



Carl von Ossietzky Universität Oldenburg  
Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften  
Department für Informatik

# **Integration einer Zuverlässigkeitsbewertung und -optimierung in den RT- und Gate-Level Entwurfsfluss**

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften der Carl von Ossietzky Universität Oldenburg zur Erlangung des Grades und Titels eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

angenommene Dissertation von  
Dipl.-Ing. Malte Christoph Metzdorf  
geboren am 27.04.1982 in Lahnstein

Gutachter:

Prof. Dr.-Ing. Wolfgang Nebel

Prof. Dr.-Ing. Steffen Paul

Tag der Disputation: 7. Mai 2018

## **Kurzzusammenfassung**

Durch die Nutzung immer kleinerer Strukturgrößen bei der Herstellung digitaler Systeme entstehen durch Alterung und Zuverlässigkeit neue Probleme. In dieser Arbeit werden Simulationsmethoden entwickelt, die in einem Industrieflow integriert werden. Anhand des bedeutenden Alterungseffektes Bias Temperature Instability wird ein Alterungsmodell entwickelt. Dabei werden die Einflussgrößen der Alterung, wie Temperatur und Versorgungsspannung, berücksichtigt und durch eigene Modelle simuliert. Diese ermöglichen die Simulation von komplexen Nutzungsszenarien. Da Zeiträume von mehreren Jahren simuliert werden, wurde bei der Entwicklung dieser Modelle darauf geachtet, dass sie effizient parallel ausgeführt werden können, um sie auf Manycore-Systemen und Grafikkarten zu beschleunigen. Abschließend wird die vorgestellte Methodik auf ein Industriedesign angewendet: Anhand eines 32bit-Prozessors wird exemplarisch die Alterung über mehrere Jahre simuliert.

**Schlagwörter:** Alterungssimulation, Zuverlässigkeit, NBTI, Thermische Simulation

## **Abstract**

The use of smaller structure sizes in the production of digital systems raises new problems caused by aging and reliability. This thesis develops simulation methods that are integrated into an industry flow and develops an aging model on the basis of the important aging effect Bias Temperature Instability. Parameters of the aging effect such as temperature and supply voltage are considered and simulated through own models that allow the simulation of complex use cases. The simulated time frames can be as long as several years. Therefore the models can be efficiently parallelized on Manycore-Systems and graphics processing units. Finally the presented method is applied to an industrial design and allows the aging simulation of an 32bit-processor over several years.

**Keywords:** Aging simulation, Reliability, NBTI, Thermal simulation



---

## Inhaltsverzeichnis

---

<b>1</b>	<b>Einleitung</b> .....	<b>5</b>
<b>2</b>	<b>Grundlagen</b> .....	<b>11</b>
2.1	Hardware-Entwurf . . . . .	11
2.2	Transistoren . . . . .	12
2.3	Alterungseffekte . . . . .	13
2.4	Verlustleistung . . . . .	17
2.5	Versorgungsnetz und IR-Drop . . . . .	18
2.6	Elektrothermische Kopplung . . . . .	19
2.7	Delay und kritischer Pfad . . . . .	20
2.8	Reduktion der kritischen Pfade . . . . .	21
2.9	Thermische Simulation . . . . .	22
2.10	Industrieflow . . . . .	25
<b>3</b>	<b>Stand der Technik</b> .....	<b>27</b>
3.1	Alterungssimulation . . . . .	27
3.2	Thermische Simulation . . . . .	30
3.3	Versorgungsspannung und IR-Drop . . . . .	33
3.4	Industrietools . . . . .	33
3.5	Patente . . . . .	34
<b>4</b>	<b>Konzepte und Modelle</b> .....	<b>37</b>
4.1	Degradations-Modell . . . . .	38
4.2	Missions- und Stressszenario Modell . . . . .	50
4.3	Thermisches und Package-Modell . . . . .	52
4.4	Verlustleistungs-Modell . . . . .	57
4.5	IR-Drop-Modell . . . . .	58

4.6	Elektrothermische Kopplung . . . . .	62
4.7	Delay-Modell . . . . .	63
4.8	Designflow . . . . .	72
4.9	Zuverlässigkeitsbewertung . . . . .	76
4.10	Zuverlässigkeitsoptimierung . . . . .	78
4.11	Zusammenfassung . . . . .	81
<b>5</b>	<b>Implementierung . . . . .</b>	<b>83</b>
5.1	OpenCL . . . . .	83
5.2	NBTI-Simulation . . . . .	89
5.3	Missions- und Stressszenario . . . . .	91
5.4	Thermische Simulation und Package . . . . .	94
5.5	Simulation der Verlustleistung . . . . .	100
5.6	IR-Drop Simulation . . . . .	101
5.7	Elektrothermische Kopplung . . . . .	104
5.8	Delay-Berechnung . . . . .	106
5.9	Designflow . . . . .	108
5.10	Zusammenfassung . . . . .	111
<b>6</b>	<b>Evaluation . . . . .</b>	<b>113</b>
6.1	Modellbewertung . . . . .	113
6.2	Überprüfung der einzelnen Modelle . . . . .	116
6.3	Stressszenario und Optimierung . . . . .	132
6.4	Zusammenfassung . . . . .	138
<b>7</b>	<b>Fazit und Ausblick . . . . .</b>	<b>141</b>
<b>A</b>	<b>NBTI-Trap-Modell . . . . .</b>	<b>145</b>
<b>B</b>	<b>Tools und Dateiformate . . . . .</b>	<b>149</b>
<b>C</b>	<b>OpenCL . . . . .</b>	<b>161</b>
<b>D</b>	<b>BSIM Erweiterung . . . . .</b>	<b>163</b>
	<b>Auflistungsverzeichnis . . . . .</b>	<b>165</b>
	<b>Tabellenverzeichnis . . . . .</b>	<b>167</b>
	<b>Abbildungsverzeichnis . . . . .</b>	<b>169</b>
	<b>Literaturverzeichnis . . . . .</b>	<b>173</b>

# KAPITEL 1

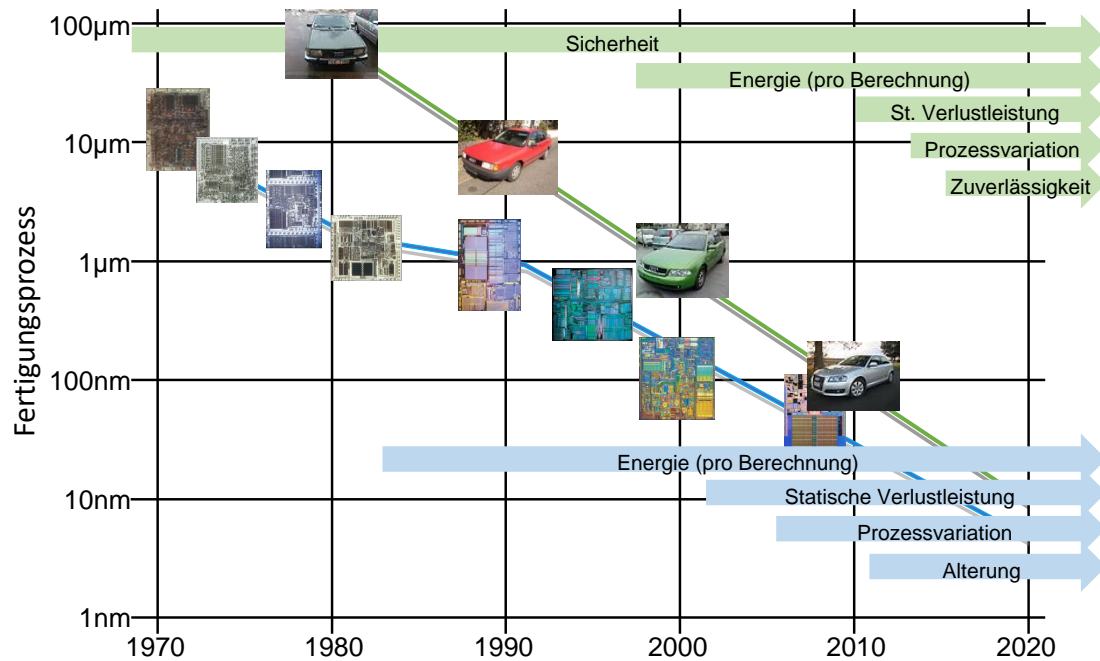
---

## Einleitung

---

### Motivation

Eine Prognose für den Umsatz des Halbleitermarktes für das Jahr 2017 zeigt, dass zum ersten Mal 400 Milliarden Dollar überschritten werden[18]. Heute befinden sich in den meisten Produkten mehrere Halbleiterbauteile, und ein Arbeiten ohne Computer ist in vielen Bereichen nicht mehr vorstellbar. Dementsprechend wird auch viel Geld in den Halbleitermarkt investiert, um neue Produkte, aber auch neue Technologien, zu entwickeln. Betrachtet man weltweit die zehn Firmen, die die höchsten Geldbeträge in Forschung und Entwicklung investieren, so kann man diese in fünf Branche unterteilen: „Automotive“, „Aerospace & Defense“, „Computer & Electronics“, „Healthcare“, „Industrials“ und „Software & Internet“. Auf einer Rangliste aus dem Jahr 2016 befinden sich auf dem 2. und 5. Platz Halbleiterhersteller[69]. Historisch betrachtet hat die Computerindustrie innerhalb des Halbleitermarktes immer eine Vorreiterrolle für die Nutzung von neuen Herstellungstechnologien inne gehabt. Dies liegt zum einen an der gewünschten Steigerung der Performance der Computer, um immer komplexere Aufgaben erfüllen zu können, zum anderen auch am Druck des Marktes, insbesondere den Gesetzen der Globalisierung, die Verringerung der Herstellungskosten pro Computer möglichst gering zu halten. Durch die neuen Herstellungstechnologien und immer komplexere Halbleiterprodukte sind die Herausforderungen in Forschung, Entwicklung und Herstellung gestiegen. Betrachtet man die Halbleiterfirmen mit den höchsten Ausgaben für Forschung und Entwicklung, so ergibt sich für die zehn Erstplatzierten Firmen eine Gesamtausgabe von 35 Milliarden Dollar[49]. Andere Produktbereiche laufen mit einem gewissen Abstand den Fertigungstechniken der Computerhalbleiter hinterher bzw. konnten zum Computerbereich in den letzten Jahren aufschließen. In Abbildung 1.1 ist der genutzte Fertigungsprozess für Computerprozessoren als blaue Linie von 1970 bis zum Jahr 2020 eingetragen.



**Abbildung 1.1: Genutzter Fertigungsprozess von der Computerindustrie zum Vergleich zu der Automobilindustrie[61].**

Im Laufe dieser Zeit entstanden einige Herausforderungen, die als blaue Pfeile in der Abbildung dargestellt sind: Ein erstes zu lösendes Problem war die steigende dynamische Verlustleistung (Energie pro Berechnung). Anfangs konnten Prozessoren ohne Kühler genutzt werden, durch den Anstieg der Taktfrequenzen mussten sie dann zuerst mit passiven Kühlkörpern und schließlich mit immer leistungsstärkeren aktiven Kühlkörpern betrieben werden. In heutiger Zeit wird ein System durch ein aktives Powermanagement bis an die thermische Leistungsgrenze betrieben, d.h. abhängig von der Temperatur werden bei Mehrkernprozessoren die Spannungen und Taktfrequenzen dynamisch angepasst. Weitere Probleme, die durch die kleineren Strukturgrößen auftreten, sind die statische Verlustleistung und die Auswirkungen der Prozessvariation. In den letzten Jahren gesellt sich zu den bereits genannten Herausforderungen eine weitere hinzu, die unter dem Oberbegriff Alterung zusammengefasst werden kann. Besonders die Alterung ist ein Problem, da die Fehler erst im laufenden Betrieb, d.h. nach einer gewissen Laufzeit auftreten können. Tests direkt nach der Herstellung können diese Fehler in einem Produkt meist nicht finden. In den letzten Jahren mussten einige Produkte aufgrund von Alterungsfehlern zurückgerufen werden und verursachten hierdurch erhebliche Kosten und Aufwand für die betroffenen Firmen von der Herstellung über den Vertrieb bis zum Endverkauf. Im Jahr 2011 gab es von Intel einen Produktionsstopp und einen Rückruf von Chipsätzen, da es zu einer erhöhten Ausfallrate kommen könne[84]. Der Fehler war ein einzelner falsch ausgelegter Transistor, der nach einer längeren Nutzungszeit ausfallen kann, wodurch die langfristige Funktion der Chipsätze von Intel nicht gewährleistet werden konnte. Der entstandene Schaden wurde von Intel mit einer Milliarde Dollar angegeben! Bei einem weiteren Fall 2017, wurden höhere Ausfallraten bei Intel



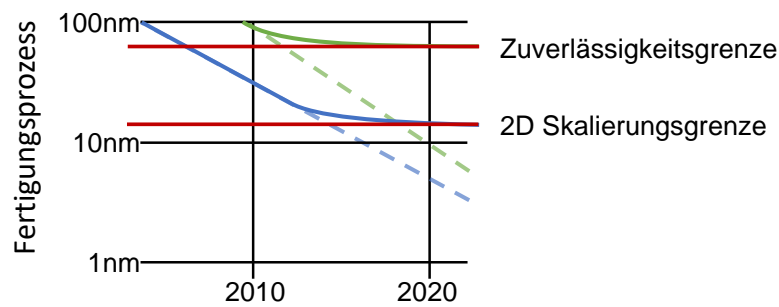
---

Atom-Prozessoren festgestellt[16]. Einige Geräte fielen bereits nach einem Jahr aus. Es wird angegeben, dass besonders die Geräte ausfallen, die häufiger stark ausgelastet sind. Die Folge ist auch hier ein Rückruf von verschiedenen Herstellern, die den Intel Prozessor benutzen.

Diese vorgestellten Probleme, die zuerst bei der Herstellung der Computerhalbleiter durch ihre fortgeschrittenen Fertigungstechniken auftreten, zeigen sich stets ab einer gewissen Technologiegeneration, je nach Anwendung zu unterschiedlichen Zeiten. Um dies zu verdeutlichen ist in der Abbildung 1.1 neben den Computerprozessoren die genutzte Halbleiter-Fertigungstechnik und die Herausforderungen in der Automobilindustrie in grüner Farbe dargestellt. Neben den bereits angeführten Herausforderungen aus der Computerindustrie, ist bei der Automobilindustrie eine wichtige Eigenschaft von besonderer Bedeutung: die Betriebssicherheit (Safety). Die genutzten Halbleiter im Automobilbereich müssen besonders robust gegenüber Ausfällen sein. Neben der Sicherheit treten die übrigen Probleme mit der Nutzung der Strukturgrößen mit dem zeitlichen Versatz dann auch bei der Automobilindustrie auf. So spielen dort dieselben Probleme wie in der Computerindustrie eine Rolle. Dieser zeitliche Versatz ist in der Abbildung durch den Abstand zwischen den blauen und grünen Pfeilen zu sehen. Die jüngste Herausforderung, die Alterung, stellt ein besonderes Problem in der Automobilindustrie dar, da sie die Zuverlässigkeit und damit auch die Sicherheit der genutzten Halbleiter beeinflusst. Ein Ausfall von Systemen während der Fahrt kann dramatische Folgen haben.

Neben den vorgestellten Schwierigkeiten kommt es bei der Entwicklung neuer und kleinerer Strukturen zu Problemen. Aufgrund von Schwierigkeiten bei der Nutzung kleinerer Strukturgrößen für Prozessoren verlangsamt sich in den letzten Jahren die Entwicklung und kommt damit an ihre physikalische Grenze. Durch zukünftige Forschungsbeiträge verschiebt sich die Skalierungsgrenze. Da es aber physikalisch eine minimale Strukturgröße gibt, weil die Strukturen nicht kleiner werden können als einzelne Atome, wird der Aufwand in der Forschung komplexer. Obwohl jede neue Generation durch das Marketing mit einer nm Zahl benannt wird, so ist die tatsächlich genutzte Strukturgröße größer als die Marketing Zahlen angeben. In einem Bericht des International Technology Roadmap for Semiconductors (ITRS) wird für die industrielle 16/14 nm Strukturgröße, die von vielen Herstellern genutzt wird, eine tatsächliche Strukturgröße der Transistoren von 24 nm angegeben[34]. Ebenso wird in diesem Bericht eine prognostizierte Grenze der effektiv genutzten Strukturgrößen bei etwa 10 nm angegeben, die als „end of 2D Domain“ bezeichnet wird. So wurde das mooresche Gesetz in den letzten Jahren mehrfach auf die neuen erkannten Probleme angepasst[82][83].

In Abbildung 1.2 ist dieser Sachverhalt, dass sowohl die Computer- als auch die Automobilindustrie an ihre Grenzen bei der Fertigungstechnik für Halbleiter kommen, durch rote Linien dargestellt, d.h. bei der Computerindustrie kommt man an die physikalischen Herstellungsgrenzen der aktuell genutzten Verfahren und Aufbaustrukturen der Transistoren. Bei der Automobilindustrie hingegen ist vor dieser physikalischen Grenze eine Nutzung der in der Computerindustrie genutzten Strukturgrößen aufgrund von Problemen mit der Zuverlässigkeit nicht mehr ohne weiteres möglich. Dies ergibt sich durch die Kombination aus Safety-Anforderungen



**Abbildung 1.2:** Ausschnitt aus Abbildung 1.1, der die Grenzen der Fertigungstechnik der Computer- (blau) und Automobilindustrie (grün) aufzeigt[61][34]. Die angegebenen Grenzen verschieben sich durch neue Forschungsergebnisse, lösen aber nicht die generellen Herausforderungen bei der Erforschung kleinerer Strukturgrößen.

(das System muss funktionieren), Alterung (das System wird irgendwann ausfallen) und Variation (der Ausfallzeitpunkt ist bei jedem System unterschiedlich). Die Zuverlässigkeitsgrenze wird durch die Alterungseffekte der Transistoren verursacht. Es bedarf einer Beurteilungsmöglichkeit der hergestellten Systeme, ob diese unter einem Szenario, bei dem das System genutzt wird, zuverlässig arbeiten können. Neben einer Beurteilung, wie stark die Alterung die Schaltung beeinflusst, wird in dieser Arbeit auch die Lücke, die es nicht ermöglicht kleinere Strukturen aufgrund von Zuverlässigkeitsproblemen zu nutzen, mit den Verfahren der vorliegenden Arbeit verringert.

### Eigener Forschungsbeitrag

Die vorliegende Arbeit soll es ermöglichen, für ein digitales System unter verschiedenen Nutzungsszenarien die Auswirkung der Alterung zu berechnen. Hierbei werden die verschiedenen Einflussgrößen berücksichtigt, die die Alterung erheblich beeinflussen. So werden unter anderem folgende, entscheidende Fragen beantwortet:

Wie entstehen die Einflussgrößen und wie hängen diese vom Szenario ab? Welche Auswirkungen können die Alterungseffekte auf unterschiedliche Technologien haben? Wie können die Alterungseffekte und die Einflussgrößen mit den bereits etablierten Industrieabläufen in Einklang gebracht werden? Wie gut lassen sich die Ansätze skalieren und können diese für größere Systeme eingesetzt werden?

Aus diesen Fragen ergeben sich folgende Ziele der Arbeit:

- Durchführung einer Alterungssimulation für digitale Systeme, die schnell und präzise unter verschiedenen Nutzungsszenarien die Alterung von Designs simuliert.
- Entwicklung von geeigneten Modellen zur Simulation der Einflussgrößen, die durch ein Nutzungsszenario beeinflusst werden.
- Integration in den bestehenden Industrieflow ohne große Anpassung.

- Skalierung der Ansätze so, dass sie für große Systeme eingesetzt werden können.
- Flexibles Gesamtkonzept, das auch für zukünftige Technologien genutzt werden kann.

Es existieren bereits einzelne Ansätze, um Alterungseffekte zu simulieren. Diese zeigen allerdings nur mit großen Einschränkungen, wie sich die Einflussgrößen über die Zeit verändern. Hierbei handelt es sich meistens um Worst-Case-Abschätzungen. Dabei hat sich bereits bei der Verlustleistung gezeigt, dass die Worst-Case-Abschätzungen nur eine geringe Aussage über realistische Szenarien geben. Beispielsweise werden deshalb bei Intel für die Auslegung der Kühlung eine „Thermal Design Power“ anstatt einer Worst-Case-Angabe genutzt[28].

Das Ziel der Arbeit ist es, eine Methodik zu entwickeln, um die Alterungssimulation von Systemen auf Gatter- und Registertransferebene für große Designs zu ermöglichen. Es gibt verschiedene Alterungseffekte. Anhand eines der zurzeit aktuellsten Alterungseffektes mit dem Namen NBTI wird das Vorgehen entwickelt. Hierbei wird nicht nur ein Alterungsmodell entwickelt, sondern auch noch Modelle, um die Einflussgrößen der Alterung zu bestimmen. Die Alterungseinflussgrößen werden zu einem Szenario zusammengefasst. Diese Alterungsgrößen entstehen durch die zeitliche Nutzung des alternden Systems und beeinflussen die Alterung erheblich. Die Ergebnisse sollen in einem frühen Entwicklungsstadium dem Designer ermöglichen, Aussagen über die Alterung des entwickelten Designs zu machen, um Optimierungen durchzuführen. Die entwickelte Methodik dieser Arbeit soll es ermöglichen, das in der Industrie verwendete Vorgehen durch die hier entstandenen Methodiken so zu erweitern, dass kein tiefer gehender Eingriff in die bereits etablierten Verfahren stattfinden muss. Durch dieses Vorgehen erlaubt die Methodik die Alterungssimulation von komplexen industriellen Designs. Am Ende soll dies evaluiert werden. Ein weiterer wichtiger Beitrag der Arbeit ist die Implementierung der Modelle und des Vorgehens. Da die zu untersuchenden Systeme aus mehreren Millionen Transistoren bestehen können und die Alterung über Zeiträume von mehreren Jahren simuliert werden soll, müssen für diese Vorhaben die Methoden besonders effizient umgesetzt werden. Die Lösung hierfür ist, die Umsetzungen hochgradig zu parallelisieren. Ohne diesen zeitintensiven Teil der Arbeit - die Implementierung - wäre es nicht möglich gewesen, die Ergebnisse von größeren Designs am Ende des Evaluationsteils durchzuführen.

### **Vorgehensweise und Aufbau der Forschungsarbeit**

Anhand des aktuell einflussreichen Alterungseffektes NBTI wird ein Modell entwickelt. Außerdem wird analysiert, welche Einflussgrößen existieren. Für die Einflussgrößen werden mehrere Modelle entwickelt. Anschließend wird der Alterungseffekt mit diesen entwickelten Modellen in den industriellen Designflow integriert. Hierbei wurde die Arbeit von Ergebnissen von Helms[24] und Eilers[14] beeinflusst und die gewonnenen Ergebnisse der vorliegenden Arbeit werden mit diesen verglichen.

Die Arbeit unterteilt sich in sechs Teile. Nach der *Einleitung*, werden in dem Kapitel *Grundlagen* die Begriffe und Verfahren erläutert, die zum Verständnis der gesamten Arbeit benötigt werden.

Anschließend werden im Kapitel *Stand der Technik* relevante Arbeiten, die sich mit dem Thema beschäftigen, analysiert und verglichen. Dabei wird eine eigenständige Problemstellung entwickelt und der eigene Ansatz herausgearbeitet. Daran schließt sich das Kapitel *Konzepte und Modelle* an. In diesem wird anhand des Alterungseffektes NBTI ein Alterungsmodell entwickelt. Ausgehend von diesem Modell werden die Modelle für die Einflussgrößen entwickelt und in den Industrieflow integriert. Im Abschluss dieses Kapitels wird diskutiert, welche Möglichkeiten es gibt, die Alterung zu verringern. Im Kapitel zur *Implementierung* werden die genutzten Modelle umgesetzt. Im Folgenden wird im Kapitel zur *Evaluation* überprüft, wie sich die entwickelten Modelle im Vergleich zu den vorhandenen Methoden verhalten. Hierbei werden zuerst die einzelnen Modelle verifiziert und anschließend das Vorgehen an industriellen Beispielen dargestellt.

### 2.1 Hardware-Entwurf

Ein typisches Vorgehen bei dem Entwurf von Hardware ist der Top-Down-Entwurf. Hierbei wird das System in einer abstrakten Darstellung zunächst hardwareunabhängig beschrieben. Anschließend wird der Entwurf in verschiedenen Schritten verfeinert, um letztendlich eine Implementierung für eine bestimmte Hardware zu erhalten. Schon 1983 haben Gajski und Kuhn das Y-Diagramm vorgestellt[17]. Dieses wurde im Laufe der Jahrzehnte von vielen Autoren genutzt und weiterentwickelt. Angelehnt an die Version von [52] und [80] wurde das Modell in Abbildung 2.1 an die Gegebenheiten der vorliegenden Arbeit angepasst und verändert.

Im Y-Diagramm sind die verschiedenen Sichtweisen und Ebenen beim Hardware-Entwurf dargestellt. Diese sind unterteilt in Verhaltens-, Struktur- und Geometriedomäne. Ausgehend von der äußeren Schicht, die die Systemebene darstellt, verfeinert jeder Schritt die Sichtweise auf das zu entwerfende Design. So beginnt die Strukturdomäne auf der höchsten Ebene mit Prozessoren und System-on-a-Chip (SoC) und endet auf der untersten Ebene bei dem System, das letztendlich aus Transistoren aufgebaut ist. Ein Wechsel der Domänen ist auf jeder Ebene möglich und oft auch notwendig.

Ein Beispiel ist durch blaue Pfeile in der Abbildung 2.1 dargestellt, diese zeigen das Vorgehen, die Ebenen und Sichtweisen, die in der vorliegenden Arbeit genutzt werden. Beginnend mit einer Designbeschreibung wird das zu entwerfende System durch eine hardwareunabhängige Synthese auf die Gatterebene gebracht und anschließend wird eine Synthese mit einer Gatterbibliothek durchgeführt (ein alternativer Weg führt über die Registertransferebene). Hiermit gelangt man von der Verhaltens- auf die Strukturdomäne des Designs. Anschließend wird auf die Transistorebene gewechselt, da für die Alterung Transistor-Informationen benötigt werden.

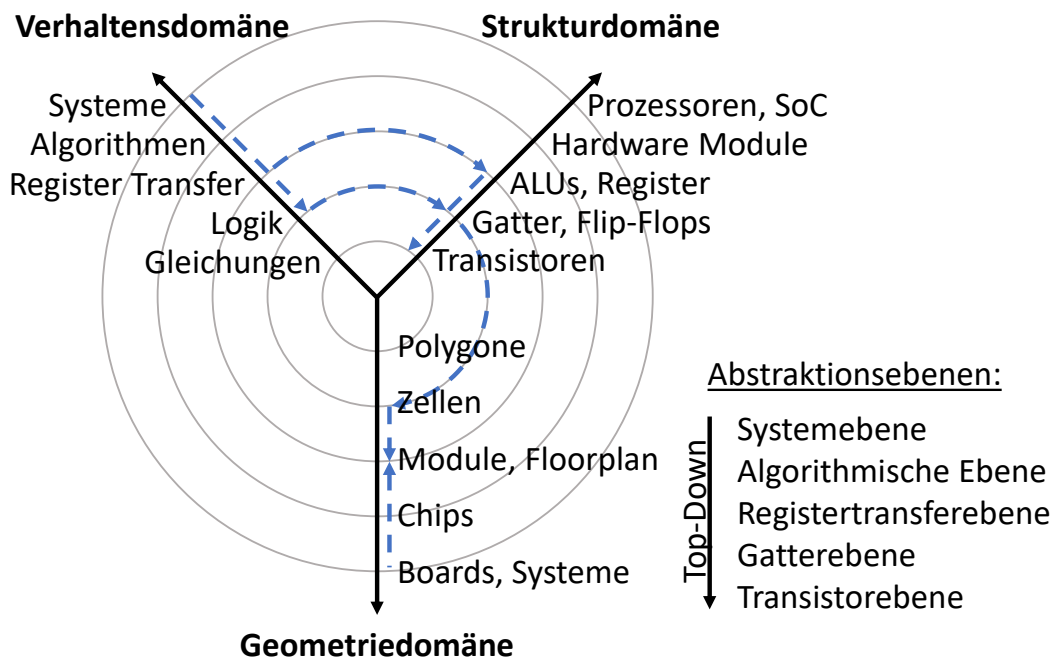
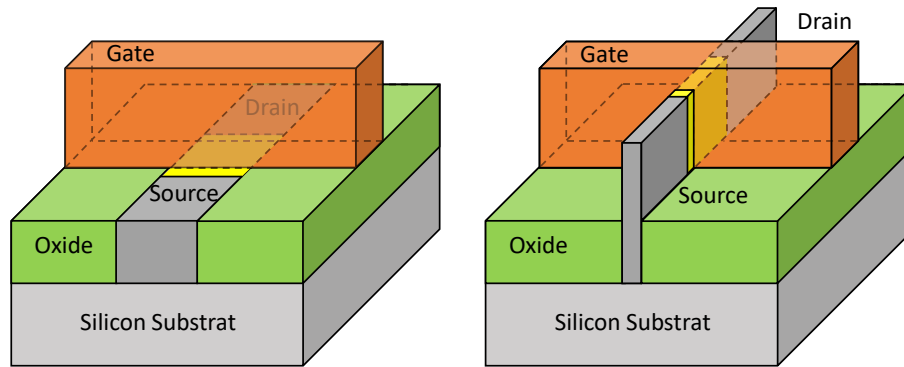


Abbildung 2.1: Y-Diagramm mit den drei Sichtweisen und den Ebenen zum Hardware-Entwurf. Die blauen Pfeile stellen den wesentlich genutzten Teil der vorliegenden Arbeit dar. Angelehnt an Quelle: [80].

