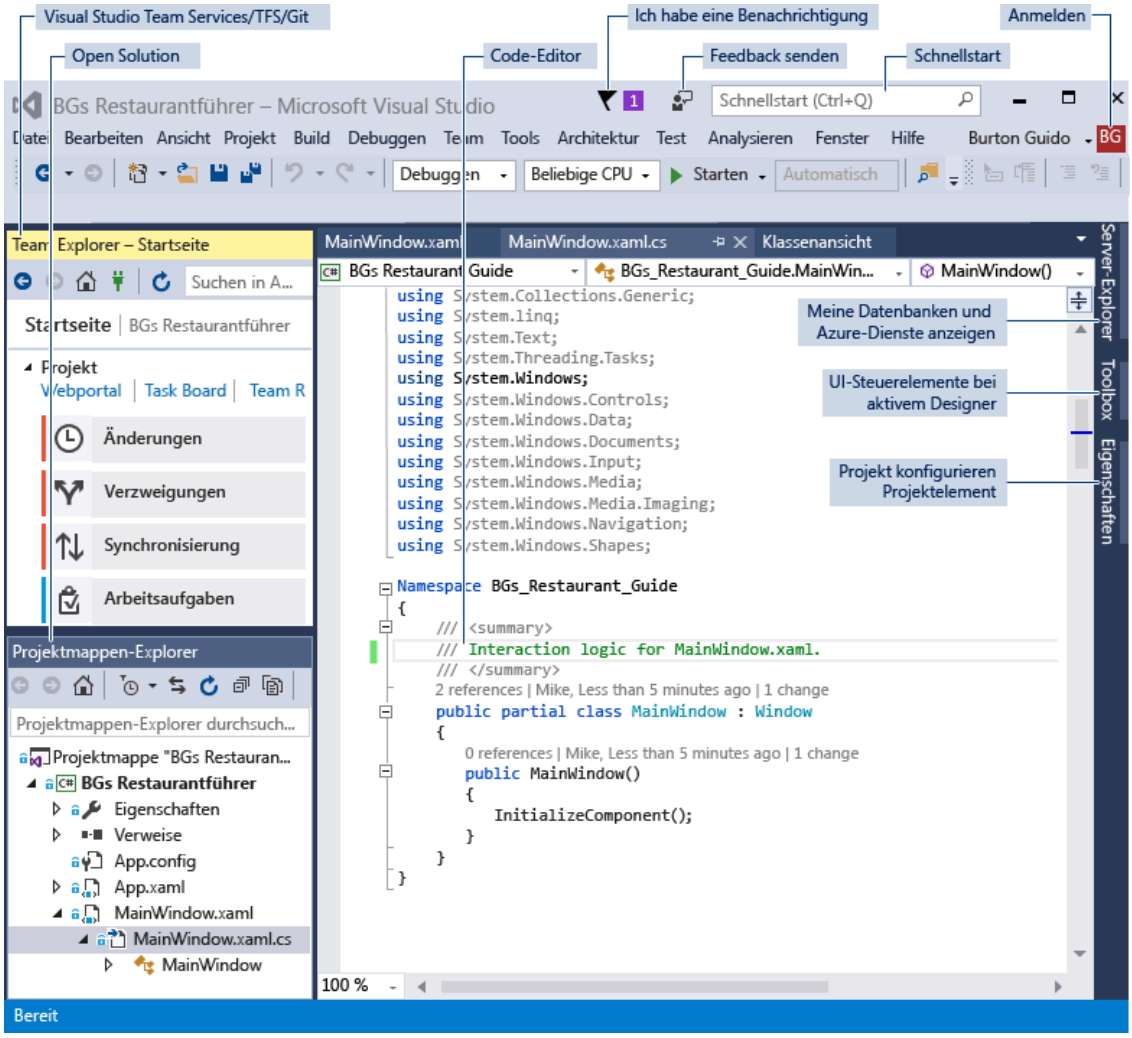


Abbildung 47: GUI von Visual Studio [20]

Umgebung in der Version 2017. Es unterstützt die Programmier- und Skriptsprachen C, C++, C#, Python, Visual Basic .Net, R-Script. Integriert ist ein Code-Debugger zur Code- und

Funktionalitätsüberprüfung. Der Aufbau von Visual Studio wird mit zusätzlichen Erklärungen der unterschiedlichen Bedienflächen in Abbildung 47 veranschaulicht.

In diesem Projekt wird Microsoft Visual Studios 15 – Professional für die Entwicklung der graphischen Testsoftwareoberfläche und Testprogrammentwicklung mit C# eingesetzt.

8.2 Beschreibung der Benutzeroberfläche zur RBB Testsoftware

In diesem Abschnitt werden die Benutzeroberfläche und die dahinterliegenden Funktionen beschrieben. Die Benutzeroberfläche wurde mit der integrierten Toolbox, welche eine Sammlung aller in Windows verwendeten Bedienelemente bereitstellt, entwickelt. Diese Toolbox dient zur raschen Entwicklung von Benutzeroberflächen in Visual Studio. Die Templates lassen sich durch Drag & Drop aus der Toolbox in den gewünschten Bereich der Benutzeroberfläche ziehen. Anschließend generiert Visual Studio automatisch ein Code-Gerüst für jedes verwendete Element in der erstellten Oberfläche. Durch die automatische Generierung des statischen Codes kann man von einer raschen Entwicklung der Oberfläche sprechen. Anschließend können Buttons, Dropdown-Menus und Textfelder im Code-Gerüst mit Funktionalität gefüllt werden. Dadurch wird die Benutzeroberfläche mit den entwickelten C#-Funktionen verbunden und kann zur Steuerung des Messplatzes herangezogen werden.

In Abbildung 48 ist die entwickelte Benutzeroberfläche zur RBB TC1.5 Testsoftware gezeigt. Dieses

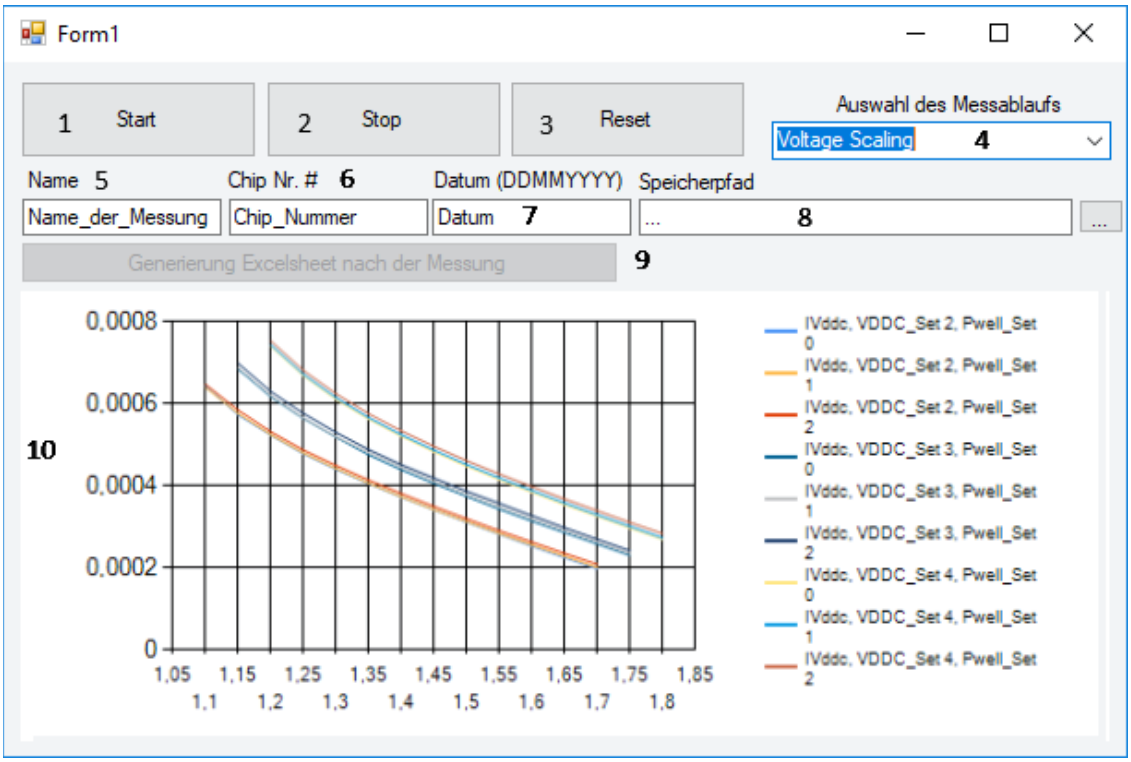


Abbildung 48: Benutzeroberfläche zur RBB TC1.5 Testsoftware

Softwaretool besteht aus zehn unterschiedlichen Elementen:

1. Start-Button
    - Dient zum Starten des Messplatzes und startet die Initialisierung der Testsoftware
2. Stop-Button
    - Dient zum Stoppen des Testprogramms
3. Reset-Button
    - Stoppt das aktuelle Testprogramm und startet es umgehend neu
4. Drop-Down-Menu
    - Drop-Down-Menu zur Auswahl der Testprogramme
5. Textfeld-Name
    - Benennung der aktuellen Messung
    - Wird für die Namensgebung der erstellten Messdaten verwendet
6. Textfeld-Chipnummer
    - Benennung, der gerade gemessenen Chips
    - Wird für die Namensgebung der erstellten Messdaten verwendet
7. Textfeld-Datum
    - Benennung, des Datums, an dem der Chip gemessen wurde
    - Wird für die Namensgebung der erstellten Messdaten verwendet
8. Browse Pfad
    - Legt den Ablageort für die gemessenen Messdaten an
9. Button „Generierung Excelsheet nach der Messung“
    - Sollte eine automatische Konvertierung von .csv zu .xls Dateien durchführen
    - Wurde nicht implementiert
    - Aufgabe hat das Post Processing übernommen
    - Button immer deaktiviert
10. Chart Messdatenerfassung
    - Stellt in einem Diagramm die aktuell gemessenen Daten grafisch dar
    - Legende rechts zeigt an, welcher Parameter zurzeit gemessen wird

Alle Elemente haben dahinterliegende Funktionen, die bei der Eingabe in die Textfelder oder beim Drücken der Buttons ausgeführt werden. Diese werden im Folgenden anhand von implementiertem Code gezeigt.

8.2.1 Start-Button

```
103         private void button1_Click(object sender, EventArgs e)
104         {
105
106             if (
107                 !(
108                 (textBox2.Text.Length == 0) ||
109                 (textBox3.Text.Length == 0) ||
110                 (textBox4.Text.Length == 0) ||
111                 (textBox5.Text.Length == 0)
112                 )
113
114                 )
115             {
116                 chart1.Series.Clear();
117                 var ComboboxInhalt = comboBox1.Text.ToString();
118                 ComboBoxInhalt = ComboboxInhalt;
119
120                 backroundWorker.WorkerSupportsCancellation = false;
121                 test.MeasurementStatus += UpdateChart;
122                 backroundWorker.RunWorkerAsync();
123                 button1.Enabled = false;
124                 button2.Enabled = true;
125             }
126             else
127             {
128                 MessageBox.Show("Bitte Eingabe überprüfen und vervollständigen");
129             }
130         }
131
```

*Abbildung 49: Inhalt der privaten Funktion **button1_Click,** die beim „klicken“ des Start-Buttons ausgeführt wird*

Die Hauptfunktion des Start-Buttons ist es, den ausgewählten Test zu starten. Bevor der Test gestartet werden kann, werden die Eingaben in den Textfeldern überprüft. Dafür wurde eine „if“ Bedingung verwendet die in Zeile 106 bis 112 in der Abbildung 49 dargestellt ist. Der logische „!“ Operator dient als Negierung des kompletten Ausdrucks. In dieser Bedingung wird die Länge des Inhalts aller Textfelder überprüft und ermittelt, ob die Länge des Inhalts gleich null ist. Wenn die Länge gleich null ist, entspricht das einem nicht ausgefüllten Feld und ist so nicht zulässig. Jedes Textfeld wird überprüft und mit einem „oder“ zusammengefügt. Der komplette Ausdruck wird anschließend, mit „!“, negiert. Diese Bedingung ist nur wahr, wenn in jedem Textfeld ein Inhalt existiert. Der eingetragene Inhalt wird nicht auf Korrektheit, sondern nur auf seine Länge hin überprüft. Wenn die Bedingung nicht zutrifft, wird dem Benutzer eine Nachricht per Fehlermeldung gegeben, damit dieser seine Eingaben überprüft (Zeile 128).

In Zeile 116 wird anschließend der Inhalt des Charts der letzten Messung entfernt. Dafür wird über das „chart1“ Objekt die existierende Reihe mit „clear()“ gelöscht.

Als nächstes wird der Inhalt des Drop-Down-Menus an die Property „ComboBoxInhalt“ übergeben. Eine Property ist sehr ähnlich zu einer globalen Variablen. Sie übergibt klassenübergreifend Inhalte, die an einer anderen Stelle des Programms benötigt werden. Durch ihre „Get/Set“ Eigenschaften sind sie aber deutlich sicherer und flexibler in Ihrer Nutzung. Der „getter“ wird immer ausgeführt, wenn der

Wert dieser Eigenschaft gelesen wird. Der „setter“ kommt immer dann zum Einsatz, wenn durch eine Funktion der Wert verändert und zurück auf die Eigenschaft geschrieben wird. In dieser Eigenschaft wurde aber keine Funktion für den „getter/setter“ hinterlegt, und somit wird nur eine einfache Variablen-Zuweisung durchgeführt.

In Zeile 120 wird „WorkerSupportsCancellation“ auf „False“ gesetzt. Die entwickelte Software besitzt die Möglichkeit, zu jedem Zeitpunkt abgebrochen und gestoppt zu werden. Dafür wurde ein Objekt vom Typ Backroundworker eingeführt, der bei einem Wechsel auf „True“ in der nächsten Überprüfung der Bedingung zum Ende der Software (Testprogram) springt und den laufenden Prozess stoppt. Damit der Prozessabbruch erkannt werden kann, muss das Testprogramm über den Backroundworker gestartet werden.

Außerdem wird nach der Auslösung des Start-Buttons der Button Start deaktiviert und kann nach dem Start nicht mehr gedrückt werden.

8.2.2 Stop-Button

Durch Auslösung des Stop-Buttons wird der Testvorgang direkt gestoppt und die gesperrten Buttons wieder freigegebenen. Das geschieht, indem der Backroundworker auf „False“ gesetzt wird und bei der nächsten Bedingung den Testvorgang umgehend stoppt.

8.2.3 Drop-Down-Menu Testprogramauswahl

Das Drop-Down-Menu wurde, wie alle anderen Elemente der Oberfläche, aus dem Standardcontainer der Toolbox in die Oberfläche hinzugefügt. Die Auswahlliste wird durch die Funktion „InserItemsInComboBox“ gefüllt. In Abbildung 50 ist das Füllen der Liste mit Hilfe der Funktion „.Items.Insert(index as Integer, Name as String)“ dargestellt. Damit die Menüpunkte in einer bestimmten Reihenfolge im Drop-Down-Menu gelistet werden, muss jedem Element ein Index zugewiesen werden. Dieser Index bezieht sich auf die Position, an der das Listenelement aufgeführt wird.

```
protected void InserItemsInComboBox()
{
    comboBox1.Items.Insert(0, "CP_Char_statisch");
    comboBox1.Items.Insert(1, "CP_Char_dynamisch");
    comboBox1.Items.Insert(2, "LDO_Line_statisch");
    comboBox1.Items.Insert(3, "Test RBB Mask");
    comboBox1.Items.Insert(4, "Voltage Scaling");
    comboBox1.Items.Insert(5, "Test FH - Voltage Scaling RBB/FBB");
    comboBox1.Items.Insert(6, "RBB/FBB Messung");
}// Liste für den eintrag ins Drop-Down-Menu
```

Abbildung 50: Zuweisen der Einträge in das Drop-Down-Menu

Die Testprogramme werden in Kapitel 10 spezifiziert und erklärt. Daher wird hier nur gezeigt, wie die Elemente in die Liste eingetragen werden.

8.2.4 Textfelder

Die Textfelder Name, Datum, Chip Nr. und Pfad müssen ausgefüllt werden, um das Testprogramm zu starten. Die Eintragungen werden direkt von den nötigen Funktionen wie z.B. für die Speicherung inkl. Pfad und Name der Messdaten übergeben.

8.2.5 Chart – Messdatenerfassung

Eine Übersicht des Messfortschritts wird in einem Chart dargestellt. In diesem Diagramm werden der Fortschritt und die Verteilung der Messwerte in Echtzeit dargestellt. Jeder aufgenommene Messwert wird in das Diagramm eingetragen, um Probleme bei den Messungen schnellstmöglich zu erkennen und gegebenenfalls neu zu starten. Damit das Chart mit Daten aus den Messungen gefüllt werden kann, wurde die Funktion „UpdateChart()“ und „AddNewCurve(Name as String)“ geschrieben.

```
122         public void AddNewCurve(string name)
123         {
124             this.chart1.Series.Add(name);
125             this.chart1.Name = name;
126             this.chart1.Series[name].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
127
128         }//Hinzufügen von neuem Graph
```

Abbildung 51: Hinzufügen eines Graphens in das Chart

Der Funktion „AddNewCurve“, dargestellt in Abbildung 51, muss ein Name der Kurve im Format „String“ übergeben werden, welcher dem Namen der dargestellten Graphen entspricht. In Zeile 124 wird über das Objekt „chart1“ eine Funktion „.Series.Add“ aufgerufen, in der der übergebene Name hinzugefügt wird. Danach wird in Zeile 125 mit „.Name“ der Name für die Legende des Graphen

```
102         public void UpdateChart(Object sender, MeasurementStatusEventArgs e)
103         {
104             if (IsHandleCreated == true)
105             {
106                 chart1.Invoke((MethodInvoker)delegate
107                 {
108
109                     foreach (var curve in e.Data)
110                     {
111                         if (!chart1.Series.Any(s => s.Name == curve.Name))
112                         {
113                             AddNewCurve(curve.Name);
114                         }
115                         this.chart1.Series[curve.Name].Points.AddXY(curve.DataX, curve.DataY);
116                     }
117                     this.chart1.Invalidate();
118
119                 });
120             }
121         }// Funktion zum eintragen der Daten in den Plot
```

Abbildung 52: Aktualisierung des Charts zur Visualisierung der aktuellen aufgenommen Messergebnisse

gesetzt. In Zeile 126 wird der Chart-Typ gesetzt, welcher in diesem Fall einem Liniendiagramm entspricht.

Mit „UpdateChart()“, dargestellt in Abbildung 52, werden die angefallenen Messdaten als X- und Y-Koordinate in das Diagramm übertragen. In Zeile 104 wird überprüft, ob das GUI schon erstellt wurde. Wurde es nicht erstellt, trägt diese Funktion auch keine Daten in das Diagramm ein. Danach wird jedes Datenelement durchsucht und geschaut, ob es einen neuen Graphen mit einem neuen Namen gibt. Ein neu erkannter Graph wird über „AddNewCurve“ hinzugefügt. Anschließend werden in Zeile 115 die Daten für die X- und Y-Werte in das Diagramm eingetragen. Nach jedem Eintrag muss das Diagramm neu gezeichnet werden. Dazu dient die Funktion „Invalidate()“. Diese Funktion bewirkt, dass das alte Chart für ungültig erklärt und das Diagramm neu gezeichnet wird. Ohne diese Funktion würden die Daten zwar eingetragen aber visuell kein Fortschritt erkennbar sein.