Zusätzlich werden für die Alterungssimulation die Temperaturen des Designs benötigt. Hierzu wird auf die Geometriedomäne gewechselt und durch einen Floorplaner die Platzierung des Designs durchgeführt. Zusätzlich wird neben dem Floorplan auch ein Package für die thermische Simulation benötigt, d.h. dieses Package ist der physikalische Aufbau des Gehäuses. An dieser Stelle wird also eine abstrakte Sichtweise auf das System benötigt. Dies ist nur ein Beispiel, welche Rolle das Y-Diagramm im Hardware-Entwurf spielt. Weitere Vertiefungen zum Ablauf der Methodik werden in den späteren Kapiteln dieser Arbeit erläutert.

## 2.2 Transistoren

Seit der ersten Patentanmeldung eines Feldeffekttransistors des Physikers Oskar Heil im Jahr 1934[29] wurden Aufbau und Funktion des Transistors stetig weiterentwickelt. Die Strukturgrößen haben sich von  $10\ \mu\text{m}$  im Jahr 1971 bis zur heutigen Zeit (2017) auf  $14\ \text{nm}$  verkleinert. Gleichzeitig hat sich die mögliche maximale Anzahl an Transistoren, die in einem System sind, auf mittlerweile 19 Milliarden erhöht[12]. Hierbei kommt es bei der Forschung, Entwicklung und Herstellung immer kleinerer und schnellerer Transistoren auch zu größeren Herausforderungen, die bereits in der Einleitung dieser Arbeit vorgestellt wurden (siehe hierzu auch Abbildung 1.1). Eine Lösung ist es unter anderem, den über einen langen Zeitraum genutzten geometrischen Aufbau der Transistoren zu ändern. In Abbildung 2.2 sind ein Metall-Oxid-Halbleiter-Feldeffekttransistor (MOSFET) in planaren und in FinFET-Aufbau zu sehen. Bei den FinFETs wurde der leitfähige



**Abbildung 2.2:** Dreidimensionale Darstellung von MOSFETs in planarer Herstellung und als FinFETs; nicht maßstabgetreue, vereinfachte Darstellungen übernommen von Intel.

Kanal des Transistors dreidimensional und hat hierdurch einen vergrößerten Kanalbereich. Der leitfähige Kanal ist in der Abbildung gelb dargestellt. Durch den vergrößerten Kanalbereich verbessern sich einige Eigenschaften des Transistors wie z.B. kürzere Schaltzeiten. Es kommt zu grundlegenden Änderungen, wie die Temperaturabhängigkeit des Delays eines Gatters, das mit FinFETs aufgebaut ist. Gatter mit planaren Transistoren haben das längste Delay bei hohen Temperaturen, wohingegen Gatter aus FinFETs bei niedrigen Temperaturen das längste Delay haben[44].

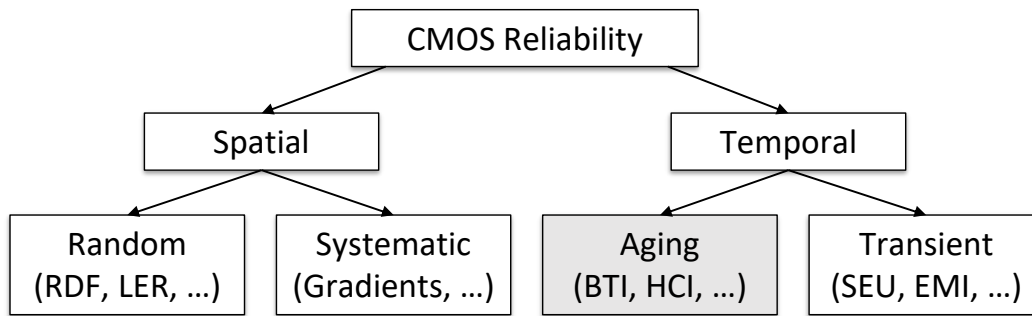
Im Kapitel zur *Evaluation* werden 45 nm planare und 16 nm FinFETs-Transistoren zum Aufbau von Logikgattern genutzt. In den immer kleineren Strukturen kommt es vermehrt zu Problemen mit den Effekten, die die Transistoren altern lassen. Diese Alterungseffekte werden im nächsten Abschnitt genauer betrachtet.

## 2.3 Alterungseffekte

Durch die immer kleineren Strukturen der Transistoren gibt es immer größere Herausforderungen bei der Herstellung der Schaltungen. Neben Problemen mit der Verlustleistung sind auch die Zuverlässigkeit der Transistoren ein Problem. In Abbildung 2.3 ist eine Übersicht von Effekten, die die Zuverlässigkeit von Transistoren beeinflussen, dargestellt.

Hierbei kann in räumliche und zeitliche Effekte unterteilt werden. Bei den räumlichen kann zusätzlich in zufällige, wie z.B. Variation der Dotierung (Random Dopant Fluctuations, RDF), und systematische unterteilt werden. Bei den zeitlichen Effekten gibt es die Unterteilung in Alterung und vorübergehende Effekte. Vorübergehende Effekte sind z.B. Single Event Upset (SEU) oder Electromagnetic Interference (EMI). Der Schwerpunkt in dieser Arbeit liegt bei den Alterungseffekten.

In Tabelle 2.1 sind bekannte Alterungseffekte und was diese Effekte beeinflusst, dargestellt. Zusätzlich ist in der Tabelle auch aufgelistet, worauf die Effekte besonders sensitiv reagieren.



**Abbildung 2.3:** Darstellung der Zuverlässigkeit von Systemen und die systematische Einordnung von Alterung (grau eingefärbt), bei der der Schwerpunkt in dieser Arbeit liegt; übernommen aus Quelle: [50].

Name	Betrifft	Einfluss*
Elektromigration (EM)	Leiterbahn	Temperatur, Material, Stromdichte Versorgungsspannung, Aktivität Feldstärke
Hot Carrier Injection (HCI)	Transistor	
Time-Dependent Dielectric Breakdown (TDDB)	Transistor	
Bias Temperature Instability (BTI)	Transistor	Temperatur, Versorgungsspannung, Aktivität

**Tabelle 2.1:** Alterungseffekte von CMOS-Schaltungen; \* Einflussgrößen sind nur exemplarisch, dabei werden einige wichtige Parameter aufgeführt.

So ist z.B. Elektromigration einer der ältesten bekannten Effekte, der materialabhängig ist, auf hohe Temperaturen reagiert und abhängig von der Stromdichte des Leiters beeinflusst wird. Hot Carrier Injection beeinflusst die Schalteigenschaften eines Transistors. Elektronen oder Löcher werden durch ein starkes elektrisches Feld beschleunigt und gelangen so in Bereiche von Transistoren, in die sie normalerweise nicht kommen sollten. Dieser Effekt kann aber auch zum Positiven genutzt werden, so z.B. bei einem nichtflüchtigen Flash-Speicher, hierbei werden die Elektronen in einem Floating Gate „gefangen“ und können so als Speicher genutzt werden. Time-Dependent Dielectric Breakdown beschreibt das Auftreten eines leitfähigen Pfades durch das Oxid des Transistorkanals. TDDB entsteht durch das Tunneln von Elektronen in das Oxid des Transistors und beschädigt dadurch den Transistor dauerhaft. Bias Temperature Instability verändert durch Vorgänge im Oxid/Kanal eines Transistors die Schalteigenschaften des Transistors. Dieser Alterungseffekt bildet den Schwerpunkt der vorliegenden Arbeit und wird daher im nächsten Unterkapitel genauer betrachtet.



### 2.3.1 Bias Temperature Instability

Durch Bias Temperature Instability (BTI) werden die Schalteigenschaften des Transistors verändert. BTI kann sowohl im PMOS- als auch im NMOS-Transistor auftreten. Hierbei wird beim PMOS-Transistor von „Negative Bias Temperature Instability“ (NBTI) und beim NMOS-Transistor von „Positive Bias Temperature Instability“ (PBTI) gesprochen. Der in letzter Zeit für aktuelle Technologien am meisten beachtete Effekt ist NBTI. NBTI ist ein Degradationseffekt, der über mehrere Jahre hinweg die Schwellspannung der PMOS-Transistoren erhöht und letztendlich zu einem Punkt führen kann, bei dem ein Transistor nicht mehr richtig durchschaltet. Dieser Schwellspannungsschaden entsteht durch Defekte im Bereich des Oxides/Kanals eines Transistors. Es gibt verschiedene Modelle in der Literatur, die die physikalischen Eigenschaften des Effektes abbilden. Diese sind aufgrund von neuen Erkenntnissen aus Messungen immer wieder überarbeitet und verbessert worden. Eines der ersten Modelle, die NBTI beschrieben, war das Reaction-Diffusion-Modell, hierbei werden die Defekte hauptsächlich durch diffundierte Wasserstoffatome aus dem Oxid eines Transistors erklärt[19]. Das Reaction-Diffusion-Modell wurde von vielen wissenschaftlichen Arbeiten auch zur Simulation der Gatteralterung genutzt. Es hat sich bei neueren Messungen herausgestellt, dass dieses Modell besonders die Erholung (Recovery) bei NBTI nicht korrekt abbilden kann[22]. Dieses hat zu einem neuen Switching-Trap-Modell geführt. Es beschreibt, wie die Fehlstellen/Defekte im Transistor als ein Trap mit zwei stabilen Zuständen (two-state-Modell) beschrieben werden kann[21], wobei jeder dieser Traps einen Beitrag zum Schwellspannungsschaden hat. Die Trap-Zustände werden als geladen (charged state) und ungeladen (uncharged state) bezeichnet. Zusätzlich hat sich bei weiteren Forschungen (Silizium Messungen) herausgestellt, dass auch dieses Modell mit nur zwei Zuständen erweitert werden muss, um die Vorgänge, die durch Messungen beobachtet wurden, durch ein Modell korrekt abbilden zu können. In Abbildung 2.4 ist ein Multi-State-Modell aus vier Zuständen dargestellt, bei dem zusätzlich zu den beiden bekannten stabilen Zuständen noch zwei meta-stabile Zustände hinzukommen, bei diesem Modell sind die Defekte auf eine Trennung von Silizium-Silizium Verbindungen zurückzuführen. Als gute Näherung kann für das komplexere Multi-State-Modell das two-state-Modell genutzt werden[19].

#### Two-state-Modell und RC-Modell

Um dieses two-state-Modell zu nutzen, kann das Modell als ein Markov-Prozess gesehen werden: Es gibt zwei feste Zustände und Übergangswahrscheinlichkeiten zwischen diesen Zuständen und die vergangenen Zustände haben keinen Einfluss auf die Folgezustände (Markov-Prozess). Die Übergangswahrscheinlichkeiten können durch Capture and Emission Time berechnet werden[20]. Diese Capture and Emission Time können auch als Lade- und Entladevorgang eines Kondensators über Widerstände aufgefasst werden. Die Gleichungen 2.1 und 2.2 beschreiben als Zeitkonstante diesen Sachverhalt. In Abbildung 2.5 ist eine Schaltung aus zwei Widerständen, zwei idealen Dioden und einem Kondensator zu sehen, der den Vorgang des Ladens und Entladens eines Defektes erklärt. Beim Ladevorgang fließt der Strom über den oberen Widerstand, beim Ent-

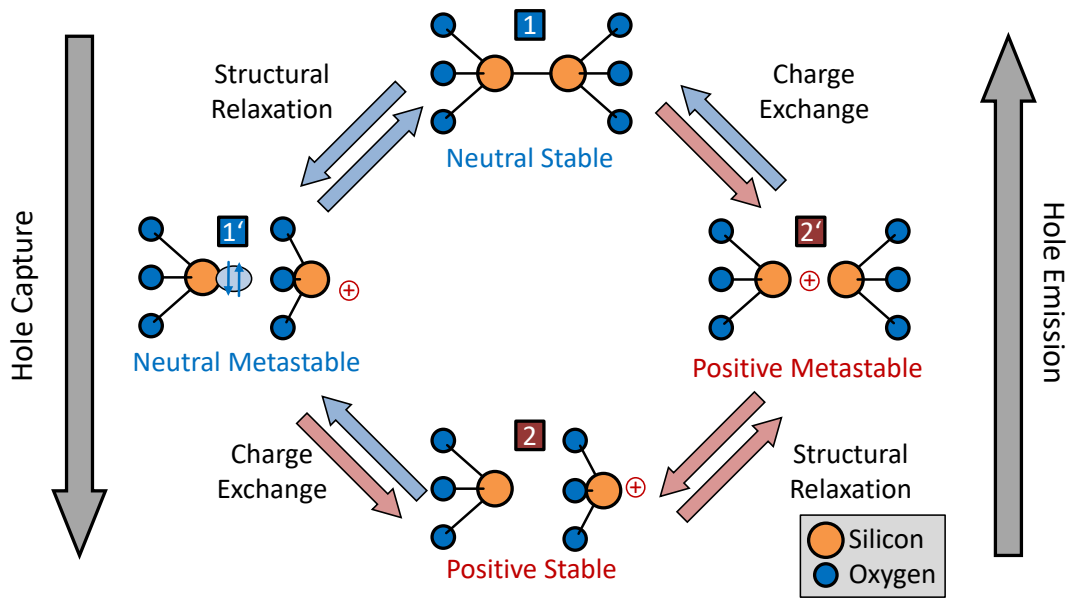


Abbildung 2.4: NBTI-Multi-State-Modell. Es zeigt das Entstehen von Defekten im Siliziumdioxid eines Transistors; übernommen aus Quelle: [20].

laden hingegen durch den unteren. Der Ladezustand des Kondensators entspricht bei dieser Veranschaulichung dem Zustand des Traps.

$$\tau_c = R_c * C_{trap} \quad (2.1)$$

$$\tau_e = R_e * C_{trap} \quad (2.2)$$

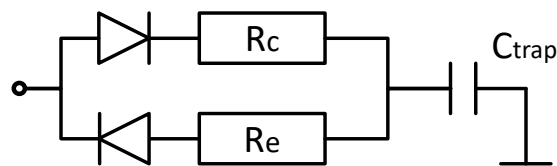


Abbildung 2.5: Dargestellt ist das RC-Trap-Modell; übernommen aus Quelle: [20].

### CET-Karten

Die Capture and Emission Times (CET) können auch zur Alterungssimulation in CET-Karten eingetragen werden, die vergleichbar mit zweidimensionalen Histogramm-Karten sind[20]. Eine CET-Karte gilt für einen festen Temperaturwert und eine feste Stressspannung, da sich die  $\tau_c$  und  $\tau_e$  Werten durch die Temperatur und die Stressspannung ändern. Auf der x-Achse der Karte sind die Werte der Capture-Zeiten und auf der y-Achse die der Emission-Zeiten aufgetragen, die durch eine Rasterung in äquidistante Klassen eingeteilt werden. Die unterschiedlichen Einflüsse der Traps auf die Schwellspannung werden dann abhängig von den  $\tau_c$  und  $\tau_e$  Werten bei den

passenden Klassen einsortiert und die Schwellspannung für diese Klasse aufsummiert. Hierbei kann die Karte mit Absolut- oder Normiert-Werten dargestellt werden. In Abbildung 2.6 ist eine solche CET-Karte dargestellt. Sowohl das RC-Modell als auch die CET-Karten können zur Alterungssimulation genutzt werden[73][14].

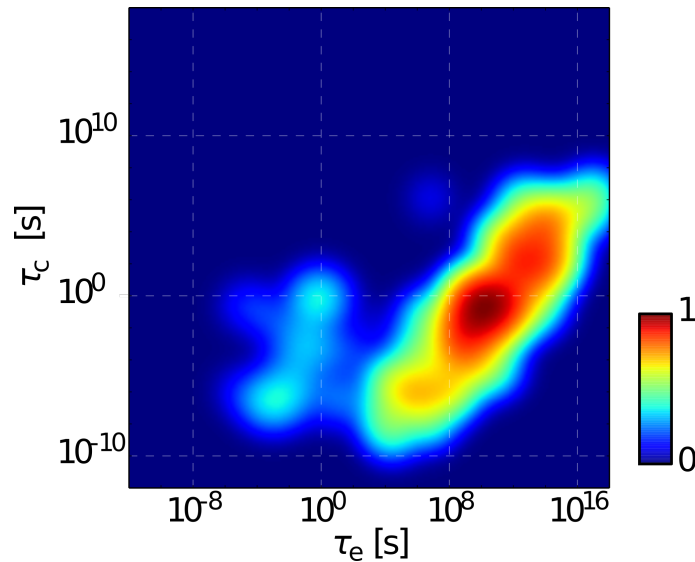


Abbildung 2.6: Abgebildet ist eine CET-Karte bei einer konstanten Temperatur und Stressspannung.

## 2.4 Verlustleistung

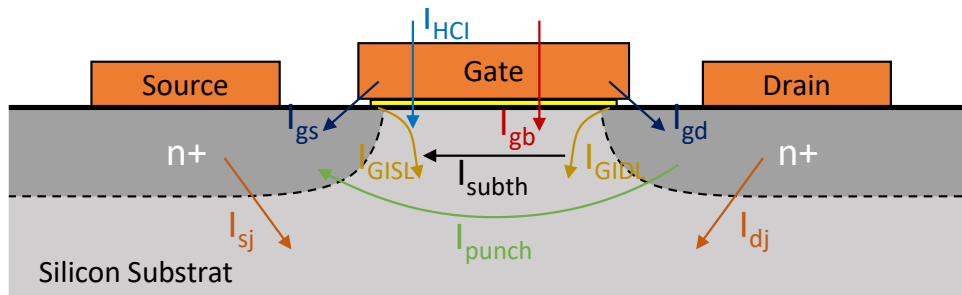
Ein entscheidender nicht funktionaler Parameter bei der Entwicklung von digitalen Systemen ist die aufgenommene Leistung des Systems, also die umgesetzte Energie bezogen auf einen Zeitraum. Besonders bei mobilen Geräten ist es ein entscheidender Faktor, wie lange ein Gerät läuft. Aber auch nichtmobile Geräte, die keine externe Einschränkung zur aufgenommenen Leistung haben, werden z.B. durch eine maximale thermische Grenze beschränkt. Die elektrische Energie in einem Halbleiter wird vollständig in Wärme umgewandelt. Letztendlich wird die elektrische Energie z.B. in Bewegung, Photonen (Licht) oder Phononen (Wärme) umgesetzt.

Die Verlustleistung kann in die statische Verlustleistung und in die dynamische unterteilt werden. Folglich ergibt sich die gesamte Verlustleistung wie in Gleichung 2.3[10].

$$P_{total} = P_{leak} + P_{dyn} \quad (2.3)$$

### 2.4.1 Statische Verlustleistung

Statische Verlustleistung (Leakage) ist die aufgenommene Leistung eines Halbleiters, die unabhängig von den Schaltvorgängen der Transistoren stattfindet, also im statischen Zustand des Transistors. Historisch betrachtet hat sich über die Zeit die Anzahl und Auswirkung der Leakage-Varianten durch die kleineren Strukturen und die genutzten Materialien erhöht. In



**Abbildung 2.7:** Abgebildet sind sieben unterschiedliche Leakage-Varianten, die in einem Querschnitt durch den Kanal aus einem planaren MOSFET aus Abbildung 2.2 eingetragen sind; übernommen aus Quelle: [24]

Abbildung 2.7 sind sieben unterschiedliche Leakage-Varianten abgebildet[24]. Diese Abbildung soll verdeutlichen, wie über die Zeit immer neue Leakage-Effekte hinzukommen und so die Verlustleistung verstärkt haben. Dies kann auch als eine Analogie zu den Alterungseffekten gesehen werden, die in ihrer Auswirkung in den letzten Jahren zugenommen haben.

Die statische Verlustleistung ist von vielen Parametern abhängig, so sind z.B. einige der Leakage-Varianten abhängig davon, ob der Transistor im sperrenden oder im leitenden Zustand ist. In Tabelle 2.2 ist die statische Verlustleistung abhängig von unterschiedlichen Eingangsbelegungen für ein NOR-Gatter bei 25 °C und 1.1 V aus der NanGate 45 nm Bibliothek[56] zu sehen.

Logischer Pegel am Eingang	Statische Verlustleistung [nW]
!A1 & !A2	20.20
!A1 & A2	16.33
A1 & !A2	18.68
A1 & A2	29.58

**Tabelle 2.2:** Leagewerte in Abhängigkeit von den Eingangswerten eines 45 nm NOR-Gatters bei 25°C und 1.1 V Versorgungsspannung

## 2.4.2 Dynamische Verlustleistung

Die dynamische Verlustleistung kann in zwei Vorgänge unterteilt werden: Der erste ist das Laden und Entladen der kapazitiven Last eines Gatters. Der andere ist der Kurzschluss, der eine kurze Zeit beim Schaltvorgang des Gatters stattfindet. Bei diesen Schaltvorgängen wird die Energie in Wärme umgesetzt. Die dynamische Verlustleistung ergibt sich aus Gleichung 2.4[33]. Sie ist abhängig von der Schaltfrequenz der  $f$ , der Aktivität  $\alpha$  und der Versorgungsspannung  $V_{DD}$ .

$$P_{dyn} = \frac{1}{2} * f * \alpha * C * V_{DD}^2 \quad (2.4)$$

## 2.5 Versorgungsnetz und IR-Drop

Abhängig vom verwendeten Material und der Geometrie haben Leitungen im Versorgungsnetz einer Schaltung unterschiedlich große Widerstände. Bei einem stromdurchflossenen Leiter kommt

es durch den Widerstand zu einem Spannungsabfall über diesem Leiter. Hierdurch wird die Spannung verringert, die letztendlich an dem zu versorgenden Bauteil anliegt. Dieser Sachverhalt, dass über die Leitung ein Spannungsabfall stattfindet, wird IR-Drop genannt. Der IR-Drop kann zeitlich gesehen sowohl statisch (also rein ohmsch) oder auch dynamisch (mit Berücksichtigung der Kapazitäten und Induktivitäten) betrachtet werden, wobei der dynamische Fall zur Unterscheidung oft als Droop bezeichnet wird[45]. In Abbildung 2.8 ist der IR-Drop exemplarisch an einer Schaltung dargestellt.

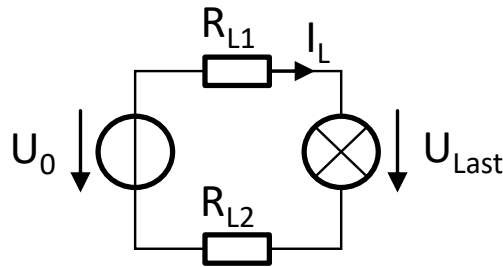


Abbildung 2.8: Schaltung zur Veranschaulichung des IR-Drops.

Die Spannungsquelle  $U_0$  soll die Last  $U_{Last}$  versorgen, dabei fließt der Strom  $I_L$  durch die Versorgungsleitungen mit den Widerständen  $R_{L1}$  und  $R_{L2}$ . Hierbei ergibt sich der wie in Gleichung 2.5 beschriebene Sachverhalt: Die Spannung an der Last wird durch die Spannungsabfälle, die an den Leitungen auftreten, reduziert.

$$U_{Last} = U_0 - R_{L1} * I_L - R_{L2} * I_L \quad (2.5)$$

Betrachtet man den Sachverhalt bei digitalen Systemen, so haben die Änderungen der Versorgungsspannung Einfluss unter anderem auf die Verlustleistung und damit auch auf die Temperatur des Systems. Dieser Sachverhalt wird im nächsten Unterkapitel elektrothermische Kopplung genauer betrachtet.

## 2.6 Elektrothermische Kopplung

Die elektrothermische Kopplung beschreibt den Sachverhalt, dass die Temperatur die Verlustleistung verändert und die Verlustleistung wiederum zu einer Veränderung der Temperatur führt[11]. In Abbildung 2.9 ist dies dargestellt.

Besonders die statische Verlustleistung ist temperaturabhängig und verändert so die gesamte Verlustleistung, die aus der dynamischen und statischen besteht. Diese neue, veränderte Verlustleistung ändert die Temperatur und dies führt dann wieder zu einer neuen veränderten Verlustleistung. In typischen Fällen führt der Effekt zu einer moderaten Veränderung der Temperatur und der Verlustleistung. Er kann aber auch in extremen Fällen, z.B. bei einer schlechten Kühlung des Systems, zu einem „thermal Runaway“<sup>1</sup> führen, so dass sich das System bis zur Zerstörung überhitzt. Neben der elektrothermischen Kopplung ist in Abbildung 2.9

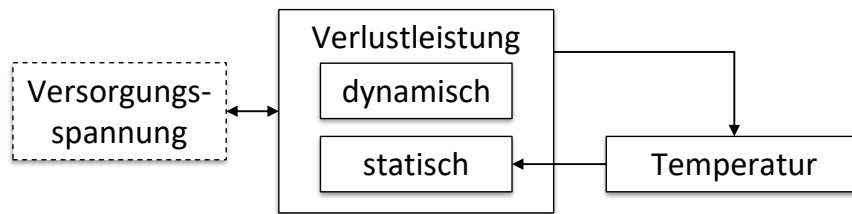


Abbildung 2.9: Abbildung der elektrothermischen Kopplung.

noch zusätzlich die Beeinflussung durch eine Veränderung der Versorgungsspannung, die z.B. durch IR-Drop hervorgerufen werden kann, dargestellt. Die Stärke des Einflusses kann durch die Gleichung 2.4 abgeschätzt werden, hier geht die Versorgungsspannung quadratisch in die Berechnung der dynamischen Verlustleistung ein.

## 2.7 Delay und kritischer Pfad

Für die Taktfrequenz eines digitalen Systems sind die Signallaufzeiten, die ein Logiksignal durch einen kombinatorischen Schaltungsblock benötigt, der entscheidende Faktor, um die maximale Taktfrequenz des Systems festzulegen. Es gibt noch weitere Faktoren, die bei der Festlegung der Taktfrequenzen eine Rolle spielen, wie z.B. die Setup- & Hold-Zeiten der Flipflops, die an dieser Stelle nicht genauer erläutert werden. Um die Ermittlung der maximalen Taktfrequenz eines Systems zu veranschaulichen, ist in Abbildung 2.10 als Beispiel der Aufbau einer synchron getakteten Schaltung dargestellt.

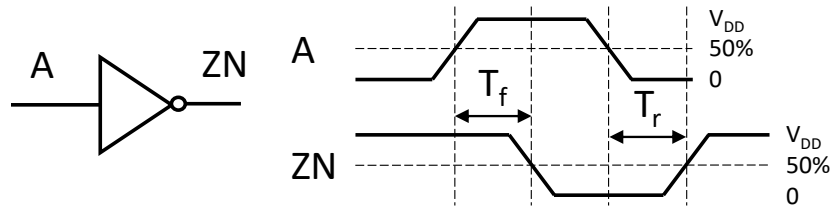


Abbildung 2.10: Synchron getaktete Digitalerschaltung.

Hierbei werden die Logiksignale in Flipflops gespeichert und bei einer steigenden Taktflanke werden neue Logiksignale gespeichert und an die Ausgänge der Flipflops weitergegeben. Für die Taktfrequenz ist also die maximale Zeit, die ein Logiksignal durch den kombinatorischen Schaltungsblock benötigt, ausschlaggebend. Hierbei ist der längste Pfad der, der sich aus den Delays der einzelnen Gatter des Pfades zusammensetzt. Dieser längste Pfad wird auch kritischer Pfad genannt. Das Delay eines Gatters ist über die Zeitspanne definiert, die ein Logiksignal von einem Eingang zum Ausgang des Gatters benötigt. In Abbildung 2.11 ist ein Eingangs- und

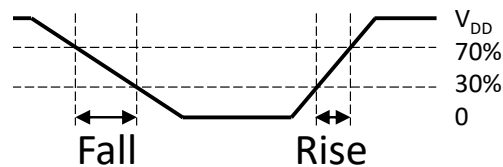
<sup>1</sup>„Thermal runaway occurs in situations where an increase in temperature changes the conditions in a way that causes a further increase in temperature, often leading to a destructive result. It is a kind of uncontrolled positive feedback.“ [[https://en.wikipedia.org/wiki/Thermal\\_runaway](https://en.wikipedia.org/wiki/Thermal_runaway); Stand 01. Juni 2017]

Ausgangssignalverlauf mit der Bestimmung des Delays für einen Inverter Gatter dargestellt.