8.3 Implementierung der Hardware

In diesem Abschnitt werden die implementierten Funktionen zur Steuerung der im Testablauf eingesetzten Messinstrumente vorgestellt. Alle Messinstrumente werden über die GPIB-Schnittstelle betrieben, wodurch jedem Instrument derselbe Kommunikationsaufbau zugrunde liegt. Die Initialisierung der GPIB Kommunikation wird jedoch an dieser Stelle nur einmalig beschrieben, um redundante Inhalte zu vermeiden.

8.3.1 GPIB Kommunikation

```
21          public KeithleySrc2450(int gpibAddress)
22          {
23              ResourceManager rMgr = new ResourceManager();
24              dev = new FormattedIO488();
25              dev.IO = (IMessage)rMgr.Open("GPIB::" + gpibAddress, AccessMode.NO_LOCK, 2000, string.Empty);
26
27          }
```

Abbildung 53: Aufbau der Kommunikation mit GPIB

Die Initialisierung der Kommunikation erfolgt in der von den Instrumentenherstellern und der IVI-VISA.NET Treiber vorgegeben Weise und ist in Abbildung 53 dargestellt. Hierfür wird zunächst über „using Ivi.Visa.Interop;“ die Interop-Library eingebunden. Als nächstes wird ein Objekt von „FormattedIO488()“ und dem „ResourceManager()“ erstellt. FormattedIO488() enthält Funktionen zum Lesen und Schreiben im Format des IEEE488-Kommunikationsprotokolls. Über „ResourceManager.Open()“ wird die GPIB-Verbindung zum Instrument aufgebaut. Dafür muss das Keyword „GPIB::“ und die Instrumenten-Adresse initialisiert werden. Durch das Argument „AccessMode.NO_LOCK“ wird das Instrument nicht gesperrt und reagiert weiterhin auf Eingaben und Änderungen über die Gerätetasten selbst. Das folgende Argument gibt die „IO.Timeout“ Zeit an, innerhalb derer das Instrument reagieren muss. Reagiert das Instrument nach den 2000 ms nicht, ist der Verbindungsaufbau fehlgeschlagen, und das Gerät wurde nicht initialisiert. Zeitgleich wird auch ein Runtime Error ausgelöst, der den Nutzer über den fehlgeschlagenen Verbindungsaufbau informiert.

Die Kommunikation über die aufgebaute Verbindung läuft eventbasiert und wird über das Interface „IMessage“ getriggert. Der Funktionsumfang der Interfaceklassen wird nicht weiter erläutert. Nähere Informationen zu den einzelnen Events, sind in Quelle [21] ab Kapitel 9 sehr detailliert erklärt.

Für die GPIB-Kommunikation wurden noch einige Support-Funktionen implementiert, die in jeder Klasse der Geräte zum Schreiben und Lesen verwendet werden. [22]

8.3.2 Funktionen für das Keithley 2000 Multimeter

Für das Keithley 2000 Multimeter wurden nur die Funktionen, die auch wirklich benutzt werden, implementiert. Es wird in Tabelle 1 eine Funktionsübersicht aller entwickelten Funktionen gezeigt. Da IEEE488.2 zum Steuern der Instrumente nur einen String über die Kommunikationsstrecke sendet, wird in der Spalte „GPIB-Kommando“ der zu sendende String aufgeführt. Die häufigste verwendete Funktion ist die Messung der aktuellen Spannung- oder des aktuellen Stromes am Eingang des Instruments. Des Weiteren kann der Messbereich für die jeweilige Messung über die Software gesetzt werden. Dieser sollte für eine bessere Vergleichbarkeit der Messungen bei jedem Gerät und jeder Messung identisch sein. Ein anderer Messbereich ruft einen anderen Innenwiderstand hervor, wodurch sich die Messwerte leicht unterscheiden können.

Tabelle 1: Keithley 2000 implementierte Funktionen

Funktion	Argumente	GPIB-Kommando	Beschreibung
Keithley2000Multimeter()	Int gpibAdress	Siehe Kap. GPIB Kommunikation	Initialisierung und Verbindungsaufbau für Keithley2000
Dispose()	-	Dev.IO.Close (VISA Funktion)	Dient zur Trennung der Verbindung und Löschung der Objektinstanz. Wird über die Funktion .Close() aus der VISA Library durchgeführt.
Reset()	-	*RST	Sendet den Befehls als String „*RST“.
ID()	-	*ID?	Sendet den Befehl als String „*ID?“. Da dieser Befehl ein Query ist, wird innerhalb der Funktion mit Read() das Resultat des Befehls ausgelesen.
GetVoltage()	-	*:READ?	Liest die aktuelle Spannung am Eingang des Multimeters aus
GetCurrent()	-	*:READ?	Liest den aktuellen Strom am Eingang des Multimeters aus
SetCurrAutoRange()	String OnOrOff	":SENS:CURR:DC:RANG:AUTO "+ OnOrOff	Aktiviert oder deaktiviert die Autorange Funktionalität während einer Strommessung
SetVoltAutoRange()	String OnOrOff	":SENS:VOLT:DC:RANG:AUTO "+ OnOrOff	Aktiviert oder deaktiviert die Autorange Funktionalität während einer Strommessung
SetCurrRange()	Int range	":SENS:CURR:DC:RANG " + range	Gibt die Möglichkeit, den Strommessbereich manuell

			festzulegen und vor einer Umschaltung zu schützen
SetVoltRange()	Int range	":SENS:VOLT:DC:RANG " + range	Gibt die Möglichkeit den Spannungsmessbereich, manuell festzulegen und vor einer automatischen Umschaltung zu schützen

8.3.3 Funktionen für das Keithley 2450 Sourcemeter

In der Tabelle 2 sind alle für das Keithley 2450 Sourcemeter implementierten Funktionen aufgelistet. Die häufigste verwendete Funktion ist mit Abstand die Messung der Spannung mit GetVoltage() und das Setzten der Spannung mit SetSourceVoltage(). Es ist darauf zu achten, dass nach der Konfiguration der Instrumentenausgang mit „SetOutputOn(„ON“)“ freigegeben wird. Wenn der Ausgang sich im Zustand „OFF“ befindet, wird weder eine Spannung noch ein Strom von der Quelle ausgegeben.

Tabelle 2: Keithley 2450 implementierte Funktionen

Funktion	Argumente	GPIB-Kommando	Beschreibung
Keithley2450Sourcemeter()	Int gpibAdress	Siehe Kap. GPIB Kommunikation	Initialisierung und Verbindungsaufbau über GPIB für das Keithley2450 Sourcemeter.
Dispose()	-	Dev.IO.Close (VISA Funktion)	Dient zur Trennung der Verbindung und Löschung der Objektinstanz. Wird über die Funktion .Close() aus der VISA Library durchgeführt.
Reset()	-	*RST	Sendet den Befehls als String „*RST“.
ID()	-	*ID?	Sendet den Befehl als String „*ID?“. Da dieser Befehl ein Query ist, wird innerhalb der Funktion mit Read() das Resultat des Befehls ausgelesen.
GetVoltage()	-	*:READ?	Liest die aktuelle Spannung am Eingang des Multimeter aus.
GetCurrent()	-	*:READ?	Liest den aktuellen Strom am Eingang des Multimeters aus.
GetRes()	-	*:READ?	Misst den aktuellen ohmschen Widerstand.
SetMeasureType()	String VoltOrCurrOrRes	"SENS:FUNC " + VoltOrCurrOrRes	Mit VOTL, CURR, RES wird ausgewählt, welcher Größe gemessen werden soll.
SetCurrAutoRange()	String OnOrOff	":SENS:CURR:DC:RANG:AUTO "+ OnOrOff	Aktiviert oder deaktiviert die Autorange Funktionalität während der Strommessung
SetVoltAutoRange()	String OnOrOff	":SENS:VOLT:DC:RANG:AUTO "+ OnOrOff	Aktiviert oder deaktiviert Autorange während einer Spannungsmessung.
SetCurrRange()	Int range	":SENS:CURR:DC:RANG " + range	Gibt die Möglichkeit, den Strommessbereich manuell festzulegen und vor einer automatischem Umschaltung zu schützen.
SetVoltRange()	Int range	":SENS:VOLT:DC:RANG " + range	Gibt die Möglichkeit, den Spannungsmessbereich manuell

			festzulegen und vor automatischem Umschalten zu schützen.
SetOutputOn()	String OutputOnorOoff	":OUTP " + OutputOnOrOff	Schaltet den Ausgang des Sourcemeters An oder Aus.
SetSourceVoltage()	String Voltage	1. ":SOUR:FUNC VOLT " 2. ":SOUR:VOLT " + Voltage	1. Mit diesem Befehl wird die Source-Funktion auf Volt gesetzt. Um Fehler zu vermeiden, wird dieser Befehl bei jedem Setzen der Spannung ausgeführt 2. Mit diesem Befehl wird die Spannung gesetzt.
SetSourceCurrent()	String Current	1. ":SOUR:FUNC CURR " 2. ":SOUR:CURR " + Current	1. Mit diesem Befehl wird die Source-Funktion auf Current gesetzt. Um Fehler zu vermeiden, wird dieser bei jedem Setzten des Stroms ausgeführt 2. Mit diesem Befehl wird der Strom gesetzt.
SetMeasVolt()	-	":SENS:FUNC \"VOLT\" "	Stellt den Messkanal auf Spannungsmessung ein
SetMeasCurr()	-	":SENS:FUNC \"CURR\" "	Stellt den Messkanal auf Strommessung ein
SetMeasRes()	-	":SENS:FUNC \"RES\" "	Stellt den Messkanal auf Widerstandsmessung ein

8.3.4 Funktionen für das R&S HMC8043

In der Tabelle 3 sind die implementierten Funktionen für die Spannungsversorgung von Rhode&Schwarz HMC8043 aufgelistet. Die am häufigsten verwendeten Funktionen sind mit Abstand GetVoltage(), GetCurrent() und SetMaster(). Da das Netzteil drei unterschiedliche Kanäle besitzt, muss auf die korrekte Handhabung der Quelle geachtet werden. Es gibt eine zweistufige Hierarchie, die unbedingt beachtet werden muss. Jeder der drei Kanäle kann „an“ und „aus“ geschaltet werden. Die Quelle gibt die Kanäle jedoch erst frei, wenn mit SetMaster() der Masteroutput auf „an“ gesetzt worden ist.