**Abbildung 2.11: Definition des Delays eines Inverter Gatters; übernommen aus Quelle: [4].**

Das Delay wird bei 50% des Signalpegels gemessen. In der Abbildung 2.11 ist dies für ein fallendes  $T_f$  und steigendes  $T_r$  Delay dargestellt. Neben der Abhängigkeit von einer steigenden und fallenden Flanke ist das Delay außerdem abhängig von der Technologie, der Versorgungsspannung, der Temperatur und der Slew Rate (Flankensteilheit). Die Slew Rate ist hierbei die Anstiegs- oder Abfallgeschwindigkeit des Logiksignals und ist wie in Abbildung 2.12 dargestellt definiert.



**Abbildung 2.12: Definition der Slew Rate; übernommen aus Quelle: [4].**

Die Slew Rate ist die Zeitspanne, die ein Logiksignal für das Erreichen des 30% und 70% Pegels benötigt, wie dies für die steigende und fallende Slew Rate in der Abbildung dargestellt ist.

## 2.8 Reduktion der kritischen Pfade

Ein System besteht nicht nur aus einem kritischen Pfad, sondern durch Alterung kann sich der Pfad auch verändern. Dies wird in späteren Kapiteln noch aufgegriffen. Im Folgenden wird ein Verfahren vorgestellt[47], dass die Anzahl der möglichen kritischen Pfade reduzieren und so den Rechenaufwand minimieren kann.

Das Verfahren besteht aus drei unabhängigen Schritten, die nacheinander ausgeführt werden und jeweils die Gesamtzahl der möglichen Gatter stark reduziert. Hier wird nun der erste Schritt erläutert, da dieser den größten Einfluss hat. Der Ausgangspunkt ist die Annahme, dass alle Pfade kritisch sein können. Mit Hilfe einer Static Timing Analysis (STA) wird die Schaltung analysiert und es werden zum einen die Delays und Timings der Gatter für den ungealterten Fall berechnet, zum anderen die Delays und Timings für den schlimmsten Alterungsfall, d.h. der Worst Case für die Alterungsparameter. Anhand dieser beiden Fälle können die kritischen Pfade, mit den drei Verfahren, reduziert werden. Bei der „Path-based“-Reduktion wird eine Static Timing Analysis

mit Worst-Case-Delay-Werten durchgeführt, die durch eine Worst-Case-Änderung entstanden sind. Die STA erhaltenen kritischen Pfade sind nur bei der Änderung von Interesse, wenn die Pfade länger sind als der ungealterte kritischste Pfad. Dieser Sachverhalt ist in Abbildung 2.13 veranschaulicht. Hierbei sind die gealterten Delays von Pfaden einer Schaltung dargestellt, wobei zusätzlich in der Grafik das ungealterte Delay als gestrichelte Linie eingetragen ist. Es sind nur die rot abgebildeten Pfade von Interesse, da die grün dargestellten Pfade in einem beliebigen Szenario nicht stärker altern können als der Worst-Case-Fall dieses kritischen Pfades. Schon durch diesen einfachen Schritt lässt sich die Anzahl der zu untersuchenden Pfade erheblich reduzieren.

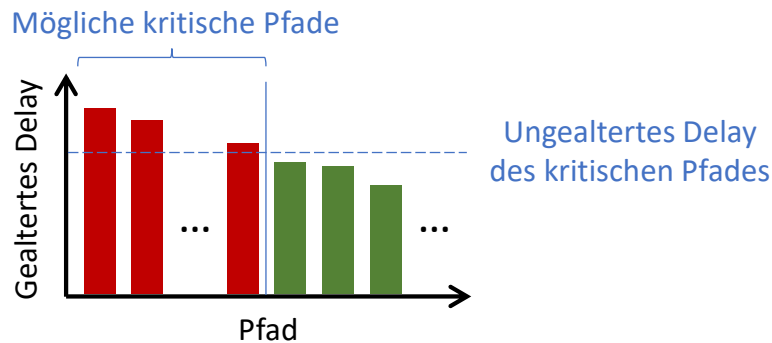


Abbildung 2.13: In der Grafik sind die Worst Case gealterten Delays von unterschiedlichen Pfaden dargestellt (sortiert beginnend mit dem längsten Pfad). Zusätzlich ist das Delay des kritischsten ungealterten Pfades als gestrichelte Linie eingezeichnet. Bei der Alterung sind nur die Pfade von Interesse, bei denen das Delay im Worst-Case-Änderungsfall über dem kritischsten ungealterten Pfad liegt. Nur diese Pfade können bei einem Szenario durch Alterung auch einen längeren Pfad als der ungealterte kritischste Pfad haben; übernommen aus Quelle: [47].

## 2.9 Thermische Simulation

Es gibt mehrere Möglichkeiten, das thermische Verhalten von Systemen zu simulieren und die Temperaturen zu berechnen. Die physikalischen Informationen, die zur Berechnung benötigt werden, sind die thermische Wärmeleitfähigkeit  $\kappa$ , die thermische Kapazität  $C_p$  und die Dichte  $\rho$  des Materials, aus dem das System aufgebaut ist. Die allgemeine Diffusionsgleichung 2.6 zeigt die zu lösende Differentialgleichung zur Berechnung der Temperaturverteilung  $T$  für nicht-konvektive thermische Systeme[43]. Neben den bereits erwähnten physikalischen Größen werden zur Berechnung noch die Leistungsverteilung  $P$  benötigt, die die Ursache der Wärmegewinnung darstellt. Die Leistungsverteilung ist bei Transistoren die Verlustleistung, die sich aus dem statischen und dynamischen Anteil zusammensetzt, wie dies bereits in dem vorherigen Kapitel dargestellt wurde.

$$\rho C_p \frac{\partial}{\partial t} T(\vec{r}, t) = \nabla(\kappa(\vec{r}, t) \nabla T(\vec{r}, t)) + P(\vec{r}, t) \quad (2.6)$$

Der Term  $P(r, t)$  der Gleichung ist der inhomogene Teil einer ansonsten mathematisch



homogenen Differenzialgleichung. Um diese Gleichung zu lösen, können zwei Fälle unterschieden werden: Die Steady State Simulation, bei der die Temperatur für nicht zeitabhängiges  $\kappa$  und  $P$  nach einer unendlich langen Zeit berechnet wird (eingeschwungener Zustand) oder die transiente Simulation, bei der die zeitliche Veränderung der Temperatur unter möglicher zeitlicher Variation, insbesondere von  $P$ , berechnet wird. Bei der Steady State Simulation vereinfacht sich die Berechnung, da der linke Teil der Gleichung 2.6 Null ist. Die Wärmekapazität und die Dichte, die zur Veränderung der Temperatur notwendige Energie (also Leistung über Zeit) bestimmen, spielen bei der Steady State Simulation keine Rolle.

Zusammengefasst werden folgende Punkte für die thermische Simulation benötigt: der physikalische Aufbau (Package), die Leistungsverteilung der aktiven Komponenten (Verlustleistung), sowie die Position der Verlustleistung (Floorplan).

### 2.9.1 Floorplan

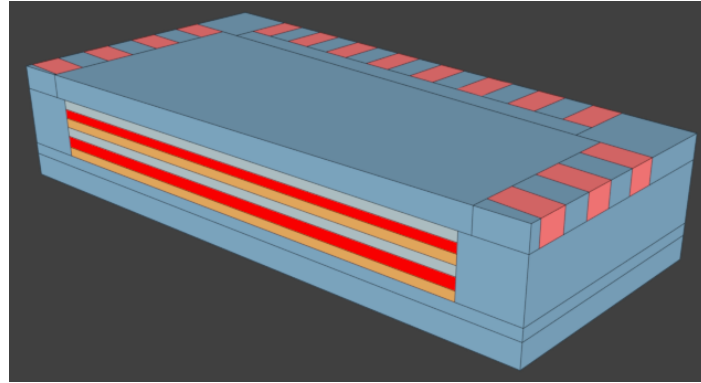
Der Floorplan beschreibt die Position der einzelnen Komponenten eines Systems. Je nach Abstraktionsebene werden unterschiedliche Daten in einem Floorplan dargestellt. So werden auf der Transistorebene die Position der Polygone, aus denen die Transistoren aufgebaut sind, dargestellt. Auf der Gatterebene werden die Zellen und auf der Registertransferebene die Module dargestellt. Hauptaufgabe des Floorplans, aus der Sicht der thermischen Simulation, ist es, die Position der Verlustleistung für das simulierte System anzugeben.

### 2.9.2 Package

Das Package beschreibt den physikalischen Aufbau des thermisch zu simulierenden Systems, d.h. aus welchen physikalischen Materialien das System aufgebaut ist. Für die thermische Simulation werden also der dreidimensionale Aufbau des Systems und die Beschreibung des Materials mit den physikalischen Materialeigenschaften Dichte, thermischer Widerstandswert und thermische Kapazität benötigt. In Abbildung 2.14 ist ein Beispiel eines typischen Packages zu sehen.

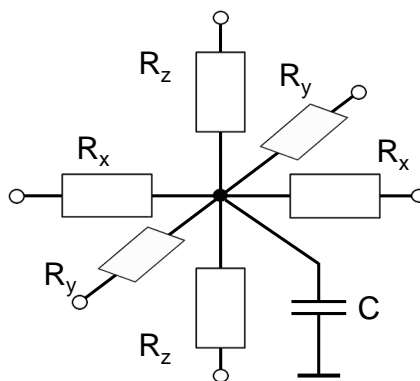
### 2.9.3 Thermische Modelle - RC-Netzwerke

Unabhängig von der zeitlichen Betrachtung gibt es verschiedene Ansätze die Temperaturen zu berechnen. Ein typisches Vorgehen zur thermischen Simulation eines Packages ist es, das Package als ein elektrisches Netzwerk, bestehend aus Widerständen und Kapazitäten (RC-Netzwerk), aufzufassen. Hierbei wird das Package in eine beliebige Anzahl an Volumina unterteilt. Je nach benötigter Auflösung führt diese Darstellung zu unterschiedlich komplexen RC-Netzwerken, wobei diese Netzwerke üblicherweise aus einzelnen Elementen, wie sie in Abbildung 2.15 dargestellt sind, bestehen. Einer der einfachsten Ansätze, eine thermische Simulation durchzuführen, ist die Finite-Differenzen-Methode (FDM). Hierbei wird das System in einzelne, gleichgroße Volumina und Zeitschritte unterteilt und der thermische Fluss zwischen diesen Volumina unter den Annahmen, dass die Temperatur in einem endlich kleinen Volumen und über eine endlich kleine



**Abbildung 2.14:** Beispiel eines QFN32-Packages, wie es zur Oberflächenmontage auf einer Platine genutzt wird; erstellt wurde das Package mit dem Package-Editor, der während dieser Arbeit entwickelt wurde (Anhang B.0.6); zu sehen ist die Unterseite eines entlang einer Achse aufgeschnittenen Packages; verschiedene Farben entsprechen unterschiedlichen Materialien; gut zu erkennen ist, dass das Package aus zwei aktiven Transistorebenen besteht.

Zeit konstant sind, iterativ berechnet. Ein in wissenschaftlichen Veröffentlichungen sehr häufig genutztes Tool, das auf diesem Modell basiert, ist HotSpot von der Universität Virginia[75]. HotSpot wird als Open Source angeboten und bietet sich daher besonders für die Forschung an. So wird im Kapitel zur *Evaluation* dieser Arbeit auch HotSpot zur Überprüfung der Ergebnisse eingesetzt.



**Abbildung 2.15:** RC-Netzwerk zur thermischen Simulation.

Eine andere Herangehensweise ist die Finite-Elemente-Methode (FEM). Hier sind die Volumina nicht mehr gleich groß, sondern können beliebig geformt sein, was zwar den Aufwand zur Überführung in ein mathematisches Problem erhöht, aber aufgrund der geeigneteren Wahl der Grenzen der Volumina deren Anzahl und damit den Aufwand der numerischen Lösung reduziert. Bekannte kommerzielle Programme sind hier ANSYS oder COMSOL Multiphysics<sup>2</sup>.

<sup>2</sup><http://www.ansys.com>, <http://www.comsol.com>

## 2.10 Industrieflow

In Abbildung 2.16 ist ein typisch industriell genutzter ASIC-Designflow zu sehen.

Beginnend mit einer Spezifikation wird das Design in einer Hardwarebeschreibungssprache erstellt. Durch eine Synthese wird das Design von der Registertransferebene auf die Gatterebene gebracht. Mit Hilfe einer Gatterbibliothek wird dann eine Netzliste des Designs erstellt. Anschließend kann mit der Netzliste eine Static Timing Analyse (STA) durchgeführt werden, um das Timing des Designs zu überprüfen. Durch das Einfügen von Teststrukturen in die Netzliste, wie Built-In Self-Test (BIST), kann die später gebaute Schaltung nach der Fertigung überprüft werden.

Der letzte Schritt ist die Analyse der Verlustleistung des Systems. Anschließend kann die Netzliste genutzt werden, um die Gatter zu platzieren und zu routen, was auch physikalische Synthese genannt wird. Durch eine formale Verifikation oder eine Simulation können die einzelnen Schritte im Designflow überprüft werden. Außerdem ist es stets notwendig zu überprüfen, ob das Design der Spezifikation entspricht, sollte z.B. bei der STA auffallen, dass das Timing der Schaltung nicht mehr eingehalten wird, muss das Design überarbeitet werden.

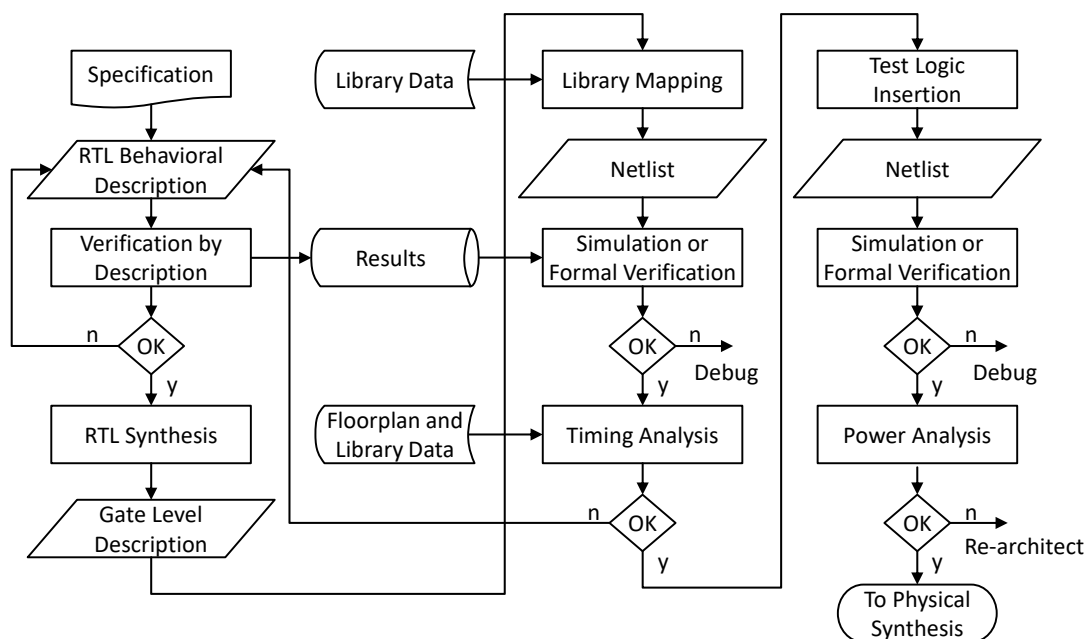


Abbildung 2.16: ASIC-Designflow, wie er in der Literatur beschrieben ist; übernommen aus Quelle: [80].



---

## Stand der Technik

---

In diesem Kapitel wird der Stand der Technik zu den relevanten Themen der vorliegenden Arbeit erläutert. Hierbei werden die Forschungslücken hervorgehoben, die im Stand der Technik gesehen werden, um darauf basierend die Ziele der Arbeit abzuleiten. Im Mittelpunkt steht das Hauptziel der Arbeit, dass es ermöglichen soll, die Alterung von digitalen Systemen auf Registertransfer- und Gatterebene, abhängig von Nutzungsszenarien und unter Berücksichtigung der Einflussfaktoren der Alterung, wie Temperatur, Versorgungsspannung und Aktivität, durchzuführen. Hierbei soll es möglich sein, auch größere Designs zu untersuchen, um die Auswirkungen von Optimierungen zu analysieren oder auch die Auswirkungen der Alterung in einer Verhaltenssimulation zu betrachten. Damit dies durchgeführt werden kann, ist neben der Wahl und der Entwicklung von geeigneten Modellen auch entscheidend auf die Performance zu achten, da Alterungen über größere Zeiträume mit mehreren tausenden Gattern durchgeführt werden sollen. Hierbei unterteilt sich der Stand der Technik auf die Bereiche der Alterungssimulation, bei der der Effekt NBTI betrachtet wird, auf den Bereich Temperatursimulation und den Bereich Versorgungsspannung. Weitere Informationen, die für die Alterungssimulation benötigt werden, werden an passenden Stellen in den späteren Kapiteln erläutert. Zusätzlich werden bereits existierende Industrietools für die Alterungssimulation und Patente in den betreffenden Bereichen betrachtet.

### 3.1 Alterungssimulation

Bei der Alterungssimulation des NBTI-Effektes lassen sich andere Arbeiten, wie in der Tabelle 3.1 dargestellt, in folgende Kategorien unterteilen und bewerten: Diese sind das genutzte NBTI-Modell und das damit unterstützte Szenario, das mit diesem Ansatz berücksichtigt werden kann. Zusätzlich wird die Performance des Ansatzes bewertet, da die Alterungssimulation in

dieser Arbeit für größere RT-Designs durchgeführt werden soll. Daher ist eine gute Performance zwingend erforderlich.

Veröffentlichung	NBTI-Modell	Szenario	Performace
Wang[77], DeBole[13], Lorenz[46] Huard[27], Mintarno[54], Cao[8]	Analytisch	Worst Case	schnell
Barke[2]	Stückweise analytisch	Variable	schnell
Kükner[39]	CET-Karten	Variable	langsam
Eilers[15]	Phasenraum	Variable	langsam
Kaczor[36], Unutulmaz[73]	Trap-Simulation	Variable	sehr langsam

**Tabelle 3.1: Vergleich von unterschiedlichen Veröffentlichungen zur Alterungssimulation von Logikgattern, sortiert nach der Performance.**

Einige der Arbeiten[77][13][46][27][54][8] nutzt zur Alterungssimulation einen analytischen Ansatz und beschreiben den Alterungsvorgang durch vereinfachte Annahmen. Hierbei wird der Alterungseffekt durch Annahmen einer konstanten Temperatur, Versorgungsspannung und Eingangsaktivität auf eine einfache Gleichung reduziert. Dies führt zu einer reinen Worst-Case-Abschätzung, die mit diesen Ansätzen möglich ist. Durch diese Ansätze wird eine unschlagbare Performance erreicht, da z.B. für jeden Transistor bei einer Alterung von zehn Jahren nur eine einfache Gleichung berechnet werden muss. In der vorliegenden Arbeit wird in dem Kapitel *Evaluation* anhand von Beispielen gezeigt, wie deutlich sich die Ergebnisse von einem Worst-Case-Szenario und einem Nutzungsszenario, bei dem die Einflussgrößen der Alterung simuliert werden, unterscheiden.

Ein Ansatz von Barke[2] zur Alterungssimulation nutzt einen abschnittswisen, zusammengesetzten, analytischen Ansatz, bei diesem wird für jeden zeitlichen Teilschritt des Nutzungsszenarios bei der Simulation die Lösung durch einen analytischen Ansatz berechnet. Durch dieses Verfahren können für die einzelnen Teilschritte unterschiedliche Temperaturen, Versorgungsspannungen und Eingangsaktivitäten gewählt und so auch komplexere Szenarien simuliert werden. Die Performance des Ansatzes ist sehr hoch, da weiterhin nur einfache Gleichungen pro Teilschritt berechnet werden müssen. Es hat sich gezeigt, dass dieser Ansatz die Vorgänge von NBTI nicht ausreichend abbilden kann und es bei der Simulation von Szenarien mit variablen Einflussgrößen zu erheblichen Abweichungen kommt[14], die ein variables Szenario nicht mehr korrekt abbilden. Ein weiterer Ansatz ist von Kükner[39], der ein Alterungsmodell nutzt, das auf CET-Karten basiert. Hierbei werden bei dem Szenario nur variable Eingangsaktivitäten der Gatter unterstützt. Die Temperatur als auch die Versorgungsspannung ist bei diesem Ansatz konstant und kann nicht über ein Nutzungsszenario simuliert werden. Der Rechenaufwand des Ansatzes ist auch deutlich aufwendiger als ein analytischer Ansatz.

Der Ansatz von Eilers[15] basiert ebenfalls auf den CET-Karten und wird Phasenraum genannt. Bei diesem Ansatz werden die Berechnungen der CET-Karten, die normalerweise für eine konstante Temperatur und Stressspannung erstellt wurden, so erweitert, dass es möglich ist, Nutzungsszenarien mit variabler Temperatur, Versorgungsspannung und Eingangsaktivität zu

simulieren. Hierbei wurde in der Arbeit von Eilers gezeigt, dass dieser Ansatz, im Vergleich zu einer Trap-Simulation, eine hohe Genauigkeit erreicht. Das Modell hat einige Nachteile, so dass es für größere Designs nicht einsetzbar ist: Die Erweiterung der CET-Karten auf eine Unterstützung von Temperatur und Stressspannung verringert die Performance so weit, dass nur Designs mit weniger als 100 Gattern in einer akzeptablen Rechenzeit durchgeführt werden können. Im Kapitel *Evaluation* folgen dazu weitere Details. Für eine neue Transistor-Technologie müssen erhebliche Charakterisierungen durchgeführt werden, die auf einem Rechnerserver zweieinhalb Wochen dauern. Dies führt zu einem Ziel der vorliegenden Arbeit: Ein flexibles Gesamtkonzept zu entwickeln, das auch für zukünftige Technologien genutzt werden kann, z.B. werden in dieser Arbeit zwei unterschiedliche Technologien zur Evaluation genutzt.

Bei Eilers werden außerdem bei dem Nutzungsszenario die Versorgungsspannung auf einzelne diskrete Werte beschränkt, da jeder weitere Wert den Speicherbedarf und Rechenaufwand des Ansatzes erheblich erhöhen würde.

Die letzten vorgestellten Ansätze kommen von Kaczer[36], bzw. Unutulmaz[73]. Sie basieren auf einer Trap-Simulation, d.h. bei der Simulation kann jeder einzelne NBTI-Trap mit einem exakten Szenario, bei dem die Temperatur, Versorgungsspannung und Aktivität variiert, simuliert werden. Diese beiden Ansätze sind von allen hier vorgestellten die langsamsten, haben aber den Vorteil, dass ein Nutzungsszenario korrekt abgebildet werden kann.

Betrachtet man die Ansätze nach der möglichen Anzahl der Gatter, die altern, so können mit den analytischen Ansätzen ganze Prozessoren untersucht werden, wie bei der Arbeit von Mintarino[54]. Bei dieser wurde ein „sub-45nm Commercial Microprocessor“ untersucht. Dies entspricht etwa einer sechs- bis siebenstelligen Anzahl an Gattern. Bei der Arbeit von Kükner[39] werden einzelne Registertransferkomponenten in der Form von Addierer und Multiplizierer betrachtet, dies entspricht einer dreistelligen Anzahl von Gattern. Bei der Arbeit von Eilers[14] wurde ein 4bit-Addierer untersucht, also eine zweistellige Anzahl an Gattern. Bei der Trap-Simulation können, wie bei der Arbeit von Unutulmaz[73], nur einzelne Transistoren und Gatter betrachtet werden.

Zusammengefasst lässt sich festhalten, dass keines dieser Modelle den Anforderungen einer Simulation eines variablen Nutzungsszenarios mit einer hohen Genauigkeit und gleichzeitig mit einer hohen Performance gerecht wird. Es zeigt sich in der Tabelle eine große Lücke zwischen den analytischen Ansätzen, die auf Worst-Case-Berechnungen zurückgreifen, und den präziseren Modellen, die eine zu niedrige Performance bieten. Daraus ist ein weiteres Ziel der vorliegenden Arbeit entstanden: Die Durchführung einer Alterungssimulation für digitale Systeme, die schnell und präzise unter verschiedenen Nutzungsszenarien die Alterung von Designs simuliert. Um die Lücke zu schließen und um eine schnelle Simulation zu erreichen, wird in dieser Arbeit die Trap-Simulation von Unutulmaz[73] aufgegriffen und das Modell so angepasst und erweitert, dass es die benötigte Performance erreicht, um größere Designs von einigen tausend Gattern zu simulieren. Hierbei ist das Ziel der Arbeit, eine Simulation eines variablen Nutzungsszenarios für eine vier- bis fünfstelligen Anzahl an Gattern zu ermöglichen. Damit wird ein Ziel der Arbeit aufgegriffen:

Die Skalierung der Ansätze so zu entwerfen, dass sie für große Systeme eingesetzt werden können. Dieses ausgewählte Modell von Unutulmaz ist zwar das rechenaufwendigste, es bietet aber durch die Berechnung jedes einzelnen Transistors auch das größte Optimierungspotential, da die Idee darauf basiert, die Traps auf einer geeigneten Hardware parallel auszuführen und gleichzeitig die Anzahl der Traps durch Optimierungen zu verringern. Neben der Entwicklung eines geeigneten Alterungsmodells, ist ein weiteres Ziel der vorliegenden Arbeit, geeignete Modelle zur Simulation der Einflussgrößen, die durch ein Nutzungsszenario beeinflusst werden, zu entwerfen. Dies ist besonders wichtig, da die Einflussgrößen bzw. das Nutzungsszenario einen erheblichen Einfluss auf die Alterungssimulation haben, d.h. zu einer Alterungssimulation gehören auch passende Modelle, die die Einflussgrößen bestimmen.

## 3.2 Thermische Simulation

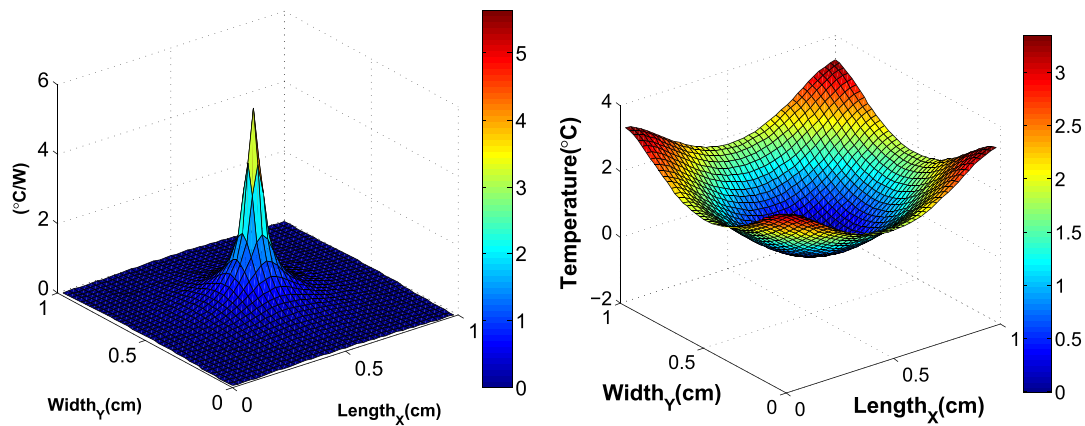
Die Anforderungen an die thermische Simulation in dieser Arbeit richten sich nach den zu simulierenden Nutzungsszenarien der Alterungssimulation. Hierbei sollen Szenarien von einigen Monaten bis zehn Jahren und mehr unterstützt werden. Eine Lösung der thermischen Gleichungen, die ein thermisches System beschreibt, kann z.B. durch das Modell eines RC-Netzwerks wie es im Kapitel zu den *Grundlagen* angesprochen wurde, durchgeführt werden. Aus Sicht dieser Arbeit ist ein großes Problem, dass viele der Lösungswege auf ein Stabilitätsproblem stoßen, das die Lösung nicht konvergieren lässt, so dass die maximale zeitliche Schrittweite bei der Simulation eingeschränkt wird[76].

Ein bekannter Vertreter des RC-Netzwerk-Modells ist das thermische Simulations Tool HotSpot[66]. Da es aber für den genutzten Ansatz in dieser Arbeit aus Performancegründen nicht ausreicht, muss ein anderer Ansatz gewählt werden. Die Performance von HotSpot wird im Kapitel zur *Evaluation* aufgezeigt.

Ein vielversprechender Ausgangspunkt für eine schnelle und gleichzeitig genaue thermische Simulation ist die Klasse von Ansätzen, die auf der Green-Funktion basieren. Im Wesentlichen wird das zu simulierende System durch eine punktförmige Wärmequelle (Impulsfunktion) angeregt, um die thermische Impulsantwort (Green-Funktion) des Systems zu erhalten. Mit dieser Impulsantwort und einer beliebigen Leistungsverteilung kann über eine Faltung die Temperatur des Systems berechnet werden[30]. In vielen der Veröffentlichungen wird auf die Probleme des Green-Ansatzes eingegangen und versucht, die Fehler des Ansatzes zu verringern, da es mehrere Eigenschaften gibt, die das System erfüllen muss, damit diese Berechnung angewendet werden kann[11]. Da dies für den weiteren Verlauf der Arbeit keine Rolle spielt, wird an dieser Stelle nicht detailliert auf die Probleme eingegangen, sondern es wird sich nur auf das Wesentliche beschränkt, um ein Verständnis für die Probleme und das Vorgehen des Ansatzes zu erhalten. Schwierigkeiten zeigen sich bei den Randwerten und den inhomogenen thermischen Strukturen innerhalb des simulierenden Systems. Hierbei ist das Hauptproblem die Annahme, dass thermische Systeme in x/y-Richtung homogen sind. Diese Annahme wird durch jedes detaillierte und



realitätsnahe Package verletzt, da es viele verschiedene Strukturen innerhalb des Chips gibt, wie z.B.: Bonddrähte, Ball Grid Arrays oder Vorzugsrichtung in der Metallisierungsebene in einem Chip. Aber auch bei 3D-Stapeln mit mehreren aktiven Ebenen müssen diese durch Leitungen (through-silicon via, TSV) untereinander verbunden werden, die sowohl aus elektrischer als auch thermischer Sicht aus leitfähigem Material bestehen, was wiederum die Homogenität verletzt. Ein anderes Problem ist auch die Annahme, dass die Wärme nur durch die z-Richtung abgeführt wird (Randwertproblem). In der Abbildung 3.1 ist auf der linken Seite eine thermische Impulsantwort (Green-Funktion), die ein System charakterisiert, dargestellt. Auf der rechten Seite ist die Temperaturabweichung zwischen einer Berechnung mit Green und mit einem kommerziellen Finite-Elemente-Simulators zu sehen. Dieses kommerzielle Programm ANSYS simuliert auch die Wärme, die an den Seiten an die Umgebung abgegeben wird. Deshalb sind in der Abbildung die Fehler an den Randwerten des simulierten Systems zu erkennen.



**Abbildung 3.1:** Auf der linken Seite ist die Package-Charakterisierung mit der Green-Funktion (Impulsantwort eines Packages) und auf der rechten Seite die Auswirkungen der Randwerte auf das Ergebnis einer thermischen Simulation mit Green-Funktion dargestellt (Differenz: Green - ANSYS); übernommen aus Quelle: [89].

Es gibt auch Ansätze, die die Green-Funktion nutzen, allerdings werden die vorgestellten Probleme nicht berücksichtigt, und so ermöglichen diese Ansätze zwar eine schnelle Simulation, haben aber auch einen deutlich größeren Fehler gegenüber RC-Netzwerk-Ansätzen[11][51]. Einen besonderen Schwerpunkt an Veröffentlichungen zu der thermischen Green-Simulation gibt es von der Quantum Electronics Group der University of California. Im Folgenden werden einige der Veröffentlichungen erläutert. Dies erfolgt in chronologischer Reihenfolge, um den Verlauf der Entwicklung des Ansatzes aufzuzeigen. Die Idee dieses Ansatzes entstand aus der Bildverarbeitung und wird als „Power Blurring“ bezeichnet, letztendlich ist es die Ausnutzung des Green-Ansatzes. Hierbei liegt der Schwerpunkt darauf, dass neben einer Beschleunigung auch eine hohe Auflösung in x/y-Richtung bei einem geringen Fehler erreicht wird. Hierzu ist das System in äquidistante Blöcke geteilt, für die die Temperaturen berechnet werden. Die erste Veröffentlichung des Green-Ansatzes für 2D-Systeme erschien 2006[38] und wurde in einigen anderen Veröffentlichungen, teilweise mit einem mathematischeren Schwerpunkt[78], oder einer

detaillierteren Analyse von Beispielen[59] noch einmal erweitert. Erste Ansätze zur Lösung der Ungenauigkeiten und Probleme von Green wurden 2007[26] veröffentlicht und basieren auf der Idee, eine für jedes zu simulierende Package eine Korrekturfunktion durch einen anderen langsameren, aber genaueren Ansatz, zu berechnen. Hierbei wurde auch eine Evaluation mit Packages gemacht, die nicht mehr homogen in x/y-Richtung aufgebaut sind. Ein Vergleich der Genauigkeit von drei verschiedenen Simulationstools (HotSpot, SescTherm und ANSYS) und der Green-Variante wurde 2011 in einem veröffentlichten Paper[88] durchgeführt. Eine Vorstellung von mehreren aktiven Schichten, also dem 3D-Einsatz, bei dem auch TSVs berücksichtigt werden können, wurde 2012 in [87] veröffentlicht. In einem Journal[89] aus dem Jahr 2014 werden die Erkenntnisse aus den bisherigen Veröffentlichungen zusammengefasst und zu einem Gesamtkonzept ausgebaut. In dem Evaluationsteil des Journals wird der Power Blurring Ansatz mit HotSpot verglichen. Hierbei benötigt eine thermische Simulation eines Prozessors, der zwei Sekunden lang läuft, 193 Minuten mit HotSpot und 67 Minuten mit Power Blurring. Zusätzlich wird eine Beschleunigung des Ansatzes durch eine Verringerung der Auflösung der Impulsantwort vorgestellt und evaluiert. Die Verringerung der Auflösung reduziert die Simulationszeit von Power Blurring auf fünf Minuten bei einem Fehler von unter 5%. Dies zeigt, dass der Ansatz eine 38-fache Beschleunigung gegenüber HotSpot bietet.

Das Ziel der vorgestellten Ansätze ist eine schnelle Berechnung einer hochauflösenden Temperaturkarte. Hierbei spielt die Genauigkeit in x/y-Richtung eine große Rolle. Aufgrund der Anforderungen der Alterungssimulation muss die Temperatursimulation für die vorliegende Arbeit Simulationen über längere Zeiträume durchführen können. Dies muss bei einer akzeptablen Geschwindigkeit möglich sein. Hierbei ist eine größere detaillierte Temperaturkarte nicht notwendig, da die Temperaturen nach einer langen Simulation auf ein hundertstel Grad genau überdimensioniert sind. So wird in dieser Arbeit eine Temperatursimulation vorgestellt und implementiert, die ebenfalls auf der Green-Funktion basiert und auf Komponentenebene die Temperaturen berechnet. Hierbei wird nicht eine hochauflösende Berechnung in x/y-Richtung durchgeführt, sondern es werden die Temperaturen der Komponenten berechnet. Dies geschieht durch eine Charakterisierung jeder einzelnen Komponente und durch den Einfluss auf andere Komponenten, d.h. es findet an dieser Stelle keine Charakterisierung des Packages statt, sondern eine Charakterisierung der Komponenten. Diese ist unterschiedlich für jedes Package. Dieses Vorgehen löst auch die Probleme, die andere Green-Ansätze mit Randwerten und inhomogenen Strukturen haben. Das Nicht-Mehr-Vorhanden-Sein der Probleme ist darauf zurückzuführen, dass jede einzelne Komponente charakterisiert wird. Weitere Details dazu folgen im Kapitel *Konzepte und Modelle*, in dem das Verfahren näher vorgestellt wird. Zusätzlich profitiert der Ansatz der vorliegenden Arbeit davon, dass er massiv parallel ausgeführt wird und so z.B. auch auf Grafikkarten, die eine hohe Performance für parallele Berechnungen haben, ausgeführt werden kann.

Ein anderer Ansatz, der auch Temperaturberechnungen auf Grafikkarten durchführt, ist von Vincenzi[74]. Hierbei wird ebenfalls eine hochauflösende Temperaturkarte berechnet. Die Imple-

mentierung basiert auf neuronalen Netzen. Der Ansatz zeigt nur eine mittelmäßige Performance gegenüber RC-Netzwerken und ist vergleichbar mit dem Vorgehen des Green-Ansatzes, bei dem eine Charakterisierung durchgeführt wird. In diesem Fall ist die Charakterisierung ein Trainieren des neuronalen Netzes.

### 3.3 Versorgungsspannung und IR-Drop

Da neben der Temperatur auch die Versorgungsspannung einen Einfluss auf den Alterungseffekt NBTI hat, sollte diese auch bei einem Nutzungsszenario berücksichtigt werden. So wird bei heutigen Systemen, z.B. durch „Dynamic Voltage and Frequency Scaling“ (DVFS), die Versorgungsspannung des Systems während des Betriebs angepasst, was eine Auswirkung auf die Alterung hat. Zusätzlich kommt es zu einer Schwankung der Versorgungsspannung durch den IR-Drop. Dieser wurde bereits im Kapitel zu den *Grundlagen* vorgestellt. Vor diesem Hintergrund wird in dieser Arbeit auch der IR-Drop bei dem Nutzungsszenario mitberücksichtigt. Hierzu wird der statische IR-Drop berechnet, der durch die Widerstände des Versorgungsnetzes entsteht. Der IR-Drop ist nur ein Randeffekt, der bei der Simulation des Nutzungsszenarios mitberücksichtigt werden kann. Auch hier ist es so, dass ein Nutzungsszenario eine Laufzeit über mehrere Jahre hat. Dabei ist die Performance für die Berechnung ein ausschlaggebender Faktor.

Folglich ist der Beitrag dieser Arbeit, ein IR-Drop-Modell zu entwickeln, das eine möglichst hohe Performance unter Ausnutzung von paralleler Berechnungen bei einer noch akzeptablen Genauigkeit bietet. Ein Flow, der die Zusammenhänge der Temperatur, der Verlustleistung, des IR-Drops und der Alterung betrachtet, wurde bereits von Rosinger[64] vorgestellt. Hierbei wird eine elektrothermische Simulation, die die Kopplung zwischen der Temperatur, der Verlustleistung und des IR-Drops durchführt, erläutert. In diesem Zusammenhang wird auch auf die Auswirkungen und das allgemeine Vorgehen zu einer Integration der simulierten Daten für eine Alterungssimulation eingegangen und der allgemeine Zusammenhang zwischen den Einflussgrößen erläutert. Für die Simulation von NBTI wird ein analytischer Ansatz vorgeschlagen. Das Vorgehen zur Berechnung der elektrothermischen Kopplung und des IR-Drops in der vorliegenden Arbeit basiert auf dieser Veröffentlichung.

Eine andere Veröffentlichung, die ähnliche Untersuchungen gemacht hat, ist von Su[70]. In dieser wird ebenfalls ein Flow für die elektrothermische Kopplung mit der Simulation des statischen IR-Drops vorgestellt. Hier wird jedoch nicht auf die Auswirkungen und die Integration einer Alterungssimulation eingegangen.

### 3.4 Industrietools

Folgende drei großen EDA-Tool-Hersteller bieten auch Tools zur Simulation von Alterungseffekten an. Diese Tools sind Mentor Graphics Reliability Simulator (Eldo), Cadence Reliability Simulator (BERT/RelXpert) und Synopsys Reliability Simulator (MOSRA). Alle drei bieten Unterstützung

für die Alterungseffekte HCI und BTI an, wobei sie auf eigene Formate für die Alterungsdaten setzen, die von Halbleiterherstellern bereitgestellt werden müssen, um diese Tools nutzen zu können. Zusätzlich bieten diese Tools an, eigene Gleichungen für die Alterungseffekte zu definieren. Die Tools nutzen die von den Herstellern angebotenen SPICE-Simulatoren. Die Alterungseffekte können hierbei während der Simulation die SPICE-Parameter der simulierten Transistoren verändern und können so auch direkte Rückwirkungen in der Schaltung aufzeigen. Die Tools nutzen bei den Modellen die analytischen Ansätze, wie sie im Unterkapitel Alterungsmodelle angesprochen wurden. Da die Simulationen mit SPICE stattfinden und diese Simulationen aus Performancegründen nicht über einen längeren Zeitraum durchgeführt werden können, können die Tools an dieser Stelle nur eine Extrapolation der Alterungseffekte durchführen. Hierbei werden nur einzelne Transistoren simuliert, um die Ergebnisse für die übrigen Transistoren in Schaltungen zu nutzen[50].

Diese Tools der Hersteller eignen sich lediglich für eine Worst-Case-Abschätzung, da sie weder eine geeignete Möglichkeit, ein Nutzungsszenario anzugeben, bieten, noch größere Schaltungen simulieren. Das Einsatzgebiet ist hauptsächlich in der Simulation von analogen Schaltungen zu sehen und spielt deshalb für diese vorliegende Arbeit keine relevante Rolle. Ein passendes Tool zur Simulation von BTI für größere digitale Schaltungen werden von diesen Herstellern zurzeit nicht angeboten. Deshalb ist es ein Ziel dieser Arbeit, eine geeignete Möglichkeit für die Integration der Alterungssimulation in den Industrieflow ohne große Anpassung zu entwickeln[53]. Hierbei werden die Tools der oben genannten Hersteller für die Entwicklung von digitalen Schaltungen genutzt.

### 3.5 Patente

Patente sind ein Ausschließlichkeitsrecht und erlauben dem Patentinhaber die ausschließliche Nutzung für einen Zeitraum von 25 Jahren des genehmigten Patent. Ein Patent wird vergeben, wenn drei Punkte erfüllt sind. Diese Punkte sind, dass die Erfindungen wirtschaftlich verwertbar und neu sind und dass eine erfinderische Höhe vorliegt[3]. Für die Recherche wurde über die Patentsuchmaschine des Europäischen Patentamtes (EPA) nach passenden Patenten gesucht[60]. Firmen beantragen Patente, um ihr Wissen, das aus Forschungstätigkeiten hervorgegangen ist, zu schützen. Seltener veröffentlichen Firmen wissenschaftliche Publikationen, da dies für sie keinen wirtschaftlichen Vorteil bringt und wissenschaftliche Ergebnisse oft Firmengeheimnisse bleiben. Deshalb beantragen Firmen Patente, da diese ihnen wie bereits beschrieben, ein Ausschließlichkeitsrecht zur Nutzung des Erforschten einräumt. Im Gegensatz zur Wissenschaft, in der meist Publikationen zum Stand der Technik entstehen, spiegeln also Patente den Stand der Technik von Firmen stärker wider. Daher werden im vorliegenden Kapitel auch Patente berücksichtigt.

Es gibt eine große Anzahl an Patenten, die den Alterungseffekt NBTI betreffen. Hierbei gibt es viele Patente, die von bekannten Halbleiterfirmen beantragt wurden, so hat z.B. IBM einige

---

Patente zur Testschaltung und Verfahren zur Erkennung von NBTI in Schaltung erhalten. So wird in dem Patent „BTI DEGRADATION TEST CIRCUIT“[35] ein Verfahren beschrieben, wie fertige Schaltungen gestresst werden und dabei NBTI gemessen werden kann. Bei einem anderen Patent von IBM[5] werden Verfahren für Alterungseffekte vorgestellt, wie Schaltungen analysiert werden sollen. Hierbei wird z.B. für den Alterungseffekt NBTI aufgezeigt, dass nur PMOS-Transistoren, die auf dem kritischen Pfad liegen, analysiert werden müssen.

Es gibt aber neben den bekannten Halbleiterfirmen auch Patente von Forschungsinstituten, wie z.B. von dem belgischen Institut IMEC, das ein Patent „Time and workload dependent circuit simulation“[9] hat. Dieses Patent ist eine Anmeldung der NBTI-Alterungssimulation, die von Kaczer[36] veröffentlicht und in Tabelle 3.1 eingetragen ist. Bei der Patentsuche wurde kein passendes Patent gefunden, das den in dieser Arbeit vorgestellten Verfahren zur Simulation von NBTI entspricht, ebenso wurde kein Patent zu der vorgestellten Integration der Alterungssimulation in die Industrietools gefunden, wie dies in dieser Arbeit vorgestellt wurde.



---

## Konzepte und Modelle

---

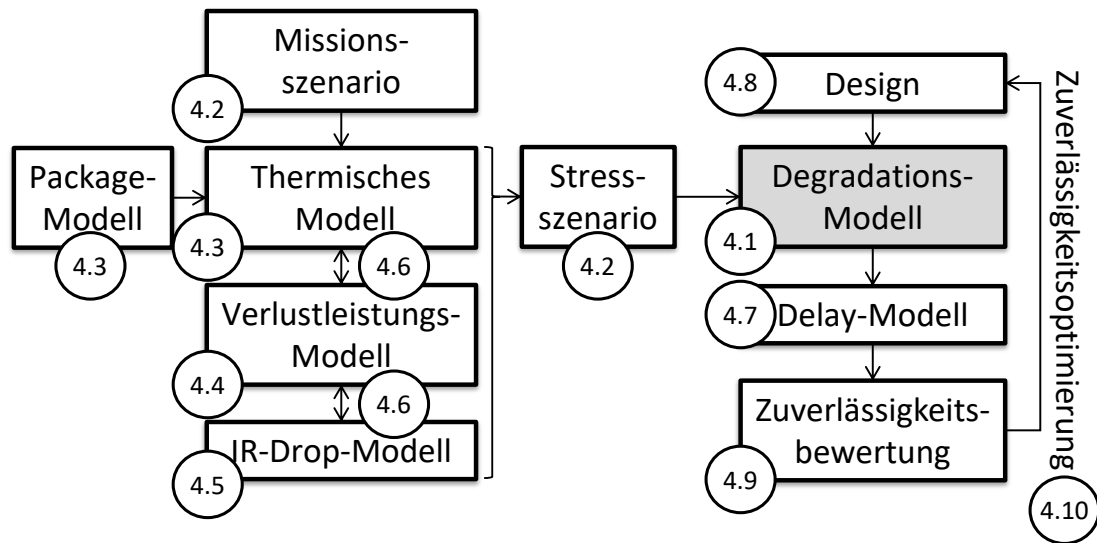
In diesem Kapitel werden die im Laufe dieser Arbeit entwickelten Modelle<sup>1</sup> vorgestellt. Diese Modelle werden genutzt, um das gesamte Konzept<sup>2</sup> der Alterungssimulation auf Gatter- und Registertransferebene durchzuführen. Das Augenmerk liegt dabei darauf, die Modelle so zu wählen, dass sie möglichst effizient<sup>3</sup> ausgeführt werden, damit auch größere Designs untersucht werden können. Die Umsetzung der Methoden erfolgt im Kapitel zur *Implementierung*. Die einzelnen Unterkapitel sind unterteilt in „Voraussetzungen“, in dem die Zusammenhänge der Modelle vorgestellt werden und in „Umsetzung“, in dem die Modelle genauer erläutert werden. In dem ersten Unterkapitel wird auf den in dieser Arbeit genutzten Alterungseffekt NBTI eingegangen und das genutzte Modell wird vorgestellt. Ausgehend von dem NBTI-Modell wird das Missions- und Stressszenario abgeleitet. Anschließend werden das thermische Modell, das Package-Modell, die Berücksichtigung der Verlustleistung mit der elektrothermischen Kopplung und die IR-Drop-Berechnung vorgestellt. Darauf folgt die Auswertung der kritischen Pfade und die Berechnung der Delays. Zum Abschluss des Kapitels wird der Zusammenhang zum Ablauf auf Gatter- und Registertransferebene aufgezeigt, dabei wird unter anderem auf die möglichen Optimierungen eingegangen. Abbildungen 4.1 stellt eine Übersicht über den Aufbau des Kapitels dar.

---

<sup>1</sup>Modell=Abstraktion von Vorgängen auf wesentliche Eigenschaften, die benötigt werden.

<sup>2</sup>Konzept=Zusammenhang und Vorgehen, wie vorgestellte Modelle genutzt werden.

<sup>3</sup>Effizient im Sinne von Optimierung auf heutige Hardware, die auf parallele Ausführung ausgerichtet ist.



**Abbildung 4.1:** Übersicht über die entwickelten und genutzten Modelle, die in diesem Kapitel erarbeitet werden. Die Kapitelnummern sind in Kreisen zu den jeweiligen Themen angegeben.

In der Arbeit werden ausgehend aus dem Degradations-Modell (NBTI-Modell) (4.1) die Anforderungen abgeleitet, die an ein Stressszenario (4.2) gestellt werden. Aus dem Missions-szenario (4.2) wird mit Hilfe des thermischen Modells (4.3), dem Package-Modell (4.3), dem Verlustleistungs-Modell (4.4), dem IR-Drop-Modell (4.5) sowie des elektrothermischen Kopplungsmodells (4.6) ein Stressszenario erzeugt. Aus dem Stressszenario und dem NBTI-Modell kann mit Hilfe des Delay-Modells (4.7) die Zuverlässigkeitbewertung (4.9) eines Systems bestimmt werden. In Kapitel 4.8 wird der Gesamtflow vorgestellt und in Kapitel 4.10 werden Optimierungsmöglichkeiten erläutert.

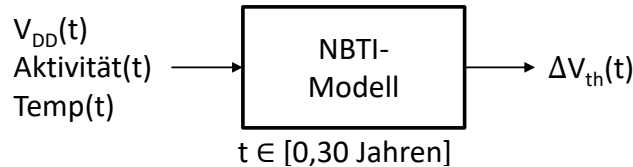
## 4.1 Degradations-Modell

### 4.1.1 Voraussetzungen

Wie bereits im Kapitel zu den *Grundlagen* bzw. im Kapitel *Stand der Technik* vorgestellt wurde, gibt es einige Modellansätze zur Simulation von NBTI, die nicht den Anforderungen gerecht werden. NBTI wird hier und in den folgenden Kapiteln als Synonym für alle Klassen von Alterungseffekten, die durch ein Trap-Modell abgebildet werden können, gesehen. Da für diese Arbeit nur technologische Informationen zu NBTI vorliegen, beschränkt sich die Entwicklung und Evaluation auf NBTI. Andere trap-basierte Modelle wie z.B. PBTI oder HCI lassen sich ebenfalls mit dem hier vorgestellten Modell simulieren. Da PBTI und HCI über die gleiche CET-Karte beschrieben werden, kann das hier entwickelte NBTI-Modell ohne große Anpassung auch für die anderen Effekte genutzt werden[32]. Vor allem die variable Temperatur- und Spannungsvariation werden in anderen Arbeiten nicht hinreichend abgebildet. Die Modelle, die es bisher gibt, können keine variable Temperatur- und Spannungsvariationen berechnen



oder haben beispielsweise wie das Trap-Modell das Problem, dass sie deutlich zu langsam sind. In Abbildung 4.2 ist die Spezifikation für das NBTI-Modell in Form des wesentlichen Ausgangspunktes und den erwarteten Ergebnissen, die in diesem Kapitel vorgestellt werden, zusammengefasst.

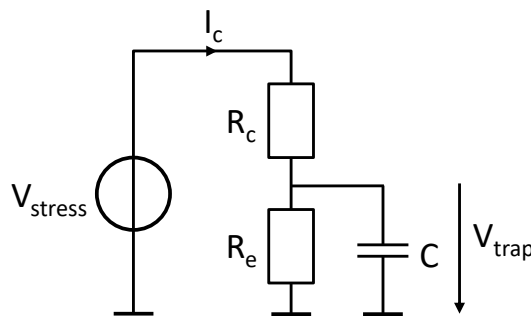


**Abbildung 4.2: Anforderungen an das NBTI-Modell:** Das Modell soll es ermöglichen, den von NBTI hervorgerufenen Schwellenspannungsschaden abhängig von der Versorgungsspannung, der Eingangsaktivität und der Temperatur zu berechnen. Alle drei genannten Größen sind dabei zeitvariabel, d.h. sie können sich beliebig über die Zeit ändern. Das Modell muss in einer dem Entwurfsprozess angemessenen Berechnungszeit Vorhersagen über lange Zeiträume (Jahre) machen.

Als Spezifikation für ein NBTI-Modell kann also zusammengefasst werden, dass ein Modell benötigt wird, welches es ermöglicht, die Auswirkungen der Temperatur, Versorgungsspannung und Eingangsaktivität über die Zeit auf die Schwellenspannung eines Transistors zu erhalten. Ausgehend von dieser Spezifikation soll im nächsten Unterkapitel ein passendes NBTI-Modell entwickelt werden.

### 4.1.2 Umsetzung

Ausgangsbasis für das in dieser Arbeit genutzte NBTI-Modell ist das NBTI-Trap-Modell. Es basiert auf der Veröffentlichung von Unutulmaz[73] und wird in diesem Abschnitt vorgestellt: Bei dem Trap-Modell wird jede Fehlstelle des Transistors in Form einer Auflade- und Entlade-Zeitkonstanten angegeben. Diese Angabe bezieht sich auf eine Ersatzschaltung, die aus einem Widerstand und einem Kondensator besteht. In dem Kapitel *Grundlagen* wird darauf näher eingegangen. Mit Hilfe dieses Ansatzes lässt sich eine Schaltung aufbauen. Diese ist in Abbildung 4.3 konkret dargestellt.



**Abbildung 4.3: Ersatzschaltung nach [73], die das Verhalten eines NBTI-Traps nachbildet.** Die Spannung  $V_{stress}$  lädt und entlädt über die Widerstände den Kondensator. Der Ladezustand des Kondensators ( $V_{trap}$ ) entspricht bei diesem Modell dem Ladezustand des Traps.

In dieser Schaltung ist  $V_{stress}$  die Spannung, mit der der Kondensator  $C$  über den Widerstand  $R_c$  auflädt, und  $R_e$  ist der Widerstand, der den Kondensator entlädt. Die Widerstände werden wie in Gleichung 4.1 aus den  $\tau_e$  und  $\tau_c$  Zeitkonstanten berechnet. Hierbei sind die  $\tau$  Werte fest durch Messungen bzw. durch TCAD-Simulation[68] vorgegeben<sup>4</sup>. Die Kapazität des Kondensators kann an dieser Stelle frei gewählt werden.

$$R_c = \frac{\tau_c}{C} \text{ und } R_e = \frac{\tau_e}{C} \quad (4.1)$$

Die Spannung des Kondensators  $V_{trap}$  beschreibt den Zustand des NBTI-Traps. Um die Spannung zu berechnen, wird durch das Aufstellen der Maschen- sowie der Knotengleichung die Differentialgleichung 4.2 gewonnen.

$$U_{stress}(t) = \frac{R_c}{R_e} U_{trap}(t) + R_c C \frac{dU_{trap}(t)}{dt} \quad (4.2)$$

Die Lösung der DGL wird im Anhang A hergeleitet, die Lösung der DGL ist in Gleichung 4.3 dargestellt. In dieser Lösungsgleichung ist  $t$  der Zeitschritt der Simulation und die Spannung  $V_{trap0}$  die Anfangsspannung des Kondensators, falls dieser bereits geladen ist.

$$V_{trap}(t) = V_{stress} \cdot \frac{\tau_e}{\tau_c + \tau_e} + (V_{trap0} - V_{stress} \cdot \frac{\tau_e}{\tau_c + \tau_e}) \cdot e^{-t \cdot (\frac{1}{\tau_e} + \frac{1}{\tau_c})} \quad (4.3)$$

Ausgehend von dieser Gleichung 4.3 kann die Gleichung 4.4 abgeleitet werden. Hierbei werden die Spannungen durch den Ladezustand (oder Wahrscheinlichkeiten) ersetzt. Dies geschieht, da die Höhe der Spannung nicht für die Lade- und Entlade-Geschwindigkeit ausschlaggebend ist. Daher kann die Berechnung unabhängig von einer Stressspannung berechnet werden.

$$p(t) = \frac{\tau_e}{\tau_c + \tau_e} + (p_0 - \frac{\tau_e}{\tau_c + \tau_e}) \cdot e^{-t \cdot (\frac{1}{\tau_e} + \frac{1}{\tau_c})} \quad (4.4)$$

Mit Gleichung 4.4 kann eine Alterungssimulation für jeden der Traps eines Transistors durchgeführt werden. Mit der Gleichung 4.5 kann im Anschluss dann der gesamte Schwellspannungsschaden, der durch die einzelnen Traps verursacht wird, ausgerechnet werden. In der Gleichung ist  $p_i(t)$  der Ladezustand jedes einzelnen Traps und  $V_{th,i}$  der Schwellspannungsschaden, den jeder Trap verursacht.

$$\Delta V_{th}(t) = \sum_i p_i(t) \cdot V_{th,i} \quad (4.5)$$

Der Beitrag eines einzelnen Traps zu dem Schwellspannungsschaden ist je nach Transistor-technologie unterschiedlich. Die Berechnung des Schwellspannungsschadens  $V_{th,i}$  ist in Gleichung 4.6 vereinfacht dargestellt. Hierbei haben neben der Breite  $w$  und Länge  $l$  des Transistors noch einige technologische Größen Einfluss auf die Spannung[20], auf die an dieser Stelle nicht

<sup>4</sup>Erläuterung zur Gewinnung der  $\tau$  Werte im Kapitel zur *Implementierung* ; TCAD = Technology Computer Aided Design

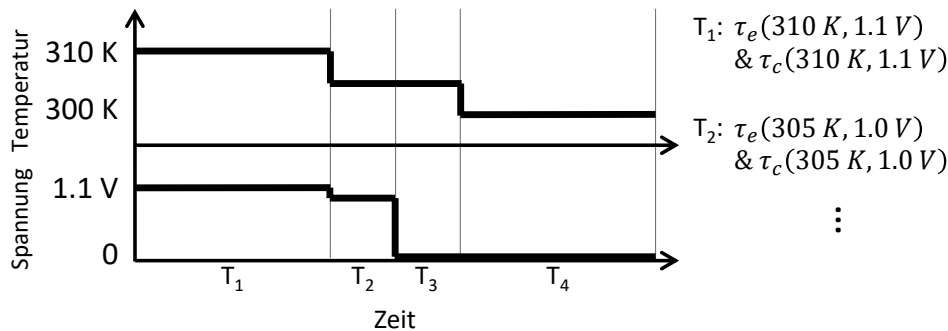
weiter eingegangen werden soll, weil es zu dem Themenschwerpunkt dieser Arbeit nicht mehr dazugehört.

$$V_{th,i} \sim \frac{1}{w \cdot l} \quad (4.6)$$

Dieses bis hier vorgestellte Trap-Modell ermöglicht es also, den Schwellspannungsschaden eines Transistors in Abhängigkeit von dessen über die Zeit ändernde Temperatur und Versorgungsspannung zu berechnen. Leider ist dieses Vorgehen zu rechenintensiv: Für jede Änderung der Temperatur und Spannung muss ein neuer Rechenschritt durchgeführt werden. Schon für einen einzelnen Transistor führt dies zu einer nicht beherrschbaren Berechnung, wenn der Transistor schnell schaltet. Da das Ziel dieser Arbeit eine schnelle und effektive Berechnung für Komponenten auf der Gatter- und Registertransferebene ist, wird in den folgenden Abschnitten das Modell erweitert, um in dem späteren Kapitel *Implementierung* eine effiziente Simulation für eine größere Anzahl an Transistoren zu ermöglichen.