Tabelle 3: Rhode und Schwarz HMC8043 implementierte Funktionen

Funktion	Argumente	GPIB-Kommando	Beschreibung
RhodeSchwarzPwrSupInit ()	Int gpibAdress	Siehe Kap. GPIB Kommunikatio	Initialisierung und Verbindungsaufbau über GPIB für die Powersupply R&S HMC8043.
Dispose()	-	Dev.IO.Close (VISA Funktion)	Dient zur Trennung der Verbindung und Löschen der Objektinstanz. Wird über die Funktion .Close() aus der VISA Library durchgeführt.
Reset()	-	*RST	Sendet den Befehls als String „*RST“.
ID()	-	*ID?	Sendet den Befehl als String „*ID?“. Da dieser Befehl ein Query ist, wird innerhalb der Funktion mit Read() das Resultat des Befehls ausgelesen.
GetVoltage()	Int channel	1. SetChannel(channel) 2."MEAS:VOLT:DC?"	1. Definiert den Kanal, bei dem die Spannung gemessen werden soll

			2.Liest die aktuelle Spannung am Eingang vom Multimeter aus.
GetCurrent()	Int channel	1. SetChannel(channel) 2."MEAS:CURR:DC?"	1. Setzt den Kanal, bei dem der Strom gemessen werden soll. 2.Liest den aktuellen Strom am Eingang des Multimeters aus.
SetMaster()	String state	"OUTP:MAST "+ state	Mit „ON“ wird die Quelle freigegeben, mit „OFF“ wird sie getrennt
SetVoltage()	String Voltage String Channel String OutputChanOnOff	1. "INST " + channel 2. "VOLT " + Voltage 3. "OUTP:CHAN " + OutputChanOnOff	1. Wählt den Kanal aus 2. Setzt die Spannung an den ausgewählten Kanal. 3. Setzt den Kanal auf „ON“ oder „OFF“
SetChannelOn()	String channel String OnOrOff	1. "INST " + channel 2. "OUTP:CHAN " + OnOrOff	1. Wählt den Kanal aus. 2. Schaltet den Kanal an oder aus mit „ON“ oder „OFF“
SetChannel()	String channel	"INST OUT"+channel	Wählt den Kanal des Instrumentes aus. Diese Funktion wird in Kombination mit GetVoltage und GetCurrent verwendet.

8.3.5 Temptronic ATS 710-M Thermostream

Der Thermostream wird ebenfalls über die GPIB-Schnittstelle gesteuert. In der Tabelle 4 sind die implementierten Funktionen aufgelistet. Die meistverwendeten Funktionen sind SetTemperature(), AtTemperature() und SetIdle(). Es ist darauf zu achten, dass vor der Konfiguration das Display auf den „Cycle“-Screen gesetzt werden muss. Solange der Thermostream sich nicht auf diesem Display befindet, ist es nicht möglich, die Konfiguration des Thermostreams zu ändern und ihn über GPIB zu steuern. Des Weiteren unterscheidet sich die Formatierung der GPIB Kommandos trotz der Verwendung des Standards IEEE488 von den anderen Geräten. Nur die Grundkommandos sind nach dem Standard erstellt worden, während alle gerätespezifischen Kommandos von dem Standard abweichen.

Tabelle 4: Temptronic ATS 710-M implementierte Funktionen

Funktion	Argumente	GPIB-Kommando	Beschreibung
ThermostreamATS710Init	Int gpibAdress	Siehe Kap. GPIB Kommunikation	Initialisierung und Verbindungsaufbau über GPIB für den Thermostream ATS710-M.
Dispose()	-	Dev.IO.Close (VISA Funktion)	Dient zur Trennung der Verbindung und Löschung der Objektinstanz. Wird über die Funktion .Close() aus der VISA Library durchgeführt.
Reset()	-	*RST	Sendet den Befehls als String „*RST“.
ID()	-	*ID?	Sendet den Befehl als String „*ID?“. Da dieser Befehl ein Query ist, wird innerhalb der Funktion mit Read() das Resultat des Befehls ausgelesen.

GetTemp()	-	TEMP?	Dient zum Auslesen der aktuellen Temperatur
SetFlow()	Int FlowOn	FLOW +FlowOn	Dient zum an- und ausschalten des Flow, 1= an und 0 = aus
GetHeadPosition()	-	HEAD?	Fragt die Position der Temperaturkammer ab, ob sie oben oder unten am DUT ist.
AtTemperature	-	Boolean Property TECR?	Überprüft, ob der Thermostream auf seiner Temperatur hochgefahren ist. Diese Eigenschaft liefert ein: „FALSE“ für Temperatur nicht erreicht „TRUE“ für Temperatur erreicht, zurück
SetIdle()	-	1. *RST 2. *CLE 3. HEAD 0 4. COOL 0	Mit dieser Funktion wird der Thermostream in seinen Idle Zustand versetzt. Es wird ein Reset ausgeführt und der Buffer gelöscht. Danach wird der Head vom DUT getrennt und der Kompressor mit COOL 0 ausgeschaltet.
SetDisplayToCycleScreen()	-	SRST	Setzt die Displayanzeige des Thermostreams auf das „Cycle“ Screen. Wenn sich der Thermostream nicht auf diesem Screen befindet, ist keine Steuerung über GPIB möglich.
SetTemperature	Double Temperature Double rampRate Double soakTime String tempWindow	"SETP " + temperature + "; " + "RAMP " + rampRate + "; " + "SOAK " + soakTime + "; " + "WNDW " + tempWindow + ";", 100	1. Überprüft die Head Position. Wenn er nicht unten am DUT ist, wird er heruntergefahren 2. Überprüft ob die eingestellte Temperatur erreicht wurde. Wenn sie überschritten wurde, wird der Flow abgeschaltet. 3. Setzt über das GPIB-Kommando die Werte für: Temperatur (temperature) Zeit der Rampe (rampRate) Vorlaufzeit (soakTime) Temperaturabweichung (tempWindow) Wartezeit (=100) 4. Wenn all diese Schritte durchgelaufen sind, wird ein „TRUE“ von der Funktion zurückgegeben

8.4 Teststeuerung

8.4.1 Maskierung des Datenstream

Die Konfiguration des DUTs wird über ein SPI-Interface durchgeführt. Damit die Konfiguration einwandfrei funktioniert, musste ein Mechanismus zum Schreiben der Registerkonfiguration entwickelt werden.

Im TC1.5 muss zuerst das Zielregister und dann das Datenwort für die Konfigurationsregister übertragen werden. Für die Charakterisierung werden die Adressen 0x180000080 (Register zum Setzen der RBB Settings) und 0x180000001 (Register zur Aktivierung und Deaktivierung der internen Reset-Strukturen) verwendet.

Als zweiter Teil der Kommunikation müssen die Konfigurationsbits übertragen werden. Jede Konfiguration kann aus unterschiedlich langen Bitkombinationen bestehen. Jedes Register besitzt aber seinen Reset-Wert, um das DUT wieder zurück zu setzten. Die Konfiguration der verwendeten Register ist im Code in Abbildung 54 zusehen.

```
TrimRegisterBitFields.RbbEn, Length = 1, Position = 0, Mask = 1, ResetValue = 0 });
TrimRegisterBitFields.Switch, Length = 3, Position = 1, Mask = 7, ResetValue = 5 });
TrimRegisterBitFields.RbbTest, Length = 1, Position = 4, Mask = 1, ResetValue = 0 });
TrimRegisterBitFields.Cpen, Length = 1, Position = 10, Mask = 1, ResetValue = 0 });
TrimRegisterBitFields.CpTrim, Length = 3, Position = 24, Mask = 7, ResetValue = 0 });
TrimRegisterBitFields.LdoTrim, Length = 8, Position = 16, Mask = 255, ResetValue = 0 });
TrimRegisterBitFields.Rst1v3OFF, Length = 1, Position = 24, Mask = 1, ResetValue = 0 });
TrimRegisterBitFields.Rst3V3OFF, Length = 1, Position = 26, Mask = 1, ResetValue = 0 });
TrimRegisterBitFields.Rst5VOFF, Length = 1, Position = 28, Mask = 1, ResetValue = 0 });
```

Abbildung 54: Konfigurationsregister für RBB

Jedes Konfigurationsregister kann unabhängig von den anderen gesetzt werden. Anhand der Position wird festgelegt, um wie viele Stellen die Bitkombination geschoben werden muss. Die Variable „Value“ wird zur Maskierung des Datenstreams und zum Zurücksetzen des DUTs verwendet.