### Spannungs- und Temperaturabhängigkeit

In diesem Abschnitt wird nun auf die  $\tau$  Werte eingegangen. Die Werte für  $\tau_e$  und  $\tau_c$  sind sowohl spannungs- als auch temperaturabhängig. Dies führt dazu, dass bei einer Alterungssimulation die Berechnung Abschnitt für Abschnitt mit gleichen Spannungen und Temperaturen durchgeführt werden muss. D.h. sobald sich zum Zeitpunkt des Simulationszeitschrittes die Temperatur oder die Spannung der simulierenden Traps ändert, ändern sich auch die  $\tau$  Werte bei der Gleichung 4.4. In Abbildung 4.4 ist ein Beispiel dieses Sachverhaltes dargestellt.



**Abbildung 4.4:** Beispiel zur Spannungs- und Temperaturabhängigkeit von  $\tau$ : Abgebildet sind ein Temperatur- und Versorgungsspannungsverlauf über die Zeit. Durch Änderung der Temperatur und Spannung ändern sich die  $\tau$  Werte. In diesem Beispiel entstehen vier unterschiedliche Intervalle mit verschiedenen Werten für  $\tau_e$  und  $\tau_c$ . Die Intervalle  $T_x$  werden im folgenden als Simulationsintervalle bezeichnet.

In dem Beispiel entstehen durch die unterschiedlichen Verläufe der Temperatur und Spannung vier Intervalle, bei denen die Werte von  $\tau_e$  und  $\tau_c$  unterschiedlich sind. In den folgenden Abschnitten werden diese Intervalle als Simulationsintervalle bezeichnet. Ein Simulationsintervall ist ein Zeitschritt, bei dem die Temperatur und Spannung als konstant angenommen werden.

Die Berechnung der Alterung wird also abschnittsweise für die einzelnen Simulationsintervalle durchgeführt, wie dies in der Abbildung 4.4 dargestellt ist.

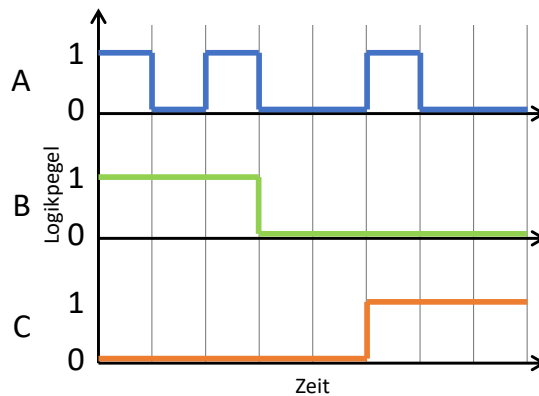
In dieser Arbeit werden rein digitale Systeme betrachtet, daher sind die Spannungspegel abhängig von den Logikpegeln 0 (Low-Pegel) oder 1 (High-Pegel) an den Eingängen der Transistoren. Es muss an dieser Stelle also eine Abbildung von den Logikpegeln auf die dazu verwendete Spannung stattfinden, um eine Alterungssimulation durchführen zu können. Durch die Betrachtung von z.B. Voltage Scaling oder Spannungsschwankungen ist der Spannungspegel zu dem Logikpegel High-Pegel nicht konstant, sondern verändert sich über die Zeit, wie dies auch im Beispiel der Fall ist. Im nächsten Abschnitt werden die digitalen Eingangssignale genauer betrachtet.

### **Eingangsaktivität**

Die Eingangsaktivitäten an den zu simulierenden Transistoren haben - wie bereits gezeigt - durch die Spannungsabhängigkeit direkte Auswirkungen auf die Alterung. PMOS-Transistoren schalten bei einem Low-Pegel an den Eingängen (aus Transistorsicht muss eine negative Spannung zwischen Source und Gate anliegen) und sperren bei einem High-Pegel. Der Signalverlauf der Logiksignale kann zeitlich beliebig verteilt sein und kann unter bestimmten Voraussetzungen für die Alterungssimulation zusammengefasst werden. Dieses Vorgehen verringert die Anzahl der Simulationsschritte und beschleunigt somit effektiv die Alterungssimulation. Um dies genauer zu analysieren, wird im folgenden Abschnitt untersucht, unter welcher Voraussetzung die Signale zur Simulation vereinfacht werden können.

### Eingangsaktivität und Signalwahrscheinlichkeit

In Abbildung 4.5 ist ein Beispiel von drei Logiksignalen dargestellt, bei denen die Ergebnisse unterschiedlicher Trap-Simulationen untersucht werden. Hierbei ist A ein Logiksignal mit beliebigen High- und Low-Pegeln. Bei den Signalen B und C wurden die High- und Low-Pegel von Signal A zusammengefasst, d.h. sie haben dieselbe Häufigkeit von High- und Low-Pegel wie Signal A mit dem Unterschied, dass bei B zuerst alle High-Pegel und bei C zeitlich alle Low-Pegel zuerst kommen. Da bei NBTI- der PMOS-Transistor bei einem Low-Pegel stärker gestresst wird als bei einem High-Pegel, bedeutet dies, dass Signal B zuerst die Traps weniger stresst und danach stärker, wohingegen Signal C zuerst stark den Transistor stresst und danach weniger.



**Abbildung 4.5:** Beispiel von Logiksignalen, die zur Untersuchung genutzt werden. Logiksignal A besteht aus einer beliebigen Anzahl verteilter High- und Low-Pegel. Die Signale B und C haben die High- und Low-Pegel zeitlich zusammengefasst, wobei bei B zuerst die High-Pegel und bei C die Low-Pegel zuerst kommen. Diese Signale werden als Stresssignale für die Trap-Simulationen genutzt und die Ergebnisse der drei unterschiedlichen Signale werden verglichen. In Abbildung 4.6 sind die Ergebnisse dargestellt.

Diese drei Signale werden als Stresssignal zur Trap-Simulation genutzt. Die Ergebnisse der Simulation werden miteinander verglichen, um zu zeigen, dass die Signale unter bestimmten Bedingungen zusammengefasst werden können. Hierzu werden Trap-Simulationen mit zufällig generierten  $\tau_e, \tau_c$  Werten, Logikpegeln und Simulationsintervallen durchgeführt. Insgesamt wurden 1 Million Simulationen eingesetzt. Die Ergebnisse von unterschiedlichen Trap-Simulationen sind in Abbildung 4.6 dargestellt. In dieser Abbildung ist auf der y-Achse der maximale relative Fehler, der zwischen dem Signal A und den Signalen B/C besteht, dargestellt. Dies ist der *maximale relative Fehler*  $= \max(|\frac{p_B}{p_A} - 1|; |\frac{p_C}{p_A} - 1|)$ , wobei  $p_x$  das Ergebnis der Trap-Simulation für ein Signal ist. Auf der x-Achse ist das Verhältnis aus dem maximalen  $\tau$  Werten und dem Simulationsintervall  $T$  aufgeführt. Dabei ist  $\tau_{max} = \max(\tau_{e,low}; \tau_{c,low}; \tau_{e,high}; \tau_{c,high})$ , wobei bei den  $\tau_{x,y}$  Werten x für die Capture/Emission-Werte und y für den Low-/High-Pegel stehen (Vergleich hierzu Abbildung 4.4).

Aus der Untersuchung kann geschlossen werden, dass bei einem großen Verhältnis von  $\tau_e, \tau_c$

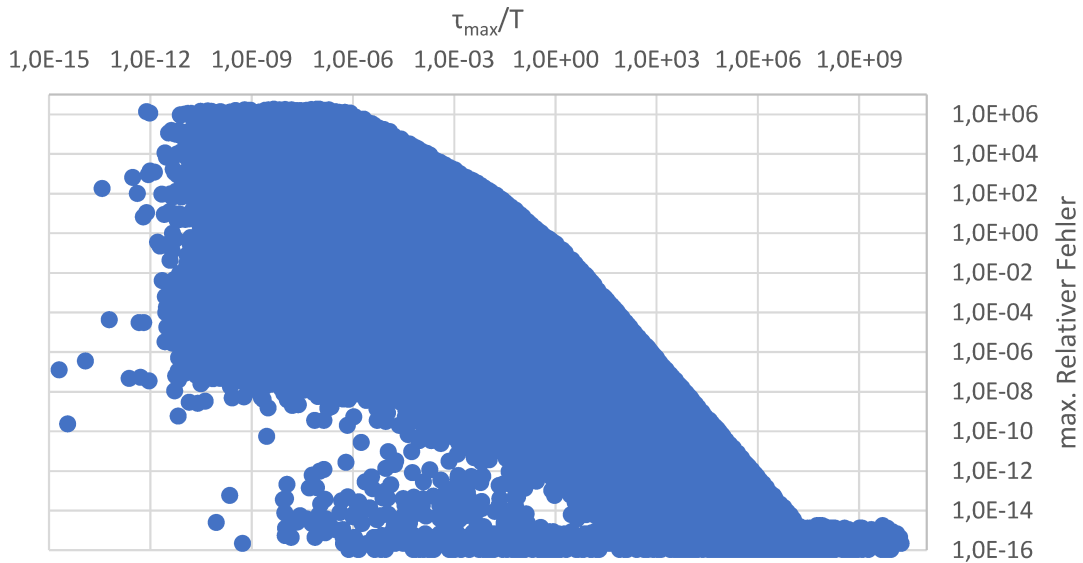


Abbildung 4.6: Ergebnisse von 1 Million unterschiedlicher Trap-Simulationen. Bei den Simulationen wurden die  $\tau_e, \tau_c$  Werte, die Logikpegel für die Stresssignale und die Simulationsintervalle zufällig generiert. In der Abbildung sind auf der x-Achse die  $\tau_{max}$  Werte im Verhältnis zum Simulationsintervall und auf der y-Achse die maximalen relativen Fehler dargestellt. Der maximale relative Fehler ist der größte relative Fehler zwischen den drei durchgeführten Trap-Simulationen mit den Signalen aus Abbildung 4.5. Bei dem Ergebnis kann erkannt werden, dass die Verteilung der maximalen relativen Fehler bei höheren  $\frac{\tau}{T}$  deutlich kleiner wird. Bei einem  $\frac{\tau}{T}$  von 10 ist der Fehler bereits unter 1%, bei  $\frac{\tau}{T} > 1000$  ist er kleiner als  $10^{-6}$  und bei  $10^7$  geht er in den numerischen Fehler der Simulation über.

zum Simulationsintervall die Logikpegel zusammengefasst werden können. Hieraus ergibt sich die Bedingung (Gleichung 4.7), dass die  $\tau_e, \tau_c$  Werte deutlich größer als das Simulationsintervall sein müssen, damit die Logikpegel für ein Intervall zusammengefasst werden können, ohne einen signifikanten Fehler bei der Alterungssimulation zu verursachen.

$$\tau_e / \tau_c \gg T \quad (4.7)$$

Diese Zusammenfassung der Logikpegel entspricht dem Berechnen der Signalwahrscheinlichkeit (Einswahrscheinlichkeit) eines Logiksignals. In Gleichung 4.8 ist die Definition der Signalwahrscheinlichkeit dargestellt.

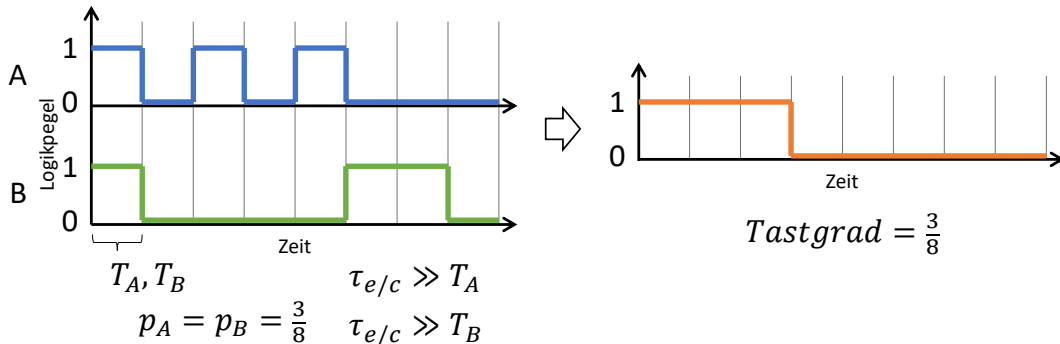
$$p = \frac{\text{Anzahl der Einsen}}{\text{Gesamtzahl der Einsen/Nullen}} \quad (4.8)$$

Es gibt eine Einswahrscheinlichkeit  $p$ , die angibt, wie wahrscheinlich es ist, dass das Signal den High-Pegel hat und es gibt die Nullwahrscheinlichkeit  $\bar{p}$ , wobei  $1 - p$  die Nullwahrscheinlichkeit ist. Die Signalwahrscheinlichkeit kann als Rechtecksignal mit einem Tastgrad<sup>5</sup> gleich der Signalwahrscheinlichkeit dargestellt werden, bzw. als Zusammenfassen der Einsen und Nullen des Signals, so dass es nur noch eine Flanke in dem Signal gibt. Diese Zusammenfassung ermög-

<sup>5</sup>Tastgrad oder auch Tastverhältnis (engl.: Duty cycle) ist definiert bei einem Rechtecksignal als das Verhältnis aus Einschaltzeit  $t$  und Periodendauer  $T$ :  $\frac{t}{T}$ ; Taschenbuch der Elektrotechnik, Kories[41], S. 493

licht es, komplexe Signale für die Alterungssimulation zu vereinfachen und die Anzahl der zu simulierenden Zeitschritte zu verringern und gleichzeitig die Schrittweite (das zu simulierende Zeitintervall) zu erhöhen.

In Abbildung 4.7 ist ein Beispiel, das das Vorgehen anhand von zwei Beispielsignalen darstellt. Beide Signale haben dieselbe Signalwahrscheinlichkeit und erfüllen die Bedingung aus Gleichung 4.7. Hierdurch können sie zu einem Rechtecksignal mit dem Tastgrad, der der Signalwahrscheinlichkeit entspricht, zusammengefasst werden.



**Abbildung 4.7:** Zwei digitale Signale A und B, die einen unterschiedlichen Verlauf haben, aber dieselbe Signalwahrscheinlichkeit. Beide Signale können für die Alterungssimulation zu einem Rechtecksignal mit einem Tastgrad von  $\frac{3}{8}$  zusammengefasst werden, da die Bedingung aus Gleichung 4.7 in diesem Beispiel erfüllt ist. Die für eine Alterungssimulation notwendigen Simulationsschritte wurden bei Signal A von 6 Schritten und bei B von 4 Schritten auf 2 Schritte reduziert (unter der Annahme, dass die Temperatur und die Spannung für diese Abschnitte konstant sind).

Ein weiteres Beispiel ist in Abbildung 4.8 dargestellt. Hier haben beide Signale A und B, wie im vorherigen Beispiel auch, dieselbe Signalwahrscheinlichkeit. Nur in diesem Fall ist bei dem Signal B die Zeitkonstante nicht deutlich größer als das Simulationsintervall. Bei der Simulation der beiden Signale unterscheiden sich die Ladeverläufe des Traps deutlich, obwohl beide Signale dieselbe Signalwahrscheinlichkeit haben. Erklären lässt sich dies, da das Signal A konstant in kleinen Zeitabschnitten den Trap lädt, wohingegen bei Signal B der Trap zuerst durch eine höhere Spannung geladen und anschließend durch eine niedrigere entladen wird.

Der Sachverhalt, dass einige Signale vereinfacht werden können und andere nicht, wird im nächsten Unterkapitel noch weiter betrachtet und analysiert.

### Short- und long-term

Die Schaltfrequenzen der digitalen Signale können bei heutigen Systemen von einigen MHz- bis in den GHz-Bereich gehen. Dieses häufige Schalten würde bei einer Alterungssimulation, bei der Traps mit niedrigen  $\tau$  Werten vorhanden sind, zu einer unbeherrschbaren Anzahl an Simulationsschritten führen (wie dies schon im letzten Abschnitt mit Abbildung 4.8 angedeutet wurde). Die Lösung für dieses Problem ist eine Unterteilung der Traps in zwei Klassen. Die erste

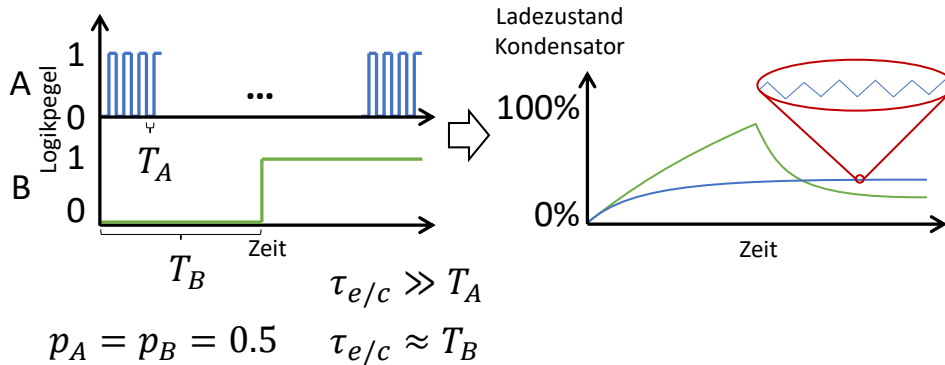


Abbildung 4.8: Gegenbeispiel zu der Abbildung 4.7: Beide Signale haben dieselbe Signalwahrscheinlichkeit. Die Simulationen führen allerdings zu unterschiedlichen Ergebnissen, da bei Signal A  $\tau$  deutlich größer als das Simulationsintervall ist, während dies bei Signal B nicht der Fall ist. Bei Signal A wechselt das Ladeverhalten in kleinen Schritten. Daher entstehen nur kleine Änderungen, die zu einer flachen Ladekurve des Traps führen. Wohingegen Signal B zuerst den Trap mit einer höheren Spannung lädt und anschließend durch eine niedrigere Spannung entlädt. Anmerkung zum Ladeverhalten von Signal B: Der PMOS wird durch einen 0 Pegel am Gate eingeschaltet und altert dadurch, wohingegen er bei einem 1 Pegel stärker heilen kann.

Klasse schaltet sehr schnell und hat kleine  $\tau$  Werte. Es ist das kurzzeitige Altern (short-term), d.h. die Traps sind schnell aufgeladen und entladen sich auch wieder schnell. Die andere Klasse sind Traps, die einen deutlich längeren Lade- und Entladevorgang haben. Es handelt sich hierbei um die Klasse der langfristigen Alterung (long-term). Diese Traps haben somit auch deutlich längere Zeitkonstanten. In Abbildung 4.9 ist dieses Vorgehen veranschaulicht, die Trapliste wird in die zwei Klassen unterteilt und anschließend wird für jede der beiden Klassen eine separate Alterungssimulation durchgeführt.

Die long-term Traps werden mit Hilfe der Signalwahrscheinlichkeiten simuliert, wie es im vorherigen Kapitel bereits erläutert wurde. Die Simulation kann durch diesen Schritt erheblich beschleunigt werden. Zusätzlich zu den Intervallen mit unterschiedlichen Signalwahrscheinlichkeiten, müssen die Intervalle noch in Teilintervalle eingeteilt werden, bei denen sich die Temperatur oder die Spannung verändert hat. Diese Temperatur- und Spannungsberücksichtigung wurde mit Abbildung 4.4 erläutert. Die Klasse der short-term Traps wird separat behandelt. Da diese Traps schnell reagieren, macht es keinen Sinn, diese Traps über einen langen Zeitraum zu simulieren. An dieser Stelle bieten sich zwei Möglichkeiten zur Behandlung der Traps an: Die erste Möglichkeit ist, die Traps erst zum Ende der Laufzeit zu simulieren, d.h. soll z.B. eine Alterung eines Systems über einen Zeitraum von zehn Jahren simuliert werden, so werden die long-term Traps für zehn Jahre simuliert und erst zum Schluss werden die short-term Traps für z.B. eine Alterungssimulation von einem Tag mit simuliert. Eine Erweiterung hierzu ist, die short-term Traps weiter zu unterteilen: Dabei kann jeder dieser Traps genau zu den passenden, relevanten Zeitpunkten bei den long-term Traps mitsimuliert werden. Nach Abschluss der Simulation werden die Auswirkungen der einzelnen Traps wie in der Gleichung 4.5 berechnet.



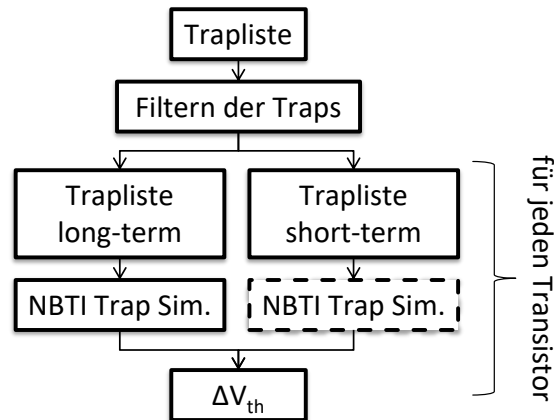


Abbildung 4.9: Aufteilung der Traps mit hoher und niedriger Zeitkonstanten. Die Traps werden anschließend separat simuliert. Die long-term Traps mit hohen Zeitkonstanten können über die Signalwahrscheinlichkeit und das bereits vorgestellte Vorgehen simuliert werden. Die short-term Traps mit einer niedrigen Zeitkonstanten können entweder nur über einen kurzen Zeitraum simuliert werden oder der Worst-Case-Fall wird angenommen, d.h., dass alle short-term Traps aktiviert werden.

Die zweite Möglichkeit, die short-term Traps zu simulieren, ist die Annahme des Worst-Case-Falles: D.h. die long-term Traps haben z.B. über einen sehr langen Zeitraum von zehn Jahren mit unterschiedlichen Temperaturen und Spannungen unterschiedliche Ladezustände erreicht. Weitere Änderungen an den Ladezuständen der long-term Traps werden einige Zeit dauern, wohingegen die short-term Traps sich teilweise bei jeder Signalveränderung laden bzw. entladen. Von daher ist die Annahme des Worst-Case-Falles, dass alle short-term Traps vollständig geladen sind, ohne sie zu simulieren, ein sinnvolles Vorgehen. Es ist schließlich die Zuverlässigkeit relevant, wie sich ein System z.B. nach etwa zehn Jahren verhält und dann genau nach einem extremen Kurzzeit-Stress (Worst Case).

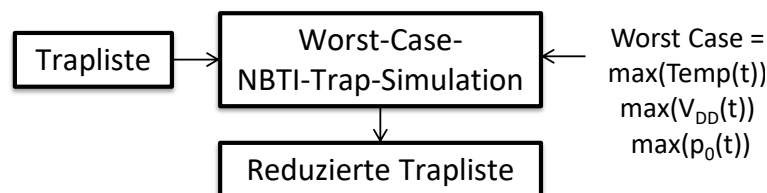
Die Möglichkeit des Auslassens der Simulation der short-term Traps durch die Signalwahrscheinlichkeit und das Laden bzw. Entladen eines Traps ist in Abbildung 4.9 durch den gestrichelten Kasten angedeutet. Es wird an dieser Stelle der Worst-Case-Fall angenommen. In Ausnahmefällen kann zur genaueren Untersuchung aber auch eine „echte“ Simulation durchgeführt werden.

### Reduktion der Traps

Je nach Anzahl der Traps pro Transistor - diese können je nach Technologie bei tausenden von Traps liegen und der Anzahl der einzelnen Temperaturen/short-term Zustände - kann die Simulation sehr lange dauern, auch mit den bereits vorgestellten Schritten. Von daher ist es sinnvoll, die Anzahl der zu simulierenden Traps weiter zu verringern und die Simulation somit weiter zu beschleunigen.

Zu einer weiteren Reduktion der Traps, die schon durch die Unterteilung in short-term und

long-term Traps geschehen ist, müssen die einzelnen Traps genauer betrachtet werden. Jeder einzelne Trap hat eine unterschiedliche Zeitkonstante (Capture and Emission Time), die von der Spannung und der Temperatur abhängig sind. Es gibt also einige Traps, die schon bei niedrigen Spannungen und Temperaturen eine niedrige Zeitkonstante (hohe Besetzungswahrscheinlichkeit) haben, wohingegen andere erst bei viel höheren Spannungen und Temperaturen aktiviert werden. Aus dieser Tatsache ergibt sich, dass es sinnvoll ist, vor der eigentlichen Simulation der unterschiedlichen Transistoren bzw. der Traps, erst eine Analyse zu machen, welche Traps überhaupt bei dem aktuellen Szenario aktiviert werden können. D.h. bevor die eigentliche Simulation stattfindet, wird erst eine Worst-Case-Trap-Simulation durchgeführt. Bei dieser wird ermittelt, welche Traps eigentlich für dieses Szenario benötigt werden. Das Worst-Case-Szenario besteht aus der höchsten Temperatur, der höchsten Versorgungsspannung und der höchsten Signalwahrscheinlichkeit, die während der geplanten Alterungssimulation auftreten kann. Hierbei werden die globalen maximalen Werte genommen, diese müssen auch nicht zur selben Zeit auftreten. Das Vorgehen ist in Abbildung 4.10 dargestellt.



**Abbildung 4.10: Reduktion der Traps durch eine Worst-Case-Simulation: Alle Traps werden mit den Worst-Case-Werten aus dem aktuellen Szenario simuliert. Die Traps, die keine Auswirkungen auf die Schwellspannung haben, werden für dieses Szenario aus der Trapliste gelöscht.**

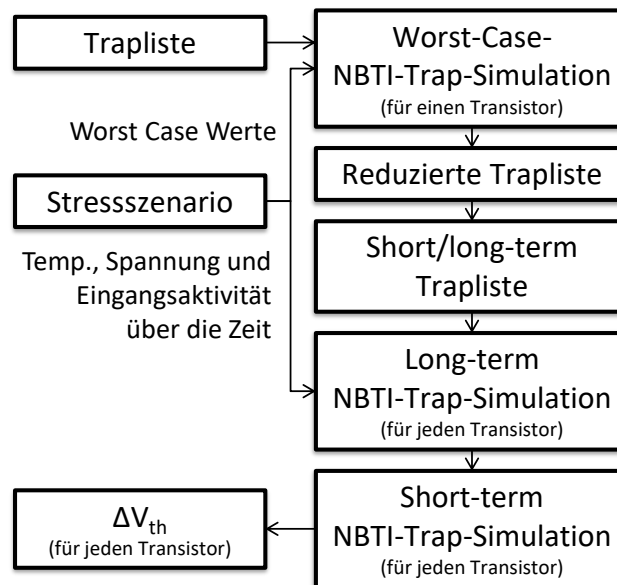
Anders als bei der short-term Simulation, wird an dieser Stelle jeder Trap aus der Trapliste einmal für den Worst-Case-Fall simuliert. Sollte ein Trap nach der simulierten Alterung keinen Beitrag oder nur einen sehr geringen zum Schwellspannungsschaden haben, so kann dieser Trap aus der Trapliste für dieses Szenario gestrichen werden. Dabei hat dieser Vorgang keinen Einfluss auf die spätere Simulation. Dieser Schritt muss nur einmal für jedes neue Szenario durchgeführt werden. Anschließend wird mit der reduzierten Trapliste jeder einzelne Transistor mit den Traps simuliert.

Tests haben ergeben, dass sich die zu simulierenden Traps, je nach Szenario, mit diesem Reduktionsschritt erheblich verringern lassen. Weitere Ergebnisse zur Reduktion der Traps sind im Kapitel *Evaluation* zu finden.

### Gesamter Ablauf

Zum Abschluss geht es nun um den Ablauf der NBTI-Alterungssimulation. Dieser ist in Abbildung 4.11 dargestellt. In einem ersten Schritt wird die Trapliste reduziert, wie dies in Abbildung 4.10 erläutert wurde. Anschließend wird die Trapliste unterteilt in short- und long-term Traps. Dies ist möglich, da die Traps bei der Berechnung des Schwellspannungsschadens nach Gleichung 4.5 unabhängig voneinander sind. Mit den long-term Traps wird eine Alterungssimulation durch ein vorher definiertes Stressszenario durchgeführt. Als letzter Schritt wird der Schwellspannungsschaden durch die short-term Traps zu den long-term Traps hinzugerechnet. Als Ergebnis erhält man den Schwellspannungsschaden für einen Transistor für ein bestimmtes Stressszenario.

In dem nächsten Kapitel wird das Stressszenario genauer definiert und erläutert, welche Informationen und Voraussetzungen für das Erstellen eines Szenarios für die Alterungssimulation benötigt werden.



**Abbildung 4.11: Ablauf der NBTI-Alterungssimulation:** Ausgehend von der Trapliste wird zuerst eine Reduzierung und anschließend eine Aufteilung der Traps durchgeführt. Die eigentliche Trap-Simulation nutzt dann die long-term Traps mit einem vorher definierten Stressszenario, das aus der Temperatur, der Spannung und der Eingangsaktivität an den Transistoren besteht. Nach der long-term Simulation wird eine short-term Simulation der Traps durchgeführt. Letztendlich erhält man aus der gesamten Simulation den Schwellspannungsschaden für die einzelnen Transistoren aus diesem Flow.

## 4.2 Missions- und Stressszenario Modell

### 4.2.1 Voraussetzungen

Ausgehend von dem vorgestellten NBTI-Modell wird in diesem Kapitel das Missionsszenario beschrieben. Das Missionsszenario legt fest, wie ein Design über die Zeit genutzt wird. Aus diesem Missionsszenario wird ein Stressszenario für die Alterungssimulation abgeleitet. Das Missionsszenario ist der Ausgangspunkt für die Alterungssimulation, da es ein Unterschied ist, wie ein System genutzt wird. Der in dieser Arbeit simulierte Alterungseffekt NBTI ist, wie bei der Vorstellung des Modells in dem letzten Kapitel bereits erläutert, temperatur-, versorgungsspannungs- und eingangsaktivitätsabhängig.

In Abbildung 4.12 sind die Temperaturen für Oldenburg aus dem Jahr 2014 zu sehen. Der Höchstwert liegt bei 31 °C und der Tiefstwerte bei -8 °C. D.h. ein System hat neben seinem inneren Zustand, in dem es sich befindet und woraus durch die Verlustleistung eine Erwärmung des Systems entsteht, auch äußere Einflussgrößen wie die Außentemperatur, die sich über die Zeit ändern kann.

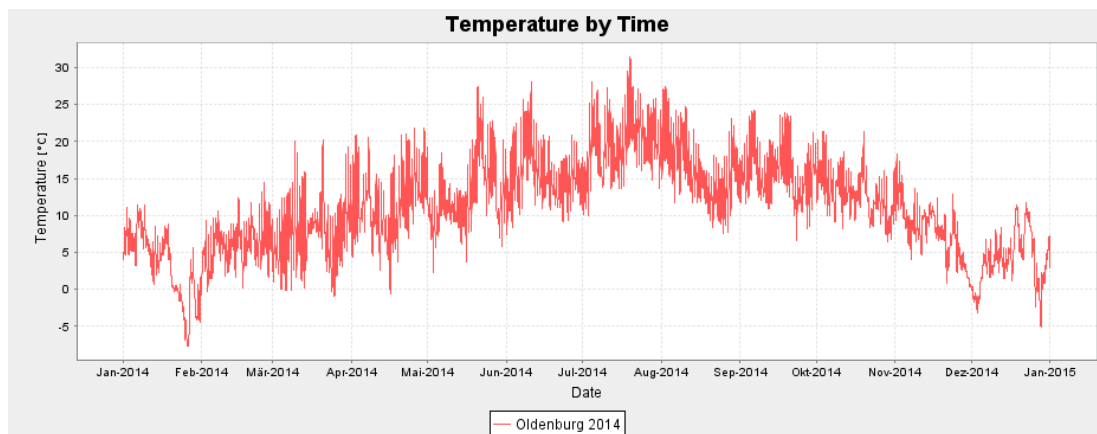
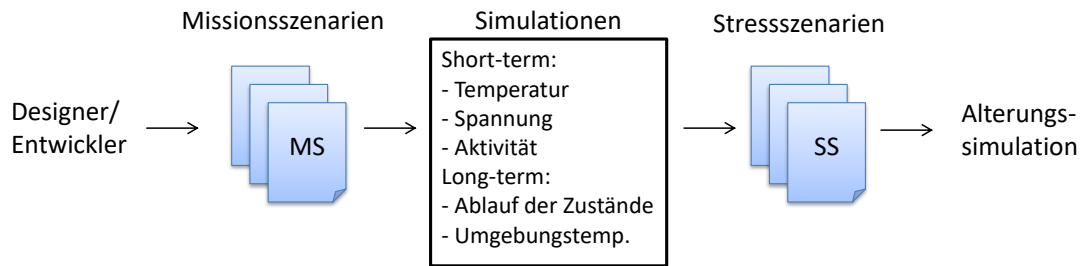


Abbildung 4.12: Temperaturverlauf in [°C] von Oldenburg im Jahr 2014. Der Screenshot stammt aus dem in dieser Arbeit entwickelten Missionsszenario-Editor; Quelle der Temperaturdaten [58].

### 4.2.2 Umsetzung

Um diesen Sachverhalt des externen Einflusses und dem Gewinnen des Stressszenarios abzubilden, muss der Entwickler des digitalen Systems in einem ersten Schritt ein Missionsszenario beschreiben. Das Missionsszenario kann wie folgt definiert werden:

$$MS = (S(t), T_{amb}(t)) \quad (4.9)$$



**Abbildung 4.13: Entstehung von Missionsszenarien und der daraus entstehenden Stressszenarien. Das Stressszenario ist der Input für die Alterungssimulation.**

Es besteht aus den Zuständen des Systems  $S$ , die sich über die Zeit ändern können und der Umgebungstemperatur  $T_{amb}$ , die ebenfalls zeitabhängig ist. Mit dem Missionsszenario wird dann, wie in Abbildung 4.13 dargestellt ist, durch verschiedene Modelle, die in den nächsten Abschnitten vorgestellt werden, ein Stressszenario erstellt. Bei der Simulation des Missionsszenarios können die zu simulierenden Größen in kurzfristige (short-term) und langfristige (long-term) unterteilt werden. Es wird betrachtet, welche Aktivität des Systems ausgeführt wird und wie die Temperatur und Versorgungsspannung in diesem Szenario ist.

Ausgehend von den verschiedenen short-term Szenarien wird dann ein long-term Szenario erstellt. Hierbei werden die short-term Szenarien in eine zeitliche Reihenfolge gebracht. Die einzelnen short-term Szenarien können beliebig häufig vorkommen, sowie in unterschiedlichen zeitlichen Längen und Reihenfolgen. Von der Gesamtlänge wird dann ein typischer Lebenszyklus des Designs simuliert, z.B. zehn Jahre. Zusätzlich kommt neben der zeitlichen Reihenfolge noch ein long-term Temperaturwert hinzu. Dieser Temperaturwert stellt die Umgebungstemperatur des Systems über das long-term Szenario dar und wird zu den short-term Temperaturwerten addiert (innere Temperatur, durch Selbsterwärmung).

Das Stressszenario ist die Eingabe für die Alterungssimulation und kann folgendermaßen definiert werden:

$$SS = (V_{DD}(t), T(t), P_x(t)) \quad (4.10)$$

Hierbei ist  $V_{DD}$  die Versorgungsspannung,  $T$  die Temperatur und  $P_x$  die Signalwahrscheinlichkeit. Dies wurde bereits in dem letzten Unterkapitel erläutert. In den nächsten Unterkapiteln wird auf die Entstehung der Stressszenarien eingegangen.

## 4.3 Thermisches und Package-Modell

### 4.3.1 Voraussetzungen

Die Temperatur ist eine der wichtigsten Einflussgrößen bei der Entwicklung zukünftiger digitaler Systeme. Zum einen kann sie durch die umgesetzte Verlustleistung das Design beschränken und damit auch die maximal genutzten aktiven Komponenten einschränken, zum anderen hat sie auch einen direkten Einfluss auf die Alterung. Dies wurde bereits in dem Degradations-Modell Kapitel gezeigt. Um eine thermische Simulation durchführen zu können, sind, wie im Kapitel zu den *Grundlagen* bereits vorgestellt wurde, grundsätzlich drei Informationen des zu simulierenden Systems nötig:

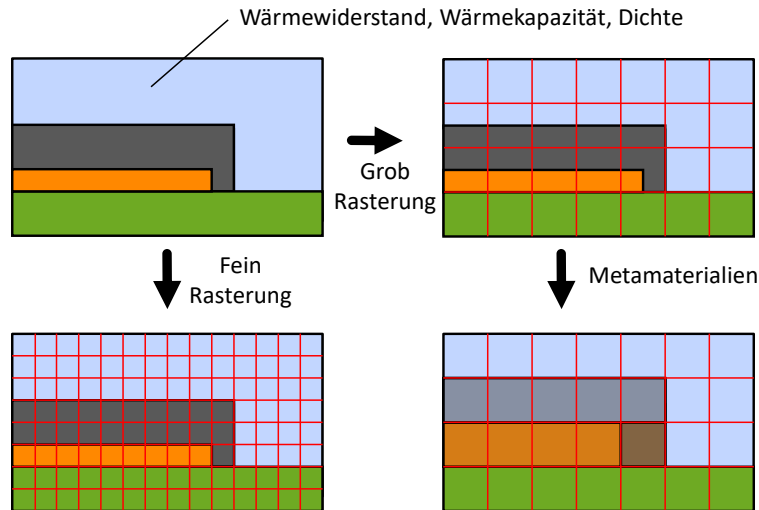
- 1) die umgesetzte Verlustleistung, die aus der dynamischen und statischen Verlustleistung besteht.
- 2) die räumliche Verteilung der Verlustleistung, d.h. wo wird sie umgesetzt und auch auf welcher Fläche.
- 3) die Information, wie das System physikalisch aufgebaut ist, d.h. genauer aus welchen Materialien das System besteht und wie es an die Umgebung angebunden ist.

Im Folgenden werden die drei oben aufgeführten Aspekte in den Unterkapiteln näher erläutert und anschließend wird der gesamte Ablauf der thermischen Simulation vorgestellt.

### 4.3.2 Umsetzung: Package-Modell

Das Package ist der physikalische Aufbau des thermisch zu simulierenden Systems. Es besteht aus den geometrischen Informationen und den Materialeigenschaften, die aus der Wärmeleitfähigkeit  $\kappa$ , der Dichte  $\rho$  und der Wärmekapazität  $C_p$  bestehen. Dies ist das Package-Modell. Diese Informationen werden durch einen Package-Editor erstellt und sind meistens eine 3-dimensionale Beschreibung des Systems. Der nächste Schritt ist eine Diskretisierung des 3-dimensionalen Packages. Diese Rasterung wird für die Simulation benötigt und hat einen direkten Einfluss auf die Simulationsgeschwindigkeit und Genauigkeit der Ergebnisse. Je nach der gewählten Rasterung, kann es auch zur Entstehung von Metamaterialien kommen, die aus verschiedenen Materialien bestehen, aber nur mit einem Satz Materialeigenschaften beschrieben werden sollen. Dies führt zur Entstehung von Metamaterialien. Diese Entstehung von Metamaterialien ist in Abbildung 4.14 dargestellt. In der Abbildung ist durch ein rotes Gitter die Rasterung des Packages zu sehen. Auf der rechten Seite der Abbildung kommt es durch die Wahl der groben Rasterung zu mehreren Blöcken, die aus unterschiedlichen Materialien bestehen, hieraus werden neue Metamaterialien erstellt.

Die Berechnung des neuen Metamaterials aus den alten Materialien, die sich in einem Rasterblock befinden, ist für die drei Materialeigenschaften unterschiedlich. Bei der Dichte  $\rho$  und der Wärmekapazität  $C_p$  wird abhängig von dem eingenommenen Volumen der Materialien ein



**Abbildung 4.14:** Ein kontinuierlich beschriebenes Package wird durch eine Rasterung diskretisiert. Hierbei kann es durch die Wahl der Rasterung (rotes Gitter), wie auf der rechten Seite dargestellt, zu neuen Metamaterialien mit anisotropen Materialeigenschaften kommen. Dies ist eine Veranschaulichung eines 2-dimensionalen Beispiels, typischerweise ist die Package-Beschreibung allerdings 3-dimensional.

Durchschnittswert gebildet. Dies ist in Gleichung 4.11 für die Dichte und in Gleichung 4.12 für die Wärmekapazität dargestellt.  $V_n$  ist die Gewichtung, wie viel Prozent des jeweiligen Materials in dem zu berechnenden Rasterblock ist.

$$\rho_M = \sum_{n=0}^N V_n \rho_n \quad (4.11)$$

$$C_{pM} = \sum_{n=0}^N V_n C_{pn} \quad (4.12)$$

Für die Berechnung der Wärmeleitfähigkeit  $\kappa$  müssen mehrere Dinge bei der Berechnung beachtet werden. Zum einen ist nicht nur das Volumen des jeweiligen Materials ausschlaggebend, sondern auch die Anordnung in dem Rasterblock, zum anderen können sich für die unterschiedlichen Richtungen des Materials andere  $\kappa$  Werte ergeben. Die Gleichungen 4.13 und 4.14 geben die Formeln zur Berechnung der neuen Wärmeleitfähigkeit an, diese sind identisch mit den bekannten Leitwerten-Formeln aus der Elektrotechnik zur Berechnung von Reihen- und Parallelschaltung<sup>6</sup> mit der Ausnahme, dass zusätzlich noch ein Volumenfaktor  $V_n$  benötigt wird.



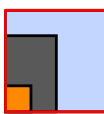
$$\kappa_M = \frac{1}{\sum_{n=0}^N V_n \frac{1}{\kappa_n}} \quad (\text{Reihenschaltung}) \quad (4.13)$$


$$\kappa_M = \sum_{n=0}^N V_n \kappa_n \quad (\text{Parallelschaltung}) \quad (4.14)$$

Dieser Sachverhalt der Berechnung wird anhand dreier Beispiele genauer erläutert. In

<sup>6</sup>Taschenbuch der Elektrotechnik, Kories[41],(1.33) und (1.35)

Abbildung 4.15 sieht man drei Rasterblöcke, die aus unterschiedlichen Materialien bestehen, wodurch es zur Bildung von Metamaterial kommt und neue Leitwerte berechnet werden müssen.

1.)		$\kappa_{x1} = \frac{1}{2}\kappa_a + \frac{1}{2}\kappa_b$ $\kappa_{y1} = \frac{1}{\frac{1}{2}\frac{1}{\kappa_a} + \frac{1}{2}\frac{1}{\kappa_b}} = 2 \frac{\kappa_a \kappa_b}{\kappa_a + \kappa_b}$	$\rho_1 = \frac{1}{2}\rho_a + \frac{1}{2}\rho_b$ $c_{p1} = \frac{1}{2}c_{pa} + \frac{1}{2}c_{pb}$
2.)		$\kappa_{x2} = \frac{1}{2}\kappa_b + \frac{1}{2}\frac{1}{\frac{1}{2}\frac{1}{\kappa_c} + \frac{1}{2}\frac{1}{\kappa_b}} = \kappa_{y2}$	$\rho_2 = \frac{3}{4}\rho_b + \frac{1}{4}\rho_c$ $c_{p2} = \frac{3}{4}c_{pb} + \frac{1}{4}c_{pc}$
3.)		$\kappa_{x3} = \frac{1}{\frac{1}{2}\frac{1}{\frac{1}{2}\frac{1}{\kappa_{x1}} + \frac{1}{2}\frac{1}{\kappa_a}} + \frac{1}{2}\frac{1}{\kappa_b}}$ $\kappa_{y3} = \frac{1}{\frac{1}{2}\frac{1}{\frac{1}{2}\frac{1}{\kappa_{y1}} + \frac{1}{2}\frac{1}{\kappa_{y2}}} + \frac{1}{2}\kappa_a}$	$\rho_3 = \frac{10}{16}\rho_a + \frac{5}{16}\rho_b + \frac{1}{16}\rho_c$ $c_{p3} = \frac{10}{16}c_{pa} + \frac{5}{16}c_{pb} + \frac{1}{16}c_{pc}$


Material: a b c

**Abbildung 4.15: Beispiele zur Berechnung der Materialeigenschaften von Metamaterialien.** Die Rasterblöcke sind aus Abbildung 4.14 entnommen. Das 1. Beispiel besteht in x-Richtung aus einer Parallelschaltung und in y-Richtung aus einer Reihenschaltung von zwei Materialien. Das 2. Beispiel hat in x- und y-Richtung denselben Leitwert. Das 3. Beispiel besteht aus mehr als zwei Materialien, wodurch die Berechnungen komplexer werden.

### 4.3.3 Umsetzung: Thermisches Modell

Neben den erwähnten Lösungswegen für die thermische Gleichung, wie sie im Kapitel zu den *Grundlagen* beschrieben wird, gibt es auch noch den Ansatz, die Gleichung über die Green-Funktion zu lösen. Die Methode der Green-Funktion ist ein nach Georg Green benanntes Verfahren. Es wird zur Lösung inhomogener Differentialgleichungen genutzt. Dieser Green'sche Lösungsansatz wird z.B. in der Elektrotechnik bei der Systemtheorie eingesetzt. Ein lineares zeitinvariantes System (LZI-System)<sup>7</sup> hat, wie der Name schon nahelegt, ein lineares Verhalten, d.h. das Superpositionsprinzip gilt und das Verhalten am Ausgang ist nicht von der absoluten Zeit abhängig. Dabei gilt: Wenn  $f(t)$  eine Lösung für die Umgebungsbedingung  $U(t)$  ist, dann ist  $f(t + T)$  auch eine Lösung für  $U(t + T)$  wobei  $\forall T \in \mathbb{R}$  (zeitinvariant).

Aufgrund dieser Eigenschaften kann das System mit einer Impulsfunktion angeregt werden und das Ergebnis am Ausgang ist für das System eine charakteristische Funktion (Impulsantwort). Aus dieser Impulsantwort kann mit einer Faltung, mit einem beliebigen Eingangssignal, das neue Ausgangsverhalten berechnet werden. Dieser Sachverhalt ist in Gleichung<sup>8</sup> 4.15 dargestellt.

$$y(t) = \int_{-\infty}^{\infty} x(\tau)g(t - \tau)d\tau = x(t) * g(t) \quad (4.15)$$

Hierbei ist  $g(t)$  die Impulsantwort des Systems, dass durch einen Impuls am Eingang des Systems erzeugt wurde und  $x(t)$  ist das neue Eingangssignal. Wobei  $*$  der Faltungsoperator ist.

<sup>7</sup>LTI-System (englisch linear time-invariant system)

<sup>8</sup>Taschenbuch der Elektrotechnik, Kories[41], (6.28)



Dieses bekannte Vorgehen aus der Elektrotechnik kann auch zur Lösung der thermischen Gleichung genutzt werden, da es auch ein LZI-System ist. Im Kapitel zu den *Grundlagen* wurde bereits vorgestellt, dass ein thermisches System als RC-Modell dargestellt werden kann. Da von der Simulation eine zeitdiskrete Simulation stattfinden soll, ergibt sich die Gleichung<sup>9</sup> 4.16 zur Berechnung der diskreten Faltung.

$$y(z) = \sum_{v=0}^n f_v g_{n-v} \quad (4.16)$$

In Abbildung 4.16 ist ein thermisches System dargestellt, das aus zwei Komponenten K1 und K2 besteht. Das System ist ein LZI-System. Die zwei Blöcke können, wie in Abbildung 4.17, als LZI-System dargestellt werden.

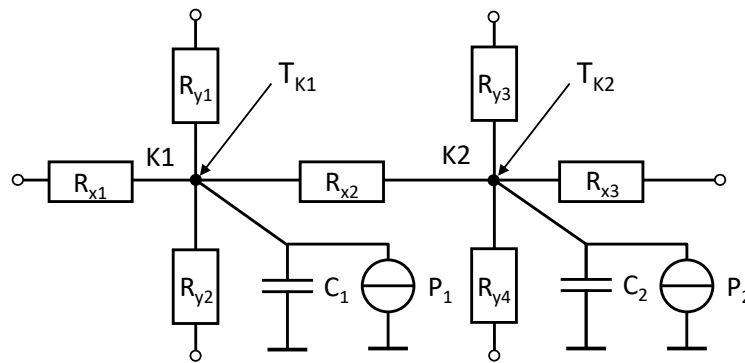


Abbildung 4.16: Abgebildet sind zwei aktive thermische Blöcke ( $K1$  und  $K2$ ), die über den Widerstand  $R_{x2}$  verbunden sind. Die Darstellung ist in diesem Beispiel 2-dimensional, typischerweise kommen für den 3-dimensionalen Fall noch zusätzliche Widerstände hinzu.  $P_x$  ist die Energiequelle, die im Fall von digitalen Systemen die Verlustleistung der Transistoren ist.  $T_{k1}$  und  $T_{k2}$  ist die Temperatur der Blöcke, die berechnet werden soll.

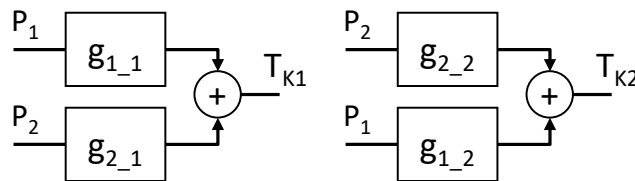


Abbildung 4.17: Darstellung der thermischen Blöcke aus Abbildung 4.16 als ein LZI-System. Die Terme  $g_{x_y}$  sind die thermischen Impulsantworten der Blöcke, wobei  $g_{1_1}$  die Impulsantwort von  $P1$  auf die Komponente  $K1$  und  $g_{1_2}$  die Impulsantwort von  $P2$  auf Komponente  $K2$  ist. Für die Darstellung des zweiten Blocks gilt das gleiche Schema.

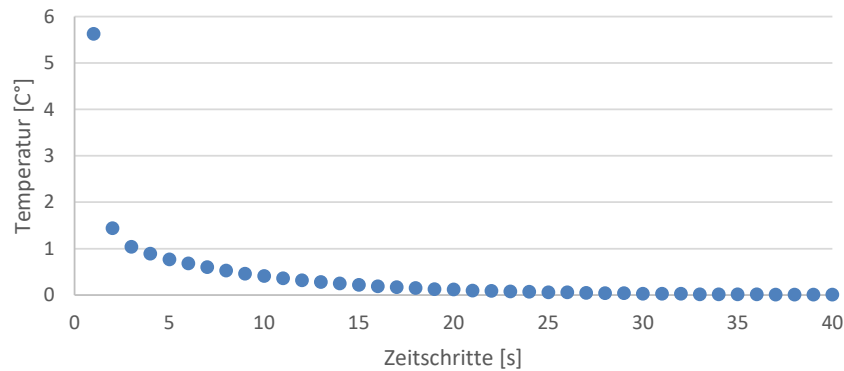
Mit Hilfe der Gleichung 4.16 kann die Temperatur der Komponenten aus den thermischen Impulsantworten und der gegebenen Leistungswerte durch diskrete Faltungen berechnet werden. In Gleichung 4.17 und 4.18 ist dies für die beiden Komponenten aus Abbildung 4.17 dargestellt.

<sup>9</sup>Taschenbuch der Mathematik, Bronstein[6], (15.119)

$$T_{K1} = P_1 * g_{1\_1} + P_2 * g_{2\_1} \quad (4.17)$$

$$T_{K2} = P_2 * g_{2\_2} + P_1 * g_{1\_2} \quad (4.18)$$

Die Impulsantworten für die unterschiedlichen Komponenten werden im Folgenden auch als thermische Charakterisierung bezeichnet. In Abbildung 4.18 ist das Ergebnis einer solchen thermischen Impulsantwort dargestellt.



**Abbildung 4.18: Diskrete Impulsantwort einer Komponente: Die Temperaturwerte werden diskret für eine bestimmte Schrittweite gespeichert. In diesem Beispiel ist die Schrittweite der Simulation eine Sekunde.**

Ausgehend von diesem Beispiel, bestehend aus zwei Komponenten, kann das Vorgehen auf eine beliebige Anzahl von Komponenten übertragen werden. Es muss für jede der Komponenten des zu simulierenden Systems die thermische Impulsantwort simuliert werden. Hierbei muss sowohl die direkte Impulsantwort, d.h. die Wirkung der angeregten Leistung auf die Komponente, als auch die Impulsantworten auf die übrigen Komponenten, simuliert und gespeichert werden. Mit Hilfe dieser Charakterisierung kann dann mit Gleichung 4.19 die Temperatur für jede Komponente berechnet werden.

$$T_{Kx} = T_{Kx,x} + T_{Kx,\dots} = P_x * g_{x\_x} + \sum_{y \in \{i \in N | i \neq x\}} P_y * g_{y\_x} \quad (4.19)$$

Die Terme der Gleichung bedeuten, dass die Temperatur einer Komponente sich zusammensetzt aus der Eigenerwärmung  $T_{Kx,x}$  und die Erwärmung der anderen Komponenten  $T_{Kx,\dots}$ . Der Teil auf der rechten Seite der Gleichung berechnet dies dann mit der Faltung aus den Impulsantworten und den Leistungswerten der Komponenten.

Im nun folgenden Abschnitt wird die Berechnung der Verlustleistung vorgestellt, die direkten Einfluss auf die Temperaturberechnung hat.

## 4.4 Verlustleistungs-Modell

### 4.4.1 Voraussetzungen

Im dem Kapitel zu den *Grundlagen* wurde bereits die Entstehung der Verlustleistung erläutert sowie der Aspekt, dass die Verlustleistung sich aus der dynamischen und der statischen Verlustleistung zusammensetzt. In diesem Unterkapitel werden nun die Modelle, die in dieser Arbeit genutzt werden, beschrieben. Es gibt zu dem Thema Verlustleistung eine Fülle von Veröffentlichungen. Diese beschäftigen sich mit der Berechnung und der Abstraktion der Verlustleistung für die unterschiedlichen Abstraktionsebenen, wie der physikalischen Ebene, wie Transistoren- über Gatterebene bis zu abstrakteren Darstellungen auf Systemebene. In dieser Arbeit ist der Ausgangspunkt, dass bereits eine Berechnung der Verlustleistung mit einem der Verfahren durchgeführt wurde. Im Kapitel zur *Implementierung* wird kurz auf die durch Industrietools unterstützte Generierung der Verlustleistung eingegangen.

### 4.4.2 Umsetzung

#### Dynamische Verlustleistung

Die Gleichung zur Berechnung der dynamischen Verlustleistung wurde im Kapitel zu den *Grundlagen* in Gleichung 2.4 aufgeführt. Die Gleichung hat eine Abhängigkeit von der Schaltaktivität und eine quadratische von der Versorgungsspannung. Für die Zwecke in dieser Arbeit ist besonders die quadratische Abhängigkeit von der Versorgungsspannung von Interesse. Es gibt keine Abhängigkeit von der Temperatur, diese ist rein auf die statische Verlustleistung beschränkt. Bei einer gegebenen dynamischen Verlustleistung und der Versorgungsspannung kann aus Gleichung 2.4 die neue Verlustleistung mit den Gleichungen 4.20 und 4.21 berechnet werden.

$$C_{dyn,ref} = \frac{P_{dyn,ref}}{V_{DD,ref}^2} \quad (4.20)$$

$$P_{dyn}(V_{DD}) = C_{dyn,ref} \cdot V_{DD}^2 \quad (4.21)$$

Hierbei ist  $P_{dyn,ref}$  die bereits vorhandene dynamische Verlustleistung für die Versorgungsspannung  $V_{DD,ref}$ . Aus  $C_{dyn,ref}$  und der neuen Versorgungsspannung  $V_{DD}$  kann dann die neue dynamische Verlustleistung berechnet werden.