Damit die einzelnen Konfigurationen zu einem Datenwort zusammengefasst werden, wird zuerst eine invertierte Maske des Registerinhaltes gebildet, um die benötigten Stellen im Datenwort auf „0“ zu setzen. Dafür wird der höchstmögliche Wert der Bitkombination, welcher unter „Mask“ in den Bitfeldsettings hinterlegt ist, zur benötigten Position im Datenwort verschoben. Danach wird die invertierte Maske mit der Resetmaske über einen „AND“ Operator verknüpft. Anschließend wird die

erhaltene Maske mit der gesetzten Konfiguration durch ein logisches „OR“ verknüpft. Durch diese Vorgehensweise werden im Datenwort nur die Stellen verändert, welche die jeweilige Konfiguration betreffen. In Abbildung 54 ist die implementierte Funktion zur Maskierung gezeigt.

8.4.2 Testentwicklung

Die Testentwicklung definiert einen allgemein gültigen Rahmen, der bei jeder Messung durchlaufen wird. Darunter zählt die Geräteinitialisierung, die Konfiguration des DUTs, die Darstellung der Messwerte und die Erstellung eines Messprotokolls. Die Initialisierung der Messinstrumente, sowie die Konfiguration des DUTs können den jeweiligen Testspezifikationen entnommen werden.

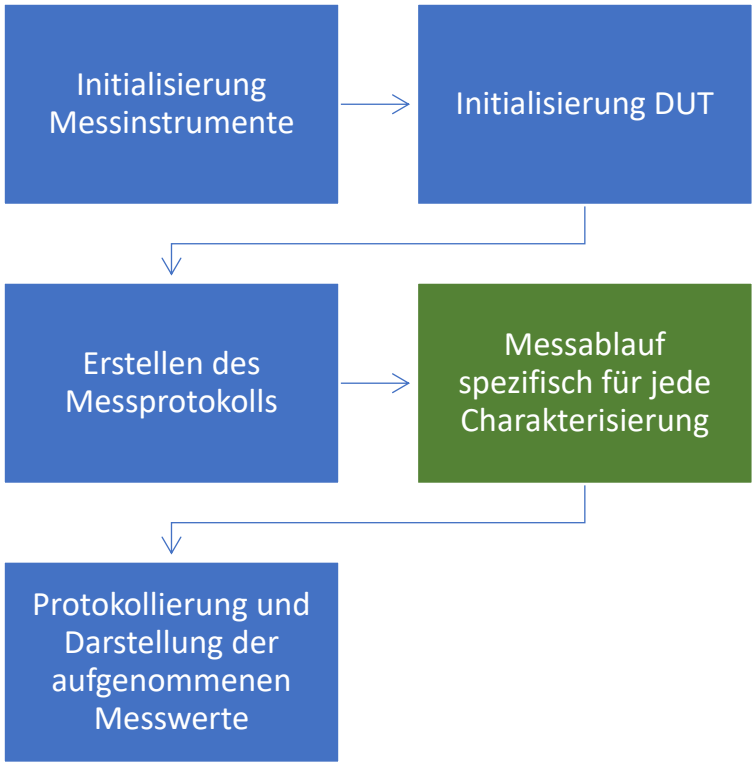


Abbildung 55: Allgemeingültiger Rahmen jeder Charakterisierungsmessung

In der Abbildung 55 sind in blau die redundanten Teile jeder Messung gezeigt. In grün ist der Teil gezeigt, in dem der eigentliche Messablauf definiert ist. Dieser Ablauf ist spezifisch für jede durchgeführte Charakterisierung.

8.5 Messaufbau zur Charakterisierung

Der Messplatz wurde im Labor der Infineon Austria AG in Villach aufgebaut und setzt sich, wie in Abbildung 56 gezeigt, zusammen. Der Messaufbau benötigt zur Versorgung zwei Spannungsquellen mit je drei Kanälen für das DUT und den SPI-Dongle. Für die Versorgung des Messaufbaus werden fünf Kanäle benötigt. In diesem Aufbau werden die Spannungsquellen HMC8043 von Rhode & Schwarz

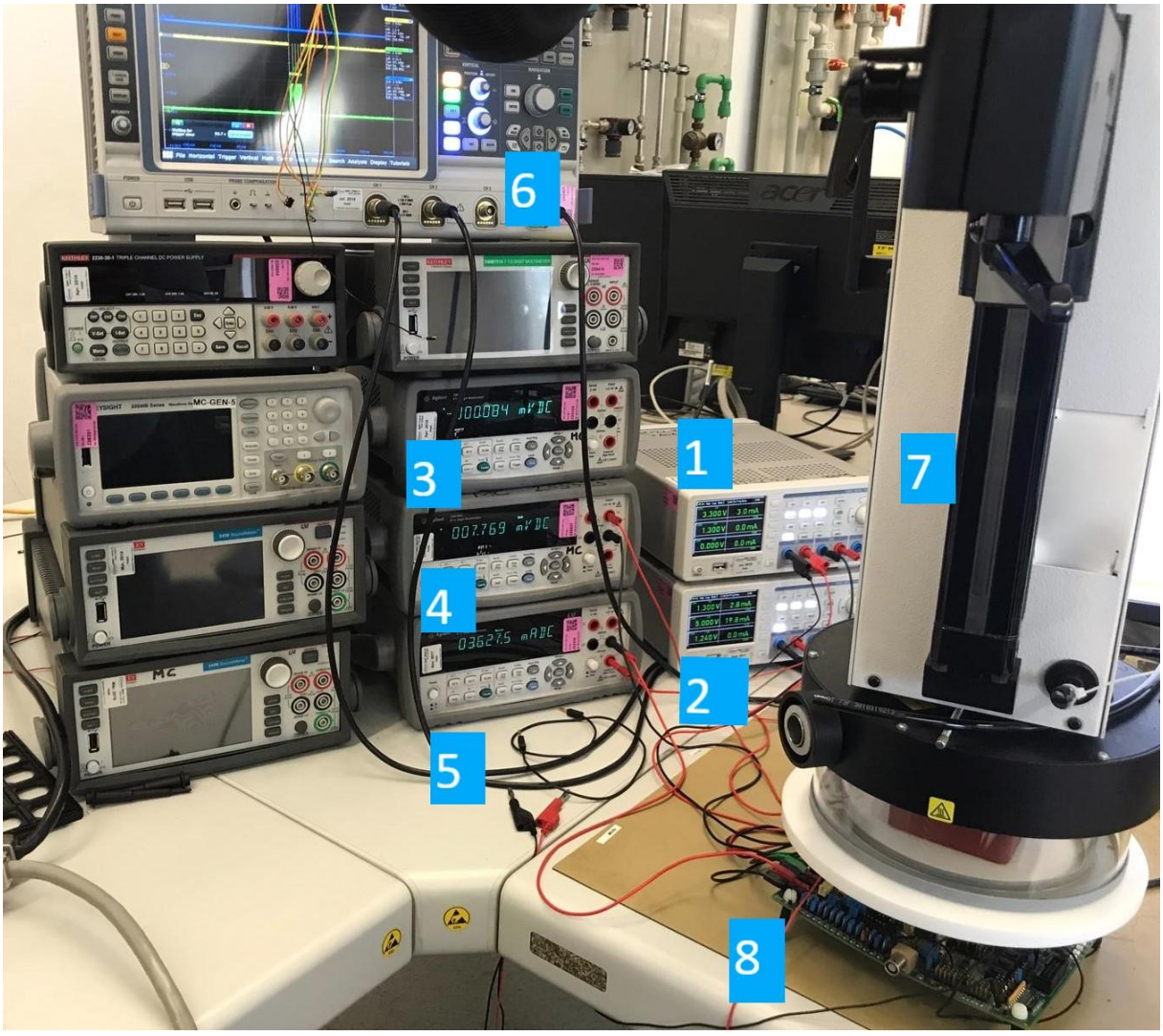


Abbildung 56: Messaufbau im Testlabor in Villach

eingesetzt. Sie sind in Abbildung 56 mit (1) und (2) gekennzeichnet und dienen zur Spannungsversorgung für VDD (5V), Vnwell (1,3V), VDDC (1,3V), V1V24 (1,24V) und Vdongle (5V). Desweiteren werden drei Multimeter genutzt, die entweder zur Strommessung oder zur Spannungsmessung eingesetzt werden. Mit (3) und (4) werden die Spannungen von VDDC und Vnwell gemessen. Mit dem Multimeter (5) wird der Strom IDDC gemessen.

Mit dem Oszilloskop (6) wird die DUT-Kommunikation kontrolliert. Im Vordergrund befindet sich das Entwicklungsboard (8) mit dem eingesetzten Testchip, auf dem der sogenannte „Head“ (7) des Thermostreams aufliegt. Durch diesen „Head“ tritt ein temperierter und getrockneter Luftstrom aus und kühlt oder erwärmt den Testchip. Wenn bei niedrigen Temperaturen gemessen wird, kondensiert jedoch trotz allem Feuchtigkeit aus der Umgebungsluft an dem Entwicklungsboard und birgt die Gefahr eines Kurzschlusses. Daher sollte nach jeder Messung bei niedrigen Temperaturen ein Trocknungsprozess in einem Ofen oder mit dem Thermostream durchgeführt werden. Dadurch wird gewährleistet, dass das PCB vollständig trocknet und die Lebensdauer nicht beeinflusst wird.

9 Datenanalyse mit Python (Post Processing)

Die Datenanalyse wird mit einem Python-Skript durchgeführt. Dieses Skript liest alle Messdaten ein, sortiert sie und erstellt alle benötigten Diagramme aus den jeweiligen Datensätzen. Da die Datensätze sich von Messung zu Messung unterscheiden, kommen verschiede Sortierungen zum Einsatz, die jedoch immer nach der gleichen Methodik durchgeführt werden. Das Post Processing wird anhand der Ordnerstruktur und der Messdaten von den Charakterisierungen der differentiellen Spannungsskalierung mit RBB und FBB gezeigt.