#### Statische Verlustleistung

Die statische Verlustleistung setzt sich, wie in dem Kapitel zu den *Grundlagen* erklärt wurde, aus vielen unterschiedlichen Leakage-Varianten zusammen und ist primär von der Technologie abhängig. Besonders ist hier auch der unterschiedlich starke Einfluss der Temperatur hervorzuheben, was letztendlich zu der Problematik der elektrothermischen Kopplung führt. Diese elektrothermische Kopplung wird in einem späteren Abschnitt genauer beschrieben. Die

statische Verlustleistung wird in dieser Arbeit durch vorberechnete Lookup-Tabellen umgesetzt. In Gleichung 4.22 ist dies dargestellt. Durch Lookup-Tabellen werden die Werte für die statische Verlustleistung abhängig von dem Gattertyp, der Temperatur  $T$ , der Versorgungsspannung  $V_{DD}$  und der Schwellspannung  $V_{th}$  ausgegeben.

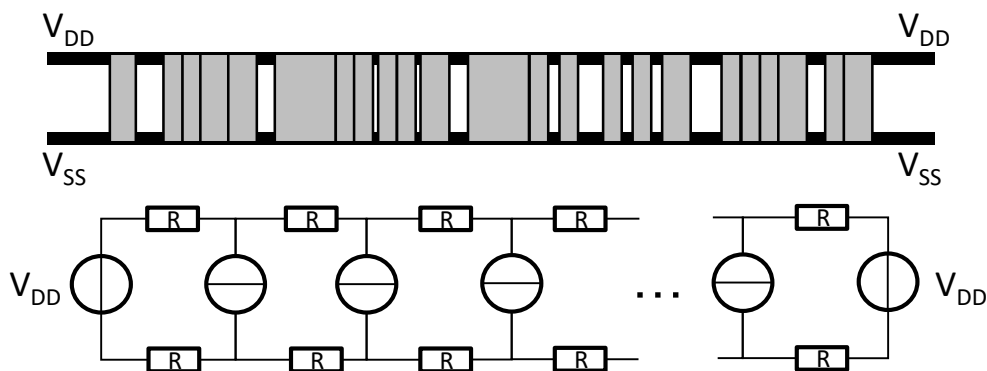
$$P_{total\_leak} = Lookup\_tabelle(Gattertyp, T, V_{DD}, V_{th}) \quad (4.22)$$

Im folgenden Unterkapitel wird auf den IR-Drop eingegangen, der die dynamische Verlustleistung beeinflusst.

## 4.5 IR-Drop-Modell

### 4.5.1 Voraussetzungen

Durch die kleineren Strukturgrößen der Transistoren werden auch die Versorgungsleitungen reduziert. Dadurch nehmen die Leitungswiderstände dieser Strukturen zu. Wie im Kapitel zu den *Grundlagen* bereits erläutert wurde, kommt es durch einen stromdurchflossenen Widerstand zu einem Spannungsabfall an dem Widerstand, der IR-Drop genannt wird. Dies geschieht auch in den Versorgungsleitungen einer digitalen Schaltung. Da durch die Versorgung der einzelnen Gatter ein nicht unerheblicher Strom im Verhältnis zu den Widerständen der Versorgungsleitungen fließt, entsteht ein Spannungsabfall an diesen Versorgungsleitungen, der eine Verringerung der Versorgungsspannung bewirkt, welche die Gatter letztendlich erreicht. In Abbildung 4.19 ist im oberen Teil der Aufbau eines Versorgungsnetzes, bestehend aus den Leitungen  $V_{DD}$  und  $V_{SS}$ , sowie den einzelnen Gattern in Grau, zu sehen.



**Abbildung 4.19:** Oben: Abbildung eines Versorgungsnetzes mit mehreren unterschiedlichen Gattern mit den Versorgungsleitungen  $V_{DD}$  und  $V_{SS}$ ; unten: Ersatzschaltung des Versorgungsnetzes, wobei die einzelnen Gatter als Stromquellen und die Leitungen als Widerstände dargestellt sind.

Im unterem Teil ist der Aufbau einer Ersatzschaltung zu sehen. Hierbei sind die Versorgungsleitungen als Widerstände zwischen den einzelnen Komponenten dargestellt und die Gatter sind Stromquellen, die abhängig von der Versorgungsspannung, der Temperatur und ihrer Aktivität einen unterschiedlichen eingepprägten Strom aufweisen. Wichtig ist hierbei auch, dass sowohl die

Versorgungsleitungen  $V_{DD}$ , als auch die  $V_{SS}$  Leitung einen Widerstand aufweisen und berücksichtigt werden müssen. Je nach Größe und Aufwand des Designs können neben der Versorgung von rechts und links auch noch zusätzliche globale Versorgungsleitungen von oben und unten hinzukommen.

### 4.5.2 Umsetzung

Zur Berechnung der Schaltung aus Abbildung 4.19 können z.B. vorhandene Schaltungssimulationstools genutzt werden wie z.B. SPICE. Da es bei der Simulation zu einer großen Anzahl von Gattern kommt, die auch über hunderttausend liegen kann, sollte die Berechnung des IR-Drop möglichst performant laufen, außerdem ist durch die elektrothermische Kopplung ein häufiges Neuberechnen des IR-Drops notwendig.

Ein Ansatz, der sich anbietet, um die Versorgungsnetzschaltung zu berechnen, ist der Überlagerungsansatz oder auch Superpositionsansatz[42]. Bei diesem Ansatz wird jede Spannungs- und Stromquelle unabhängig voneinander betrachtet und die jeweiligen Ströme durch die Widerstände werden separat berechnet. Die in jedem Schritt nicht betrachteten Stromquellen werden als offene Leitungen und Spannungsquellen als Kurzschlüsse behandelt. Dieser Ansatz berechnet die exakte Lösung der Ströme/Spannungen in der Schaltung und ist keine Heuristik, d.h. bei der Überprüfung im Kapitel *Evaluation* sollten sehr geringe Abweichungen im Vergleich zur SPICE-Simulation auftreten.

In Abbildung 4.20 ist ein Beispiel, bestehend aus zwei Stromquellen und zwei Spannungsquellen, dargestellt. Mit Hilfe dieses Beispiels werden im Folgenden die Gleichungen zur Berechnung des IR-Drops hergeleitet:

In einem ersten Schritt (Abbildung 4.20 a) wird jede Spannungs- und Stromquelle in der Schaltung einzeln betrachtet. In der Abbildung 4.20 zeigt die rechte Seite die betrachteten Stromquellen und auf der linken Seite die Spannungsquellen. Hierbei werden - wie bereits erwähnt - die anderen Quellen durch Kurzschlüsse (Spannungsquelle) und offene Leitungen (Stromquelle) ersetzt. D.h. aus einer Schaltung, die aus  $N$  Quellen besteht, wird eine Berechnung auf  $N$  Teilrechnungen aufgeteilt. Durch Teilschaltungen hat sich das Berechnen der Ströme durch die Widerstände erheblich vereinfacht, da die Widerstände zusammengerechnet werden können. Die Widerstände bilden eine Reihenschaltung und werden wie in Gleichung 4.23 jeweils für den rechten und linken Teil der Schaltung addiert.

$$R_{links/rechts} = \sum_{links/rechts} R_x \quad (4.23)$$

Die Ströme in den Schaltungen teilen sich bei der Stromquelle in einen Strom, der nach links fließt und einen, der nach rechts fließt, auf. Die Schaltung ist ein einfacher Stromteiler (Abbildung 4.20 b linke Seite). Hieraus ergeben sich für die beiden Ströme die Gleichungen 4.24 und 4.25.

$$I_{Lx} = \frac{R_{rechts}}{R_{links} + R_{rechts}} I_x \quad (4.24)$$

$$I_{Rx} = \frac{R_{links}}{R_{rechts} + R_{links}} I_x \quad (4.25)$$

Bei den Spannungsquellen ergibt sich die rechte Schaltung in Abbildung 4.20 b, die Schaltung besteht nur noch aus der Spannungsquelle und einem Widerstand. Der Gesamtwiderstand ist eine Reihenschaltung aus den einzelnen Widerständen (Gleichung 4.26).

$$R_{gesamt} = \sum R \quad (4.26)$$

Der Strom, der durch den Widerstand fließt, lässt sich durch das Ohm'sche Gesetz, wie es in Gleichung 4.27 steht, berechnen.

$$I_{Vx} = \frac{V_{DD}}{R_{gesamt}} \quad (4.27)$$

Nachdem alle Ströme durch die Widerstände berechnet sind, können nun die Ströme der Ursprungsschaltung durch Superposition bestimmt werden. Bei dieser Berechnung heben sich die Ströme, die durch die beiden Spannungsquellen fließen, auf (4.20 c), da sie gleich sind und in entgegengesetzte Richtungen fließen.

In einem letzten Schritt wird aus dem Widerstand und dem Strom durch das Ohm'sche Gesetz die Spannung über dem Widerstand berechnet, was letztendlich der IR-Drop des einzelnen Widerstandes ist. Zur Berechnung der verringerten Versorgungsspannung müssen nun noch die einzelnen Spannungen über den Widerständen abgezogen werden, die bis zu jedem einzelnen Gatter liegen (Gleichung 4.28). Dies sind sowohl die Spannungen über den Widerständen auf der  $V_{DD}$  Leitung als auch die Spannungen über den Widerständen der  $V_{SS}$  Leitung.

$$V_{irdrop(n)} = V_{DD} - \sum_{i=0}^{n-1} R_{con} \cdot I_i(V_i) \quad (4.28)$$

Dieser Ansatz ist ideal für die parallele Ausführung geeignet, um so die Berechnung zu beschleunigen. Es kann bei einer Anzahl von  $N$  Gattern in einem Versorgungsnetz die Berechnung in  $N$  Teilschaltungen aufgeteilt werden. Sobald diese Berechnungen abgeschossen sind, können durch Superposition die Gesamtströme und hiermit auch die Spannungen berechnet werden.

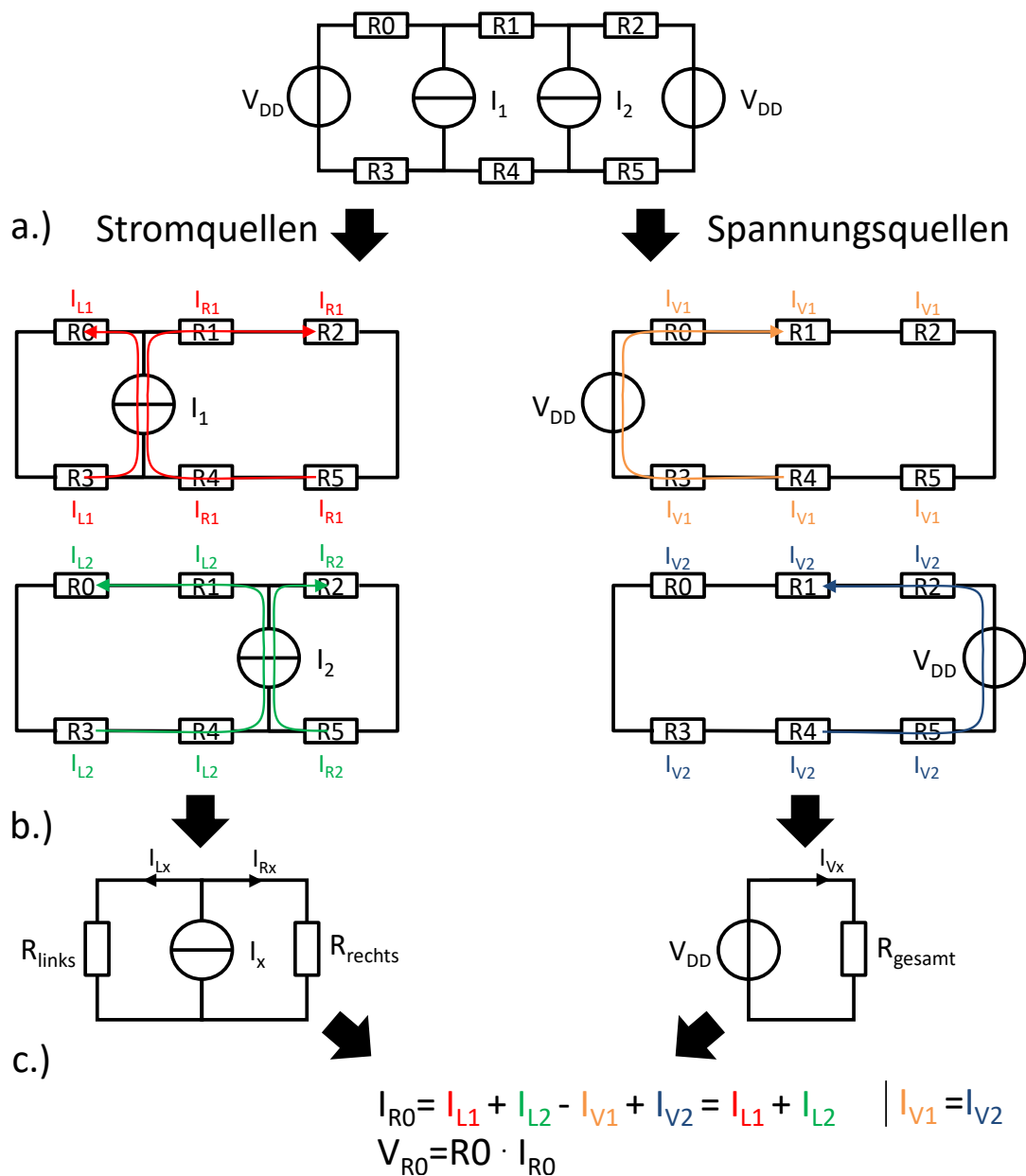


Abbildung 4.20: Superpositionsansatz zur Berechnung der Ströme in der obigen Schaltung, die aus zwei Spannungsquellen und zwei Stromquellen besteht. Beim Superpositionsansatz werden die Spannungs-/Stromquellen unabhängig voneinander betrachtet, hierbei werden die nicht betrachteten Stromquellen durch offene Leitungen und die Spannungsquellen durch Kurzschlüsse ersetzt. Um auf den Wert des Stromes durch einen Widerstand zu kommen, werden die einzelnen Ströme zum Gesamtstrom addiert (Superposition), wobei auf die Stromrichtungen der Ströme geachtet werden muss: Die Ströme der beiden Spannungsquellen heben sich z.B. auf, da sie gleich sind und in gegensätzliche Richtungen fließen.

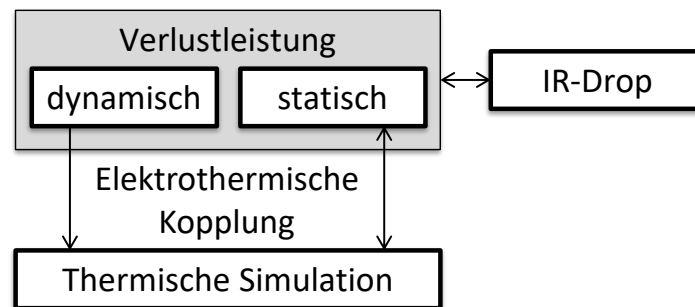
## 4.6 Elektrothermische Kopplung

### 4.6.1 Voraussetzungen

Wie bereits im Abschnitt über die Verlustleistung genauer erläutert wurde, ist die statische Verlustleistung temperaturabhängig. Durch die Berechnung der Temperatur aus der Verlustleistung entsteht hier eine Kopplung, die elektrothermische Kopplung genannt wird.

### 4.6.2 Umsetzung

In Abbildung 4.21 ist der Zusammenhang zwischen Temperatur und Verlustleistung noch einmal als Kopplung eingetragen. Zusätzlich zur Temperaturabhängigkeit kommt noch die Spannungsabhängigkeit bei der statischen und dynamischen Verlustleistung hinzu. D.h. wie neben der elektrothermischen Kopplung der Abbildung dargestellt ist, beeinflussen auch Spannungsveränderungen durch einen IR-Drop die Verlustleistung und damit auch die Temperatur des Systems.



**Abbildung 4.21:** In der Abbildung ist der Zusammenhang zwischen der thermischen Simulation und der Verlustleistung dargestellt, die als elektrothermische Kopplung verstanden wird. Zusätzlich zur elektrothermischen Kopplung hat die Versorgungsspannung einen Einfluss auf die Verlustleistung und damit auch auf die Temperatur.

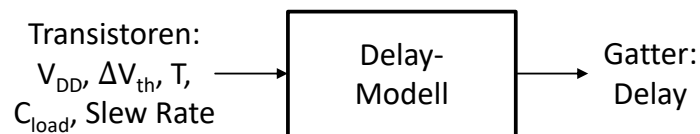
Im nächsten Unterkapitel wird das Delay der Gatter untersucht. Dabei wird ein Modell entwickelt, das zur Berechnung des gealterten Delays genutzt wird.



## 4.7 Delay-Modell

### 4.7.1 Voraussetzungen

In diesem Kapitel wird der Zusammenhang zwischen den alternden Transistoren innerhalb eines Gatters und der Berechnung des Gatter-Delays hergestellt. Bei der NBTI-Alterungssimulation des Designs altern alle Transistoren unterschiedlich. Abhängig ist dies z.B. von Temperatur, Spannung und Eingangsaktivität der Transistoren. In Abbildung 4.22 ist die Anforderung an das Delay-Modell dargestellt. Mit den Parametern der einzelnen Transistoren, die zusammen ein Gatter ergeben, soll auf das Delay des Gatters geschlossen werden.



**Abbildung 4.22: Anforderungen an das Delay-Modell: Das Modell soll mit Hilfe der Ergebnisse aus der Alterungssimulation (also dem Schwellspannungsschaden) und den Parametern Versorgungsspannung, Temperatur, Slew Rate und kapazitive Ausgangslast das Delay der einzelner Gatter berechnet werden.**

Zusätzlich soll in diesem Unterkapitel die Anzahl der zu alternden Transistoren auf die wesentlichen reduziert werden, um die Simulation weiter zu beschleunigen und um das Untersuchen von größeren Designs zu ermöglichen. Deshalb stellt sich die wichtige Frage, welche Transistoren letztendlich betrachtet werden müssen, die auf die Funktion einer Schaltung (bei Alterung dieser) eine Auswirkung haben? Bereits im Kapitel zu den *Grundlagen* wurde geschrieben, dass die Anzahl der zu betrachteten Transistoren auf die beschränkt werden kann, die auf einem kritischen Pfad liegen. Hierbei ist es so, dass es nicht nur einen kritischen Pfad gibt, sondern es werden je nach Design, mehrere Pfade existieren, die ein ähnliches Delay aufweisen. Durch die unterschiedliche Alterung der Transistoren kann es vorkommen, dass ein Pfad bei einem nicht gealterten System nicht der kritischste Pfad ist, der später durch die Alterung dann zu dem mit dem höchstem Delay wird.

Von daher wird im ersten Unterkapitel auf die eigentliche Delay-Berechnung eingegangen. Anschließend wird untersucht, welche Transistoren innerhalb eines Gatters altern und welche Informationen von der Transistorebene zur Gatterebene benötigt werden. Im Anschluss wird die Kritische-Pfad-Analyse vorgestellt, die die betrachteten Pfade auf die einschränkt, die durch die Alterung zu einem kritischen Pfad werden können.

### 4.7.2 Umsetzung

#### Gatter-Modell

In Abbildung 4.23 ist auf der linken Seite der Aufbau eines AND-Gatters abgebildet. Es besteht aus drei NMOS- und drei PMOS-Transistoren. Der Alterungseffekt NBTI betrifft nur die PMOS-Transistoren, die in der Abbildung 4.23 rot hervorgehoben sind. Für die Simulation wird

unter anderem die Signalwahrscheinlichkeit für die einzelnen Transistoren benötigt, die auch für die Berechnung der dynamischen Verlustleistung verwendet wird und auf dieselbe Art berechnet werden kann[33]. Bei den Transistoren an den Eingängen ist dies die Signalwahrscheinlichkeit des Eingangssignals, wie dies auch im Degradations-Modell Kapitel erläutert wurde. Wie in der Abbildung 4.23 zu sehen ist, liegt der Eingang von Transistor T3 auf einem internen Netz innerhalb des Gatters. Das bedeutet, die Berechnung der Signalwahrscheinlichkeit für dieses interne Netz wird aus den an den Eingängen des Gatters liegenden Signalwahrscheinlichkeiten durchgeführt. Das AND-Gatter kann als ein Hintereinanderschalten eines NAND-Gatters und eines Inverters aufgefasst werden, wie dies in der Abbildung 4.23 auf der linken Seite abgebildet ist.

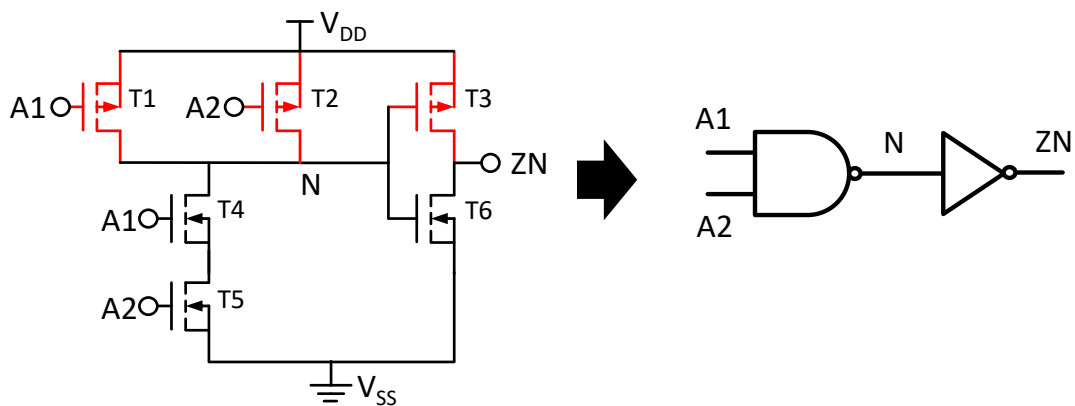


Abbildung 4.23: Dargestellt ist ein AND-Gatter mit zwei Eingängen, das in eine NAND-Gatter und einen anschließenden Inverter aufgeteilt werden kann. Die für die NBTI-Alterung relevanten PMOS-Transistoren sind rot dargestellt. Sind die Signalwahrscheinlichkeiten für die Signale  $A1$  und  $A2$  vorhanden, so kann mit Hilfe der Formel aus Tabelle 4.1 die Signalwahrscheinlichkeit für das interne Signal  $N$  berechnet werden, das für die Alterung des Transistors  $T3$  benötigt wird.

Mit Hilfe von den Berechnungsformeln aus Tabelle 4.1 können die Signalwahrscheinlichkeiten für einzelne Gatter berechnet werden.

Gattername	Signalwahrscheinlichkeit
INV	$1 - p_A$
AND	$p_A \cdot p_B \dots = \prod p_x$
NAND	$1 - \prod p_x$
OR	$1 - \prod \overline{p_x}$
NOR	$\prod \overline{p_x}$

Tabelle 4.1: Berechnung der Signalwahrscheinlichkeit am Ausgang von unterschiedlichen Gattertypen;  $p_x$  stellt die einzelnen Eingänge eines Gatters dar.

Die Herleitung der Formeln ist wie folgt: ein Inverter, wie der Name bereits sagt, invertiert das Eingangssignal, d.h. die Signalwahrscheinlichkeit ist eins minus der Eingangssignalwahrscheinlichkeit. Bei dem AND-Gatter werden die Eingangs-Signalwahrscheinlichkeiten miteinander multipliziert, wie dies aus der Wahrscheinlichkeitsrechnung bekannt ist<sup>10</sup>. Ein NAND-Gatter ist ein AND-Gatter mit einem invertierten Ausgang, folglich kann durch das Nutzen der beiden Formeln vom AND-Gatter und Inverter die Gleichung des NAND Gatters berechnet werden. Das OR-Gatter kann aus dem NAND-Gatter mit der Hilfe von der De Morganschen Regel<sup>11</sup> berechnet werden. Ein NOR kann durch Invertierung der Eingänge des OR berechnet werden.

In Tabelle 4.2 ist ein Beispiel für das AND-Gatter aus Abbildung 4.23 durchgeführt worden.

Transistorname	Berechnung	Signalwahrscheinlichkeit
T1	P1	0.2
T2	P2	0.4
T3	=1-P1*P2	0.92
T4	P1	0.2
T5	P2	0.4
T6	=1-P1*P2	0.92

**Tabelle 4.2: Signalwahrscheinlichkeiten der Eingänge und der einzelnen Transistoren eines AND-Gatters mit zwei Eingängen. Der Aufbau entspricht einem NAND-Gatters mit anschließendem Inverter. Bei diesem Beispiel haben die Eingänge A1 eine Signalwahrscheinlichkeit  $P1 = 0.2$  und  $P2 = 0.4$  für A2; für NBTI sind nur die Transistoren T1,T2 und T3 von Interesse.**

Mit diesem Vorgehen kann für jeden Transistor eines Gatters, falls der Transistor nicht ein Eingang ist, die Signalwahrscheinlichkeiten berechnet werden. Diese interne Berechnung der Signalwahrscheinlichkeiten wird für alle Gatter durchgeführt, für die eine Alterungssimulation durchgeführt werden soll.

Zusätzlich muss bei in Reihe geschalteten Transistoren berücksichtigt werden, dass sie je nach Position innerhalb der Reihenschaltung unterschiedlich altern. Dies ist auf den Umstand zurückzuführen, dass PMOS-Transistoren bei NBTI altern, wenn eine Spannung zwischen Source-/Gate-Anschluss anliegt (für den Schaltvorgang wird eine Spannung von  $V_{GS} < V_{th}$  benötigt) und zusätzlich der Transistor-Kanal in Inversion<sup>12</sup> ist. Hierzu muss der Source-Eingang des Transistors auf  $V_{DD}$  liegen. Durch die Reihenschaltung von PMOS-Transistoren kann es nun dazu kommen, dass einer der näher zu  $V_{DD}$  liegenden Transistoren ausgeschaltet ist und somit alle darunter liegenden Transistoren von  $V_{DD}$  getrennt sind und nicht altern. Dieser Sachverhalt ist vergleichbar mit Power Gating. Dies kann zur Reduktion der statischen Verlustleistung genutzt

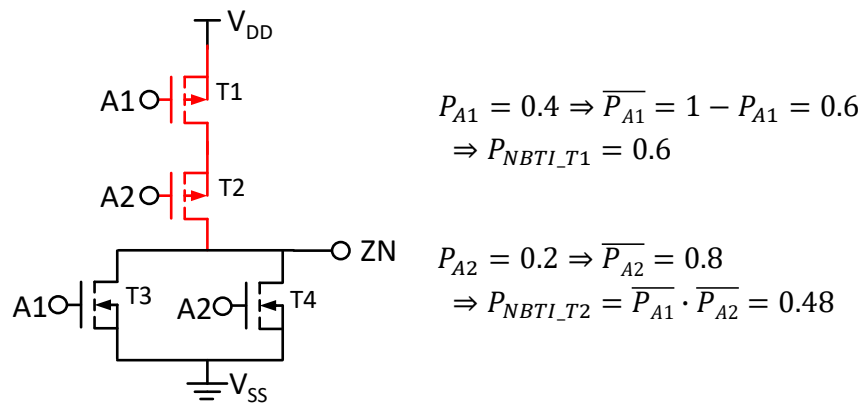
<sup>10</sup>Taschenbuch der Mathematik, Bronstein[6], (16.41b): für Unabhängige Ereignisse A und B gilt:  $P(AB) = P(A)P(B)$ .

<sup>11</sup>De Morganschen Regel [Taschenbuch der Mathematik, Bronstein[6], (5.255)]:  $\overline{a \text{ AND } b} = \bar{a} \text{ OR } \bar{b} \rightarrow a \text{ OR } b = \bar{\bar{a} \text{ AND } \bar{b}}$ .

<sup>12</sup>Inversion= Entstehung des leitenden Kanals beim MOSFET-Transistors; die Dichte der Minoritätsladungsträger erreicht oder übersteigt die Dichte der Majoritätsladungsträger

werden. Hierbei wird ein Transistor seriell in die Versorgung einer Schaltung eingebaut, der die Schaltung von der Versorgungsspannung trennen kann und somit die statische Verlustleistung verringert.