9.1 Einlesen der Daten

Die Messdaten befinden sich in unterschiedlichen Ordnern, die zum Einlesen durchlaufen werden müssen. Da der Pfad und die Ordnerstruktur bekannt sind, wird die Ordnerstruktur in einem Array als Konstante definiert und mittels for-Schleifen prozessiert. Die Ordner Struktur sieht wie folgt aus (Abbildung 57):

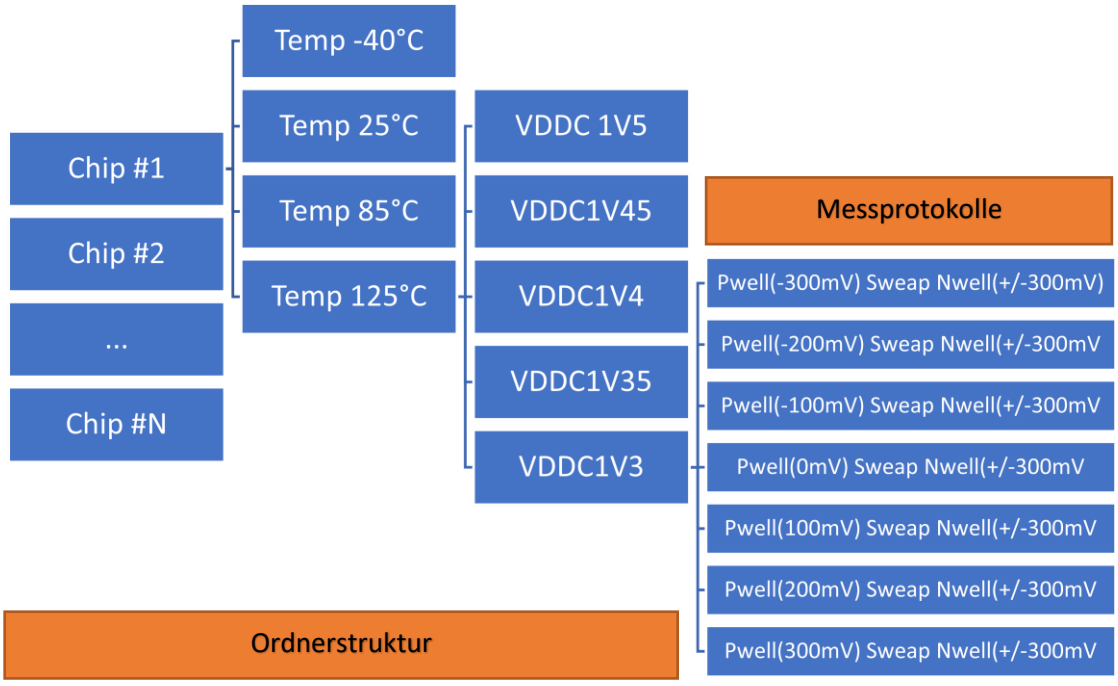


Abbildung 57: Beispiel einer Datensatzsortierung nach einer Temperatur von 125°C und 1,35V VDDC (erster Wert VDDC)

Der Durchlauf der for-Schleifen ergibt eine neue Pfadkombination und führt zur hierarchischen Abarbeitung der Ordnerstruktur. In den jeweiligen Zielordnern werden alle vorhanden Messdateien in das Array eingelesen, und als zusätzliche Information wird der Name der Messdatei dem Array angehangen. Der Name der Messdatei entspricht zugleich auch dem Namen des Diagramms. Wenn ein Diagramm aus mehreren Quellen besteht, werden feste Namen hinterlegt, die bei der Generierung verwendet werden. In jedem Zielordner werden alle Dateien eingelesen und dem Array angehangen. Beim Einlesen der Dateien musste jedoch darauf geachtet werden, dass die erste Zeile ausgelassen

wird, da es sich hierbei um den Datei-Header handelt. Außerdem werden alle eingelesenen Daten vom Datentyp String in den Datentyp Float umgewandelt. Die anschießende Darstellung der Daten benötigt floating point Zahlen, sodass diese Konvertierung unumgänglich ist.

Zum Einlesen und Verarbeitung der Daten werden folgende import‘s benötigt:

- Os
    - Ist zuständig, um eine Datei zu öffnen
- Shutil
    - Enthält Funktionen zum Kopieren, Erstellen, Verschieben von Dateien/Ordnern
- Csv
    - Dient speziell zum Öffnen, Lesen und Bearbeiten von *.csv Dateien
- Numpy
    - Ist eine spezielle Arraystruktur die zusätzliche Manipulationen, Funktionen und mathematische Operatoren enthält.
    - Wird auch für das Post-Processing benötigt

9.2 Post Processing

Beim Post Processing werden die eingelesenen Daten aus dem Array verarbeitet und sortiert. Dadurch werden Datensätze für die Diagrammausgabe vorbereitet und im Speicher abgelegt. Diese können dann bei der Erstellung der Diagramme sofort verwendet werden, wodurch die Darstellung der Messdaten deutlich vereinfacht wird. Anschließend werden die erstellten Datensätze an die Diagrammfunktionen übergeben. Die Sortierungskriterien lehnen sich auch gleichzeitig an die vorgegebene Ordnerstruktur an, die in Abbildung 57 dargestellt ist:

1. Chip
2. Temperatur
3. Core-Spannung, VDDC
4. Vpwell Spannung (Trimmwert)
5. Sweap von Vnwell (bei konstantem Vpwell)

Nach diesen fünf Kriterien werden die Datensätze sortiert und in einem mehrdimensionalen Array hinterlegt. Dadurch können verschachtelte for-Schleifen die Daten durchlaufen und der Funktion zum Erstellen der Diagramme übergeben werden. Da gleichzeitig alle Daten im Array hinterlegt wurden, können direkt Vergleiche zwischen den Messdaten gezogen werden. Ein Beispiel wäre, dass eine identische Messung bei allen drei Temperaturen verglichen wird, um den Temperaturdrift darzustellen

und zu analysieren. Jedes einzelne Kriterium fügt außerdem dem Array eine Dimension hinzu. Daraus resultiert ein fünfdimensionales Array. In der nachfolgenden Abbildung 58 wird ein Beispiel mit

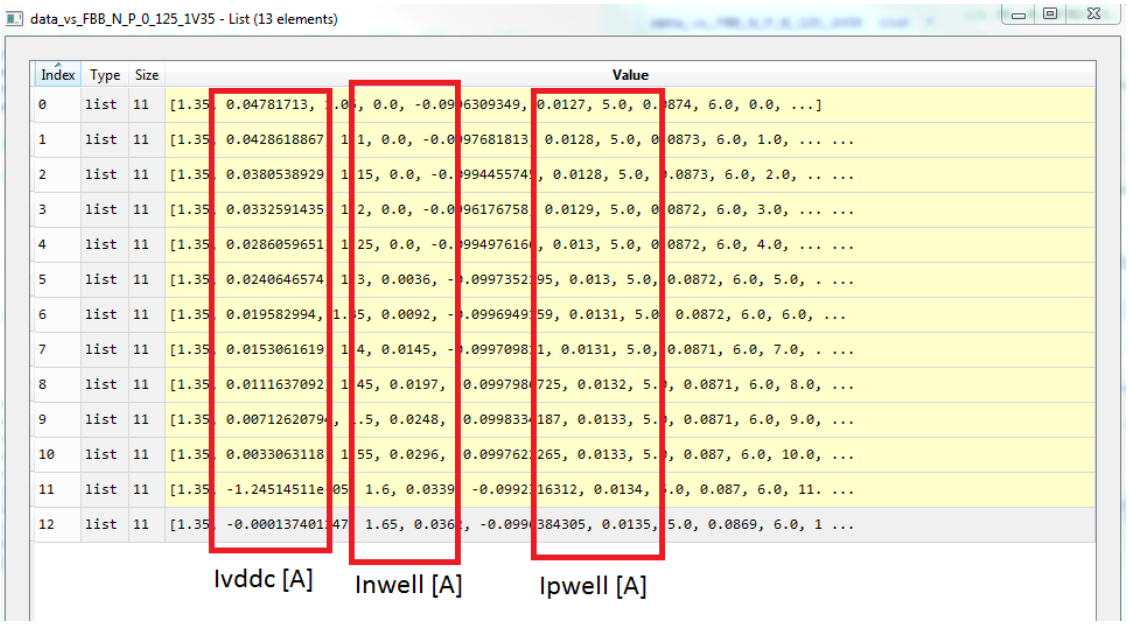


Abbildung 58: Ein aufgenommener Datensatz bei der Temperatur von 125°C und 1,35V VDDC (erster Wert VDDC)

extrahierten Datensätzen gezeigt. Da die Datensätze von verschiedenen Parametern zur selben Zeit eingelesen und sortiert werden, ist es einfach, Vergleichsdiagramme zwischen unterschiedlichen Temperaturen oder verschiedenen VDDC Spannungen zu erstellen.

Als Beispiel wird nun ein Datensatz mit einer exemplarischen Sortierung gezeigt. Für die Sortierung der Daten wurden die folgenden Kriterien verwendet:

- Temp: 125°C
- VDDC: 1V3, 1V35, 1V4, 1V45, 1V5
- Pwell = -300mV
- Sweep von Nwell (VDDC +/-300mV)

Das daraus resultierende Diagramm ist in (Abbildung entfernt) dargestellt.

9.3 Diagrammausgabe

Die Diagramme werden mit der Library matplotlib erstellt. Diese Bibliothek bietet alle arithmetischen Funktionen und Diagramm-Elemente, die in MATLAB vorhanden sind. Daher eignet sie sich ausgezeichnet für wissenschaftliche Auswertungen und mathematische Diagramme. Zur Erstellung der Diagramme werden die verschiedenen Datensätze zu einem Diagramm zusammengefasst. Des Weiteren wird dem Diagramm eine Legende mit den einzelnen Parametern, eine Achsenbeschriftung und eine Überschrift hinzugefügt. Das Ergebnis dieser Einstellungen ist in **Fehler! Verweisquelle konnte nicht gefunden werden.**. zu sehen. (Diese Abbildung unterliegt der Geheimhaltung)

9.4 Entwickelte Skripte zur Auswertung

Aufgrund der Komplexität der einzelnen Messungen und unterschiedlichen Ordnerstrukturen, ist für jede Charakterisierung ein eigenes Auswertungsskript erstellt worden. Durch entsprechende Optimierungen konnte auch der Umfang einiger Quellcodes reduziert werden, obwohl die Verarbeitung der Daten in den verschiedenen Skripten im Kern gleich bleibt. In der folgenden Aufzählung sind die entwickelten Python Skripte aufgelistet:

- Auswertungsskript für Spannungsregulierung LDO
    - (Dateiname: Plotting_LDO.py)
- Auswertungsskript für Spannungs- und Stromregulierung CP
    - (Dateiname: Plotting_CP.py)
- Auswertungsskript für Reverse and Forward Body Biasing (statisch / dynamisch)
    - (Dateiname: Plotting_RBBFBB_DVS.py)
- Auswertungsskript für Differentielle Spannungs-Skalierung (statisch / dynamisch)
    - (Dateiname: Plotting_RBBFBB_DVS.py)

10 Testspezifikation und Ablauf

Dieses Kapitel unterliegt der Geheimhaltung.

11 Auswertung der Messergebnisse

Dieses Kapitel unterliegt der Geheimhaltung.

12 Zusammenfassung der Messergebnisse

Dieses Kapitel unterliegt der Geheimhaltung.

13 Ausblick

Der Ausblick auf zukünftige Schritte ergibt sich aus den erhaltenen Ergebnissen und den aufgetretenen Problemen. Sowohl bei der Charakterisierung der Schaltung für die Spannungsversorgung als auch bei der Aufnahme der Messreihen für RBB und FBB mit DVS sind instabile Zustände aufgetreten, die nicht verhindert werden konnten. Zu Beginn wurde die Nutzung des DUTs für die Überprüfung der Stabilität einer CMOS Schaltung bei Anwendung von RBB und FBB als Ziel definiert. Das implementierte DUT konnte jedoch nie in Betrieb genommen werden, da es sich nicht in den Zustand, der für die Ausführung der Analyse nötig gewesen wäre, bringen ließ. Das DUT wurde mit einem FPGA-Board verbunden und die Routinen zur Steuerung mithilfe von VHDL in die Logik integriert. Das DUT wies allerdings nicht das geplante Verhalten auf. Insbesondere wurde das DUT bei Auftreten eines Fehlers nicht angehalten bzw. Fehler wurden nicht signalisiert.

Daher wäre der erste Schritt einen neuen Testchip zu entwickeln und ein Redesign des DUT durchzuführen. Dadurch könnten die gewünschten Stabilitätstests durchgeführt und untersucht werden, ob es sich um systematische oder willkürlich auftretende Fehler in bestimmten Bereichen des DUTs handelt. Dies ist notwendig, um letztendlich eine klare Aussage über die Robustheit der Schaltung, und im speziellen über die Robustheit des Features RBB und FBB zu treffen.

Ein weiterer Ansatz sollte sein, die Ursache für das Fehlverhalten des LDO und der CP zu identifizieren, so dass eine zuverlässige Trimmung des LDOs durchgeführt werden kann. Desweiteren sollte darüber nachgedacht werden den trimmbaren Bereich des LDOs zu verändern, damit auch FBBN mit On-Chip Versorgung charakterisiert werden kann. Außerdem sollten die Probleme bei der Kommunikation mit dem LDO gelöst werden, um die ungewollte Auslösung des POR zu verhindern. Dieses Verhalten trat unregelmäßig und nicht reproduzierbar auf.

Außerdem muss das Problem der CP bei einem Trimmwert von 0 gelöst werden, um eine Spannung von 0V generieren zu können.

Auf jeden Fall sollte die SPI-Schnittstelle überarbeitet werden. Aktuell können die Register nur gesetzt aber nicht gelesen werden, was eine entsprechende Fehleranalyse deutlich erschwert. Wäre es möglich gewesen die Register auszulesen, hätten fundierte Rückschlüsse auf die Art des Resets gezogen werden können. Jeder einzelne Reset besitzt ein Register indem die Auslösung angezeigt wird. Es hätte diesem Projekt die Möglichkeit eröffnet bessere Aussagen über die Ursachen für das instabile Verhalten des Testchips zu treffen.

Trotz all dieser Schwierigkeiten kann anhand des Testchips festgestellt werden, dass durch den kombinierten Einsatz von RBB mit DVS der Stromverbrauch im Betrieb gesenkt werden kann. Diese Methodik könnte daher in zukünftigen Entwicklungen herangezogen werden, um den Stromverbrauch

zu reduzieren und Chips noch effizienter zu machen. Die Messergebnisse, die in Kapitel 12 zusammengetragen wurden, sprechen infolgedessen für die Weiterführung der Aktivitäten zu RBB.

14 Fazit

Diese Masterthesis ist bedingt durch Umfang und Länge sowie die vielen verschiedenen behandelten Themen differenziert zu betrachten. Es wurden viele Grundlagen in Bezug auf das Reverse und Forward Body Biasing, den Transistor und die eingesetzten Programmiersprachen erarbeitet. Dabei wurde eine detaillierte Funktionsbeschreibung des MOSFET erstellt, wobei der pn-Übergang mit seinen unterschiedlichen Betriebsmodi im Vordergrund stand. Durch RBB und FBB werden diese physikalischen Mechanismen des pn-Übergangs und des Body-Effekts so ausgenutzt, dass die Schwellenspannung nach Bedarf nachjustiert werden kann, um eine Schaltgeschwindigkeitserhöhung oder eine Stromreduzierung hervorzurufen.

Reverse- und Forward-Body-Biasing wurde als Powermanagement Methode in Mikrocontrollern vorgestellt und das Potenzial mit Bezug auf die Reduktion und Schaltgeschwindigkeitserhöhung betrachtet. Im Gegensatz zu anderen Optimierungsmethoden kann dieses Feature mit wenig Aufwand in Mikrocontrollern implementiert werden.

Eingeleitet wurde der Charakterisierungsprozess mit der Vorstellung des eingesetzten Testchip TC1.5, welcher bei der Infineon Austria AG entworfen wurde. In diesem Abschnitt wurde die Funktion des PMS erklärt, welches sich in viele kleine Sub-Module unterteilt. Es wurde erläutert, wie die Power-Sequencing-State-Maschine, das SCU-Interface, das TCU-Interface, die ENPS-Control und noch weitere Module funktionieren. Außerdem war das sogenannte RBB DUT auf dem Testchip als Teststruktur implementiert. Dazu wurde der interne Schaltungsaufbau, sowie die vorhandenen bzw. verwendeten Register für das RBB Feature vorgestellt. Außerdem musste eine spezielle RBB Startup-Routine durchgeführt werden, um das Device in das korrekte Setup zu bringen.

Damit die Charakterisierung durchgeführt werden konnte, wurde eine Testsoftware für die Steuerung des Labormessplatzes samt Thermostream erfolgreich entwickelt. Es wurden fundamentale Grundlagen in der Programmiersprache C# und der Skriptsprache Python vorgestellt. Es wurde die Frage geklärt, wie ein Messplatz für eine Charakterisierung aufgebaut sein muss und welche Kommunikationsprotokolle verwendet werden müssen, um ein Messinstrument über den PC steuern zu können. Durch das angeeignete Wissen wurde erfolgreich eine Testsoftware und ein automatisierter Charakterisierungsmessplatz entwickelt. Prinzipiell wäre es sogar möglich gewesen, jede Charakterisierung bei unterschiedlichen Temperaturen und verschiedenen Frequenzen vollautomatisiert durchlaufen zu lassen. In der Praxis hat sich dieses Vorgehen für die gegebene Aufgabe aber nicht als vorteilhaft herausgestellt. Durch die Instabilität des Testchips und den Verlust seiner Konfiguration durch das Auslösen eines Reset‘s wurde auf diese Möglichkeit verzichtet. Die Ursache für das Problem konnte trotz einiger Untersuchungen nicht gefunden werden. Ein Reset wird

beispielsweise manchmal ausgelöst, wenn ein neues Datenwort zum Testchip übertragen wird. Als Fehlerquelle wurde die Implementierung des Kommunikations-Interfaces (SPI) in die RBB-Testsoftware ausgeschlossen, da sie identisch ist mit der bereitgestellten Steuerungssoftware für den TC1.5 von Infineon ist. Als potenzielle Ursache für das Fehlverhalten wird eine Schwachstelle im SPI-Modul vermutet. Daher liegt es nahe, dass auch die unbeabsichtigte Auslösung eines Resets ebenfalls auf das nicht einwandfrei implementierte SPI-Modul zurückzuführen ist. Das Problem wurde durch die Versorgung der Nwell bzw. Pwell Spannung über externe Spannungsquellen gelöst. Es bestand die Möglichkeit, die Schalter für RBB so zu verstellen, dass die Potenziale der Wannen extern definiert werden konnten. Aufgrund dieser Instabilität wurde auf eine weitere Automatisierungsstufe verzichtet. Jeder einzelne Test wurde somit von Hand gestartet. Nach Beendigung der Messung wurden die Daten manuell auf Plausibilität überprüft. Als Entscheidungskriterium für die Validität der Messdaten wurden die aufgenommenen Ströme in den Zweigen Vnwell, Vpwell und VDDC herangezogen. Diese Ströme sollten linear abfallen. War in den Daten jedoch ein rapider Abfall erkennbar, konnte von einer fehlerhaften Messung ausgegangen werden. Dieser Punkt wird in der Auswertung der Messergebnisse deutlich dargestellt.

Trotz dieser Probleme wurden die Messungen erfolgreich mit dem entwickelten Charakterisierungsmessplatz und der entwickelten Testsoftware durchgeführt und beendet. Durch eine Verbesserung der Software könnte eine Reduktion der Testzeit erzielt werden. Dafür müssten Wartezeiten reduziert und angepasst werden, die zwischen den einzelnen Kommandos der Messinstrumente eingehalten werden. Dafür müssten aber detaillierte Untersuchungen zu den einzelnen Messgeräten getätigt werden, um genaue Informationen bezüglich der Einschwingzeiten treffen zu können. Dies war in diesem Umfang im Rahmen des Projekts nicht möglich und wurde nur soweit optimiert, bis eine moderate Testzeit erreicht wurde.

(Dieser Abschnitt unterliegt der Geheimhaltung.)

Diese Masterthesis hat einen sehr guten Einblick in die Arbeit eines Testingenieurs gegeben. Es ist deutlich geworden, wie die Automatisierung eines Messplatzes umgesetzt wird, wie der allgemeine Ablauf einer Charakterisierung aussieht und wie eine Testsoftware für einen Messplatz entwickelt wird.

Abbildungsverzeichnis

TABELLENVERZEICHNIS

Quellen

[1] Microsoft, „Übersicht: Common Language Runtime (CLR)“, https://docs.microsoft.com/de-de/dotnet/standard/clr [abgerufen am 31.01.2019]

[2] Microsoft, “ECMA C# and Common Language Infrastructure Standards”, https://visualstudio.microsoft.com/license-terms/ecma-c-common-language-infrastructure-standards/ [Abgerufen am 31.01.2019]

[3] Python Foundation, “What is Python Good For?. In: General Python FAQ.“, https://www.python.org/doc/faq/general/#what-is-python-good-for, [abgerufen am 31.01.2019]

[4] Python Foundation, “History and License”, https://docs.python.org/3/license.html , [Abgerufen am 31.01.2019]

[5] Wikipedia, „Informationen zu Python“ https://de.wikipedia.org/wiki/Python (Programmiersprache), [abgerufen am 31.01.2019]

[6] Digital Pioneers, „Die beliebtesten Programmiersprachen: Python überholt C#“ https://t3n.de/news/beliebtesten-programmiersprachen-979869/

[7] Keysight, „82357B USB/GPIB Interface High-Speed USB 2.0“ https://www.keysight.com/en/pd-851808-pn-82357B/usb-gpib-interface-high-speed-usb-20?cc=DE&lc=ger, [Abgerufen am 31.01.2019]

[8] Keysight, „10833A GPIB Cable, 1 meter” https://www.keysight.com/de/pd-1000000290%3Aepsg%3Apro-pn-10833A/gpib-cable-1-meter?pm=PL&nid=-32518.536881030&cc=DE&lc=ger, [Abgerufen am 31.01.2019]

[9] Contec, „Basic Knowledge and Glossary for GPIB Communication” https://www.contec.com/support/basic-knowledge/daq-control/gpib-communication/, [Abgerufen am 31.01.2019]

[10] Larry Davis, “GPIB Bus“ http://www.interfacebus.com/Design_Connector_GPIB.html, [abgerufen am 31.01.2019]

[11] ITWissen.info, Bild: „Konzept über Talker, Controller, Listener“ https://www.itwissen.info/lex-images/daten-und-steuerleitungen-vom-gpib-bus-iec-488-und-ieee-625.png, [Abgerufen am 31.01.2019]

[12] Ivi Foundation, „SCPI Standard“, http://www.ivifoundation.org/docs/scpi-99.pdf, [abgerufen am 31.01.2019]

[13] Tektronik, „DS Keysight 2000 Multimeter“, https://de.tek.com/datasheet/broad-purpose-digital-multimeters/model-2000-6-1-2-digit-multimeter, [Abgerufen am 31.01.2019]

[14] Tektronik, „DS Keithley Sourcemeter 2450“, https://de.tek.com/keithley-source-measure-units/keithley-smu-2400-series-sourcemeter#, [Abgerufen am 31.01.2019]

[15] Tektronik, „DS Keithley Sourcemeter 2450“, http://download.tek.com/datasheet/2450-Datasheet_1KW-60904-0.pdf, [abgerufen am 31.01.2019]

[16] Rhode&Schwarz, „DS Rhode&Schwarz HMC804x Spannungsquelle“, https://www.rohde-schwarz.com/de/produkt/hmc804x-produkt-startseite_63493-61542.html, [Abgerufen am 31.01.2019]

[17] Rhode&Schwarz, „Fact-Sheet Rhode&Schwarz HMC804x Spannungsquelle“, https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/service_support_30/170728_HMC804x_Fact_Sheet.pdf, [Abgerufen am 31.01.2019]

[18] Simac, „General Information about Thermostream ATS 710M“ http://www.simac.com/en/simac-masic/distribution/semiconductor-electronics/environmental-testing-sterilization/intest-thermal-solutions, [Abgerufen am 31.01.2019]

[19] InTest, „Temperature-Forcing Thermostream ATS 710M“ https://www.intestthermal.com/pdfs/Thermal_ds/TemperatureForcing_ATS710M.pdf, [abgerufen am 31.01.2019]

[20] Microsoft, Bild: „Bedienelemente von Visual Studio“, https://docs.microsoft.com/de-de/visualstudio/ide/media/visualstudioide.png, [abgerufen am 31.01.2019]

[21] IVI Foundation, „VISA Implementation Specification for .NET”, http://www.ivifoundation.org/docs/vpp436_2016-06-07.pdf, [Abgerufen am 31.01.2019]

[22] Agilent, „IVI-COM driver and VISA-COM I/O programming examples in Microsoft Visual C#”, http://literature.cdn.keysight.com/litweb/pdf/5991-0603EN.pdf, [Abgerufen am 31.01.2019]

[23] P. Gargini, “Roadmap Past , Present and Future,” no. April, Conference, https://linxconferences.com/wp-content/uploads/2016/04/02-01-Gargini-ITRS-2.0-2.pdf, 2016. [abgerufen am 11.03.2019]

[24] Xillinx, “Xcell Journal 86.”, Xillinx, https://www.xilinx.com/publications/archives/xcell/Xcell86.pdf , 2014, [abgerufen am 11.03.2019]

[25] ISSCC2017, “SESSION 26 Processor-power management and clocking,” 2017.

[26] P. S. Ho, G. Wang, M. Ding, J.-H. Zhao, and X. Dai, “Reliability issues for flip-chip packages,” Microelectron. Reliab., vol. 44, no. 5, pp. 719–737, May 2004.

[27] R. Saleh, “Lecture 1 Design and Technology Trends.”

[28] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, “Leakage current mechanisms and

leakage reduction techniques in deep-submicrometer CMOS circuits," Proc. IEEE, vol. 91, no. 2, pp. 305–327, 2003.

[29] S. C. Y. Shin, "Session 26 Overview : Processor-Power Management DIGITAL CIRCUITS SUBCOMMITTEE," 2017.

[30] R. Hyman, N. Ranganathan, T. Bingel, and D. T. Vo, "A clock control strategy for peak power and RMS current reduction using path clustering," IEEE Trans. Very Large Scale Integr. Syst., vol. 21, no. 2, pp. 259–269, 2013.

[31] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," Low Power Electron. Des. (ISLPED), 2007 ACM/IEEE Int. Symp., pp. 38–43, 2007.

[32] S. Dighe, S. R. Vangal, P. Aseron, S. Kumar, T. Jacob, K. A. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, V. K. De, and S. Borkar, "Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor," IEEE J. Solid-State Circuits, vol. 46, no. 1, pp. 184–193, Jan. 2011.

[33] T. Baumann, "Variationen und ihre Kompensation in CMOS Digitalschaltungen Danksagung," 2010.

[34] H. Iwai, "Technology Scaling and Roadmap for 22nm CMOS logic and beyond," 2008.

[35] A. S. Sedra and K. C. Smith, "Microelectronic Circuits Revised Edition," Oxford Ser. Electr. Comput. Eng., p. 1392, 2007.

[36] Hugo , www.onmyphd.com, Web, [abgerufen am 31.01.2019]

[37] S. Borkar, "Digital Design for Low Power Systems.", Stanford Universitiy, Web, https://web.stanford.edu/class/ee311/NOTES/BORKAR.PDF [abgerufen am 11.03.2019]

[38] Ulrich Tietze, Christoph Schenk, Eberhard Gamm, "Halbleiter-Schaltungstechnik.", Springer Vieweg, ISBN: 978-3-662-48354-1, 2016

[39] S. Linz, I. Dcv, and A. T. V Mc, "AurixPlus-digital EVR and PMS", interne Infineon Dokumentation, 2014.

[40] R. Jacob Baker, „CMOS: Circuit Design, Layout, and Simulation", 2011

[41] Hugo , www.onmyphd.com, Web, [abgerufen am 31.01.2019]

[42] Theo Ungerer , „Mikrokontroller und Mikroprozessoren“, Springer-Verlag Berlin Heidelberg, 2007, ISBN 978-3-540-46819-6

[43] Markus Lippold, „Entwurf einer Delay-Locked Loop für die Nutzung als Time-toDigital Converter in einer Time-of-Flight Anwendung in 350nm CMOS Technologie“, Masterthesis – FH-Dortmund, Juni 2018, https://www.fh-dortmund.de/de/fb/3/personen/lehr/karagounis/medien/masterthesis_markus_lippold.pdf [abgerufen am 11.03.2019]

[44] Winfried Gehrke, Marco Winzker, Klaus Urbanski, Roland Woitowitz, „Digitaltechnik: Grundlagen, VHDL, FPGAs, Mikrocontroller“, Springer-Verlag, ISBN: 9783662497319, 2016