In Abbildung 4.24 wird der Sachverhalt anhand eines NOR-Gatters erläutert. Die Signalwahrscheinlichkeit (Einschaltwahrscheinlichkeit) von Transistor T1 ist 0.4 und 0.2 für T2. Der PMOS-Transistor schaltet bei einem Low-Pegel am Gate-Eingang, d.h. bei der Alterung von PMOS-Transistoren ist die Nullwahrscheinlichkeit von Interesse. Diese ist 0.6 für T1 und 0.8 für T2. Da beide Transistoren im NOR-Gatter in Reihe geschaltet sind, kann der Transistor T1 den Transistor T2 von der Versorgungsspannung  $V_{DD}$  trennen. D.h. für die NBTI-Alterung bedeutet dies, dass bei Transistor T1 der Wert von 0.6 genommen wird und bei T2 die Werte von T1 und T2 miteinander multipliziert werden müssen, wie dies auch bei der Berechnung der Signalwahrscheinlichkeit vom AND-Gatter in Tabelle 4.1 geschehen ist. Diese Berechnung gilt nur, wenn die Signalwahrscheinlichkeiten unabhängig voneinander sind. Sind die Eingänge nicht unabhängig, so kann als Worst-Case-Abschätzung angenommen werden, dass sich die Einschaltvorgänge der Transistoren maximal überlappen. Für das Beispiel bedeutet dies, dass der Transistor T1 zu 60% der Zeit angeschaltet ist und Transistor T2 zu 80% der Zeit zwar ein Low-Pegel hat, aber nur 60% eine Verbindung zu  $V_{DD}$ . Hieraus folgt das T2 für die Alterung auch 60% der Zeit gestresst wird. Der Sachverhalt der NBTI-Alterung von in Reihe geschalteten PMOS-Transistoren wurde auch bereits von [48] verwendet.



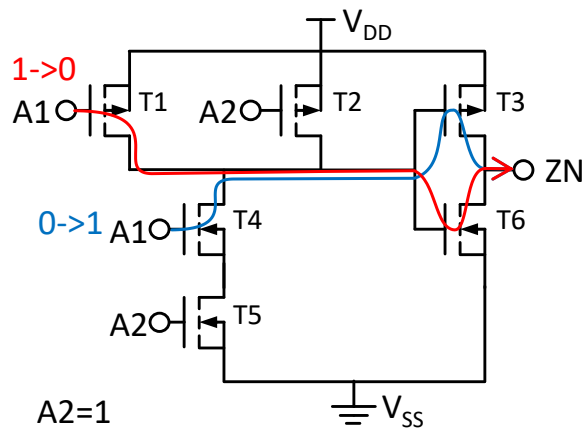
**Abbildung 4.24: Alterung eines NOR-Gatters.** Da die PMOS-Transistoren in Reihe liegen, können die Signalwahrscheinlichkeiten nicht direkt zur Berechnung genutzt werden. Es muss berücksichtigt werden, dass Transistor T1 den Transistor T2 von der Versorgungsspannung  $V_{DD}$  trennt. Ein PMOS-Transistor schaltet bei einem Low-Pegel, daher muss zuerst die 1-Signalwahrscheinlichkeit berechnet werden. Für Transistor T2 bedeutet dies, dass die NBTI-Stress-Wahrscheinlichkeit eine UND-Verknüpfung aus der  $P_{A1}$  und  $P_{A2}$  Wahrscheinlichkeit ist (bei unabhängigen Signalwahrscheinlichkeiten).

Der vorgestellte Sachverhalt kann auch zur Alterungsoptimierung genutzt werden und wird im nächsten Kapitel sowie bei den vorgestellten Optimierungsmöglichkeiten noch einmal genauer erläutert. Im nächsten Abschnitt werden einzelne Transistoren noch einmal genauer betrachtet,

die unter bestimmten Umständen nicht bei der Alterung berücksichtigt werden müssen, da sie keinen Einfluss auf das Gatter-Delay haben.

### Transistor-Reduktion

Neben der Frage, welche Gatter eigentlich auf einem kritischen Pfad liegen und bei der Alterungssimulation betrachtet werden müssen, kann auch die Frage gestellt werden, welche Transistoren eines Gatters überhaupt betrachtet werden müssen? Der Alterungseffekt NBTI betrifft nur PMOS-Transistoren, d.h. die NMOS-Transistoren werden nicht mit simuliert. Bei Betrachtung von z.B. Abbildung 4.25, bei der der kritische Pfad über ein AND-Gatter und von Eingang A1 zu ZN verläuft, stellt sich die Frage, ob die Alterung des Transistors T2 Auswirkungen auf den kritischen Pfad hat.



**Abbildung 4.25:** In der Abbildung ist dargestellt, welche Transistoren bei Schaltvorgängen aktiv sind und somit einen Einfluss auf das Delay haben und bei der Alterung berücksichtigt werden müssen. In diesem Beispiel ist ein AND-Gatter dargestellt, bei dem der Eingang A2 konstant auf 1 liegt und der kritische Pfad über den Eingang A1 geht. In diesem Fall muss der PMOS-Transistor T2 für die NBTI-Alterung nicht berücksichtigt werden. Roter Pfeil: Bei einem Signalwechsel von 1 auf 0 an dem Eingang A1 schalten die Transistoren T1 und T6. Da nur T1 ein PMOS ist, müsste nur dieser Transistor für die Alterung simuliert werden. Bei dem Beispiel mit dem blauen Pfeil ist nur der Transistor T3 von Interesse.

Das Beispiel aus Abbildung 4.25 zeigt, dass die Verbindung A1 zu ZN auf einem kritischen Pfad liegt. Hierbei muss A2 auf eins liegen, damit der Pfad aktiviert ist. Wie man durch die zwei Pfeile erkennen kann, gibt es zwei Schalt-Konstellationen, die bei diesem Beispiel auftreten können: Bei einer steigenden Flanke an Eingang A1 werden die Transistoren T4 (NMOS) und T3 (PMOS) geschaltet. Bei einer fallenden Flanke an A1 schalten die Transistoren T1 (PMOS) und T6 (NMOS). D.h. bei diesem Beispiel sind für die Alterungssimulation von NBTI nur die PMOS-Transistoren T1 und T3 von Interesse. Die NMOS-Transistoren spielen bei NBTI keine Rolle und der PMOS Transistor T2 liegt konstant auf eins. Hier kann erwartet werden, dass die Alterung von Transistor T2 keine Auswirkungen auf das Delay des kritischen Pfades hat, da der Transistor parallel zu dem Transistor T1 angeschlossen ist. Wie würde das bei einer

Reihenschaltung von Transistoren aussehen? Dies soll nun anhand von SPICE-Simulationen überprüft werden.

### Untersuchung der Transistor-Reduktion

Um diese Frage zu klären, wird jeweils das Delay eines NAND- und NOR-Gatters aus Abbildung 4.26 genauer mit Hilfe von Simulationen in SPICE untersucht. Bei dem NAND-Gatter liegen die PMOS-Transistoren parallel zueinander und bei dem NOR-Gatter in Reihe.

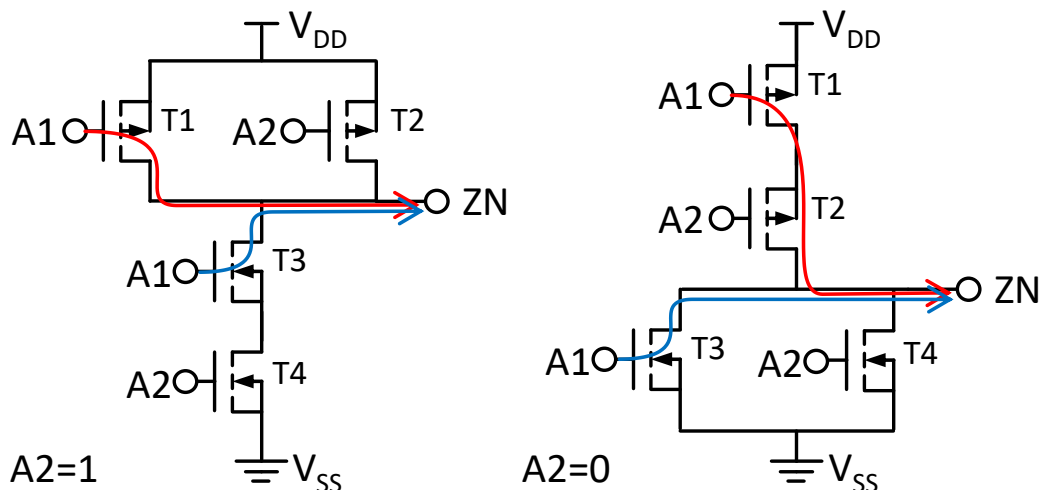


Abbildung 4.26: Linke Seite: Aufbau eines NAND-Gatters mit der Annahme, dass der kritische Pfad über den Eingang A1 verläuft, die PMOS-Transistoren sind parallel geschaltet; rechte Seite: NOR-Gatter mit in Reihe geschalteten PMOS-Transistoren.

Um die Untersuchung durchzuführen, wurden fünf Gatter eines Typs in Reihe zu einem kritischen Pfad über die Eingänge A1 verschaltet, wie dies in Abbildung 4.27 beispielhaft für das NAND-Gatter dargestellt ist. Anschließend wurden die PMOS-Transistoren mit Schwellspannungsschäden versehen und die Delays vom Eingang zum Ausgang des letzten Gatters in SPICE simuliert. Hierbei wurden vier Fälle für die beiden Transistoren T1 und T2 betrachtet: Keiner, jeweils einer der Transistoren und beide Transistoren haben einen Schwellspannungsschaden.

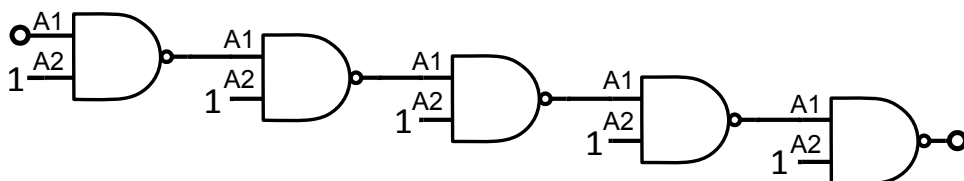


Abbildung 4.27: Mit dieser Schaltung wird untersucht, ob nur die Transistoren eines Gatters, die auf einem kritischen Pfad liegen, Auswirkungen auf das Delay haben. Der kritische Pfad besteht aus fünf NAND-Gattern. Damit er aktiv ist, sind die Eingänge, die nicht auf dem kritischen Pfad liegen, auf Eins geschaltet. Für NOR-Gatter müssen die Eingänge auf Null geschaltet werden.

Die Tabelle 4.3 zeigt die Simulationsergebnisse der beiden Schaltungen. Zusätzlich wurde der Schwellspannungsschaden der Transistoren eingetragen. Wie bereits vermutet wurde, hat bei der Parallelschaltung, der Schwellspannungsschaden des Transistors T2 beim NAND-Gatter keine Auswirkungen auf das Delay. Im Gegensatz hierzu hat bei dem NOR-Gatter (Reihenschaltung) der Transistor T2 eine Auswirkung auf das Delay des Gatters und kann nicht ohne Fehler vernachlässigt werden. Die Abweichung kann bei anderen Szenarien, bei größeren Gattern bzw. bei anderer Technologie noch weit stärker ausfallen als bei diesem Beispiel.

Zusätzlich fällt bei dieser Untersuchung des NOR-Gatters auf, dass bei gleicher Alterung die Position der Transistoren bei der Reihenschaltung Auswirkungen auf das Delay hat. Dies wird in dem Fall sichtbar, wenn der Eingang A1 des NOR-Gatters auf dem kritischen Pfad bei der Alterung von T2 (T1=keine Alterung, T2=Alterung) liegt im Vergleich zu dem Fall, wenn A2 auf dem kritischen Pfad (T1=Alterung, T2=keine Alterung) liegt. Diese Eigenschaft kann für die Alterungsoptimierung ausgenutzt werden. In Kombination mit dem im letzten Unterkapitel bereits angesprochenen Sachverhalt wird die Optimierung als Pin Swapping bezeichnet. Hierbei wird der kritische Pfad auf den Eingang der Gatter gelegt, die ein geringeres Delay aufweisen. Das Vorgehen wird im Unterkapitel zur Optimierung noch einmal aufgegriffen.

Untersuchtes Gatter	Alterungsparameter (alle Gatter)		Ausgangsdelay nach fünf Gattern
NAND Eingang A1	T1=0	T2=0	$d_{rise} = 100\%$ ; $d_{fall} = 100\%$
	T1=A	T2=0	$d_{rise} = 123\%$ ; $d_{fall} = 125\%$
	T1=0	T2=A	$d_{rise} = 100\%$ ; $d_{fall} = 100\%$
	T1=A	T2=A	$d_{rise} = 123\%$ ; $d_{fall} = 125\%$
NOR Eingang A1	T1=0	T2=0	$d_{rise} = 100\%$ ; $d_{fall} = 100\%$
	T1=A	T2=0	$d_{rise} = 121\%$ ; $d_{fall} = 125\%$
	T1=0	T2=A	$d_{rise} = 109\%$ ; $d_{fall} = 114\%$
	T1=A	T2=A	$d_{rise} = 129\%$ ; $d_{fall} = 139\%$
NOR Eingang A2	T1=0	T2=0	$d_{rise} = 100\%$ ; $d_{fall} = 100\%$
	T1=A	T2=0	$d_{rise} = 108\%$ ; $d_{fall} = 111\%$
	T1=0	T2=A	$d_{rise} = 119\%$ ; $d_{fall} = 125\%$
	T1=A	T2=A	$d_{rise} = 127\%$ ; $d_{fall} = 136\%$

**Tabelle 4.3: Delay-Simulation von kritischen Pfaden bestehend aus fünf NAND-Gatter oder fünf NOR-Gattern, wie dies in Abbildung 4.27 abgebildet ist. Die Angabe der Gatter-Eingänge bezieht sich auf die genutzten für den kritischen Pfad. Die Alterungsparameter T1=0 und T2=A, bedeuten das Transistor T1 ungealtert ist und T2 einen Schwellspannungsschaden hat. Die Ergebnisse beziehen sich jeweils auf den ungealterten Zustand (= 100%). Es ist gut zu erkennen, dass beim NAND-Gatter die Alterung des parallel liegenden Transistors keine Auswirkungen auf das Delay hat. Bei dem NOR-Gatter (Eingang A1) gibt es deutliche Unterschiede zwischen dem Fall das (T1=A, T2=0) und dem Fall, wenn beide Transistoren altern (T1=A, T2=A). Die Simulation wurde mit 45 nm Transistoren bei einem typischen Schwellspannungsschaden durchgeführt.**

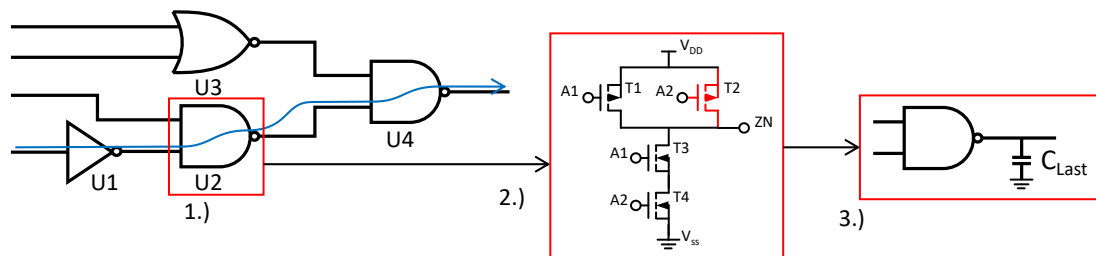
Als Ergebnis der Untersuchung kann festgehalten werden, dass Transistoren, die parallel liegen, nicht mitberücksichtigt werden müssen. Im Gegensatz hierzu haben Transistoren in Reihe eine Auswirkung bei der Alterungssimulation und müssen daher berücksichtigt werden. Im nächsten Abschnitt wird auf die eigentliche Berechnung des Delays eingegangen.

### Separation der Gatter

Um eine Alterungssimulation eines Designs durchzuführen, werden die einzelnen Gatter separiert und einzeln bei der Alterung betrachtet. Hierbei werden die Signalwahrscheinlichkeiten und gegebenenfalls die internen Signalwahrscheinlichkeiten für die einzelnen Gatter berechnet. Anschließend wird für die relevanten Transistoren innerhalb des Gatters die Alterung simuliert. Je nach Gattertype können einzelne Transistoren ausgelassen werden, die keinen Einfluss auf die Alterung haben, wie dies im letzten Abschnitt untersucht wurde. Als Ergebnis wird ein Schwellspannungsschaden für die einzelnen Transistoren der Gatter erhalten. Mit Hilfe einer vorher erstellten Datenbank werden die Delays und Slew Rates für die Gatter interpoliert. Hierzu werden neben dem Schwellspannungsschaden, die Versorgungsspannung, die Temperatur, die Lastkapazität und die Slew Rate (Flankensteilheit) benötigt. In der Gleichung 4.29 ist dieser Zusammenhang zur Berechnung des Delays und der Slew Rate dargestellt.

$$\text{Delay, SlewRate} = \text{Interpolate}(\text{Gattertype}, V_{th}, V_{DD}, T, C, \text{SlewRate}) \quad (4.29)$$

In Abbildung 4.28 ist ein Beispiel dargestellt, in dem ein Gatter aus einem kritischen Pfad betrachtet wird. Da in diesem Beispiel nur einer der Eingänge auf dem kritischen Pfad liegt, wird auch nur ein Transistor gealtert. Im letzten Schritt wird das neue Delay mit den Ergebnissen der Simulation berechnet.



**Abbildung 4.28:** Beispiel einer Alterung und Delay-Berechnung einer Gatterschaltung. Bei der Simulation werden die Gatter einzeln behandelt, losgelöst von der eigentlichen Schaltung (1.). Hierzu werden die notwendigen Transistoren erkannt (2. roter Transistor). Nach der Alterung wird das Delay der einzelnen Gatter mit Hilfe einer Interpolation berechnet. Hierbei ist neben einiger anderen Größen auch die Lastkapazität von großem Einfluss.

Da bereits alle anderen Parameter außer der Lastkapazität betrachtet wurden, wird dies im nächsten Abschnitt vorgenommen.

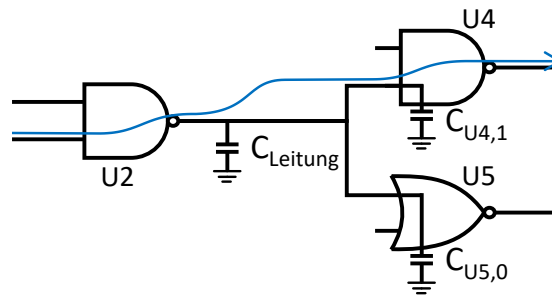
### Lastkapazität

Die Lastkapazität ist eine der Größen, die das Delay beeinflussen. Die Kapazität setzt sich aus der Leitungs- und der Eingangskapazität der angeschlossenen Gatter zusammen. Der Zusammenhang ist in Gleichung 4.30 dargestellt. In Abbildung 4.29 ist ein Beispiel zu der Berechnung dargestellt. Soll für das Gatter U2 das Delay berechnet werden, so muss die Leitungskapazität und die



Eingangskapazitäten der Gatter U4 und U5 berücksichtigt werden. Es reicht bei der Delay-Betrachtung also nicht aus, sich nur auf die Gatter zu beschränken, die auf dem kritischen Pfad liegen, sondern es sind auch Informationen der Gatter nötig, die ebenfalls an die Leitungen des kritischen Pfades angeschlossen sind.

$$C_{Last} = C_{Leitung} + \sum_{n=0}^N C_{input(n)} \quad (4.30)$$



**Abbildung 4.29: Beispiel zur Kapazitätsberechnung.** Die Berechnung des Delays von Gatter U2 benötigt die Eingangskapazitäten von U4 und U5. D.h. selbst wenn nur der kritische Pfad von Interesse ist (durch den blauen Pfeil dargestellt) müssen Informationen zu dem anderen angeschlossenen Gatter vorliegen.

Nachdem in den letzten Unterkapiteln alle Parameter, die für die Delay-Berechnung benötigt werden, besprochen wurden, werden im nächsten Abschnitt die kritischen Pfade betrachtet und es wird analysiert, wie diese reduziert werden können.

### Kritische Pfade

Wie schon an einigen anderen Stellen erwähnt wurde, gibt es bei einem Design im ungealterten Zustand einen kritischen Pfad mit dem längsten Delay, oder sogar mehrere, die das gleiche Delay haben. Durch die Alterung kann es dazu kommen, dass ein anderer Pfad stärker altert als der anfänglich kritische Pfad. Die Gründe hierfür können z.B. sein, dass die Signalwahrscheinlichkeiten auf den einzelnen Pfaden die Alterung so beeinflussen. In Abbildung 4.30 ist ein solcher Fall dargestellt. Der anfänglich kritische Pfad (blauer Pfeil) hat drei Gatter und in diesem Beispiel das längste Delay. Bei den angegebenen Signalwahrscheinlichkeiten in diesem Beispiel kann es passieren, dass der andere Pfad, der durch einen grünen Pfeil dargestellt ist, nach einiger Zeit ein höheres Delay durch die Alterung hat.

Deshalb reicht es nicht aus, für die Alterungssimulation nur den kritischen Pfad einer Schaltung zu betrachten, da er sich durch die Alterung der Transistoren ändern kann. Es macht aber auch keinen Sinn, alle Transistoren bei der Alterungssimulation zu berücksichtigen, da zum einen die Menge der Transistoren bei größeren Schaltungen nicht mehr handhabbar sind und zum anderen ist nicht jeder mögliche Pfad, den es in einer Schaltung gibt, durch Alterung automatisch ein kritischer Pfad.

In dem Kapitel zu den *Grundlagen* wurde bereits ein Verfahren vorgestellt, dass die Anzahl

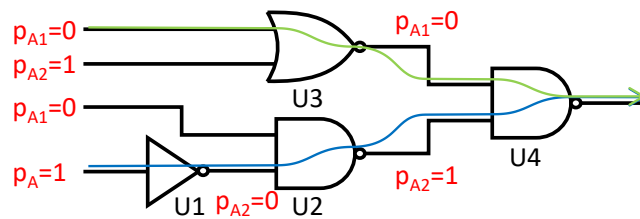


Abbildung 4.30: In der Abbildung ist ein Beispiel dargestellt, bei dem sich der kritische Pfad durch Alterung verändert. In diesem Beispiel passiert dies durch die angegebenen Signalwahrscheinlichkeiten. Am Anfang besteht der kritische Pfad aus den Gattern U1, U2 und U4. Bei diesem Pfad ist nur der A2-Eingang von Gatter U2 auf dem 0 Pegel, wodurch der PMOS-Transistor durchschaltet. Bei dem Pfad, bestehend aus U3 und U4, altern bei beiden Gattern die PMOS-Transistoren maximal, da beide Eingänge einen 0 Pegel haben.

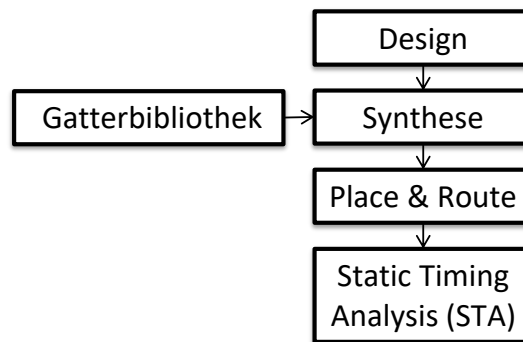
der kritischen Pfade reduziert und damit die Anzahl der bei einer Alterungssimulation zu betrachteten Transistoren verringert. Dieser Ansatz wird auch in dieser Arbeit genutzt. Im nächsten Unterkapitel wird der gesamte Designflow vorgestellt. Hierbei wird unter anderem dann auch das Reduktionsverfahren der kritischen Pfade eingesetzt.

## 4.8 Designflow

### 4.8.1 Voraussetzungen

In diesem Abschnitt werden die bisher vorgestellten Modelle zu einem Gesamtkonzept zusammengesetzt. Wichtig bei diesem Schritt ist die Integration der Modelle in den industriellen Toolflow. Dies hat den Vorteil, dass nicht von Grund auf ein neues Vorgehen entwickelt werden muss. Es können bereits entwickelte Standards, Formate und Tools genutzt werden. Gleichzeitig können die entwickelten Modelle auf einfache Art und Weise mit anderen Industrietools genutzt werden. Ausgehend von dem Industrieflow im Kapitel *Grundlagen* wurde dieser Flow auf die wesentlichen Schritte, die in dieser Arbeit von Interesse sind, vereinfacht. Dargestellt ist dies in Abbildung 4.31. Ausgehend von einer Hardwarebeschreibung auf Registertransferebene wird mit einer gegebenen Gatterbibliothek (.lib) eine Gatter-Netzliste erstellt. Mit Hilfe dieser Netzliste wird das Design platziert und verdrahtet (Place & Route). Anschließend kann mit den erhaltenen Informationen des Delays eine Static Timing Analysis durchgeführt werden.

In den folgenden Unterkapiteln wird zuerst die Aus- und Weitergabe der gealterten Delays erläutert. Anschließend wird der Gesamtflow vorgestellt und zum Abschluss noch ein Konzept zur Ausführung der einzelnen Designschritte präsentiert.



**Abbildung 4.31:** Auf die wesentlichen Punkte, die in dieser Arbeit benötigt werden, reduzierter Industrieflow, verglichen mit dem vorgestellten Flow aus der Literatur in Abbildung 2.16 hat diese Reduktion keine Auswirkungen auf die Anwendbarkeit.

## 4.8.2 Umsetzung

### Aus- und Weitergabe der Ergebnisse

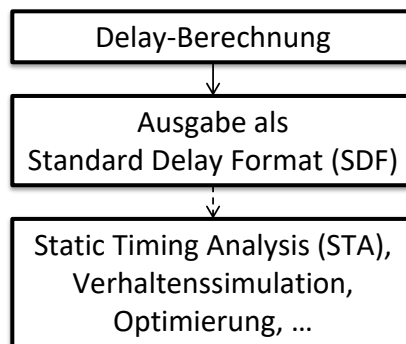
Durch den NBTI-Alterungseffekt erhalten die PMOS-Transistoren einen Schwellspannungsschaden und mit Hilfe des Delay-Modells wird das Delay eines Gatters berechnet. Nachdem die neuen Delays für alle Gatter eines Designs vorhanden sind, müssen diese wieder in den Designflow zurückgeführt werden, damit z.B. mit diesen neuen Werten eine Timing Analyse durchgeführt werden kann. Hier bieten sich mehrere Möglichkeiten an, die Delay-Werte anderen Tools zur Verfügung zu stellen. Ausgangspunkt ist der Industrieflow aus Abbildung 4.31. Eine Möglichkeit wäre z.B., die neuen Delay-Werte in die Gatterbibliothek zu schreiben und die alten Werte dort zu ersetzen. Ein Problem an dieser Stelle ist, dass pro Gattertyp nur ein Eintrag in der Bibliothek vorhanden ist und durch die Alterungssimulation werden für jede Gatterinstanz eines Gattertyps unterschiedliche Delays erzeugt. Neue Einträge für unterschiedlich gealterte Gattertypen können in die Bibliothek eingefügt werden. Der Aufwand und vor allem der Speicherbedarf wären dabei immens.

Als eine sehr gute und auch praktisch anwendbare Lösung hat sich die Rückführung der neuen Delay-Werte in das sogenannte Standard Delay Format (SDF) erwiesen. SDF ist ein 2001 entstandener IEEE Standard[31] zum Speichern und Austauschen von Timing Daten. Er wird von sehr vielen industriellen Tools z.B. zur Timing Analyse oder zur Schaltungssimulation genutzt. Im Anhang B.0.4 ist ein Auszug einer solchen SDF-Datei dargestellt. Es ist möglich, in das SDF für jede Instanz eines Gatters die Delay-Werte für alle Ein-/Ausgangsverbindungen zu schreiben und die SDF-Datei mit externen Programmen z.B. zur Timing Analyse zu nutzen.

Die SDF-Datei muss außerdem nicht aus allen Gattern des Design bestehen, sondern sie kann auch nur aus den Delay-Werten bestehen, die von Interesse sind. Für den Alterungsflow bedeutet dies, dass auch nur die Delays, die auf einem kritischen Pfad liegen, in die SDF-Datei geschrieben werden müssen. Bei dem Einlesen einer SDF-Datei mit einem Timing Tool werden die bereits aus der Gatterbibliothek erzeugten Delay-Werte dann mit denen aus der SDF-Datei in der internen

Datenbank des Tools überschrieben. Anschließend kann eine STA durchgeführt werden, wobei die Gatter, die nicht alterungsrelevant sind, auch keine gealterten Delays haben, sondern nur Gatter, die kritisch werden können. Bei einer Verhaltenssimulation mit gealterten Delay-Werten sollten im Gegensatz zu der Timing Analyse alle Gatter gealtert werden. Diese werden dabei in die SDF-Datei geschrieben, um eine exakte Simulation zu erhalten, in der auch durch Alterung des Delays entstandene Glitches und Hazards<sup>13</sup> berücksichtigt werden.

In Abbildung 4.32 ist der Ablauf zur Erstellung einer SDF dargestellt. Dieses Vorgehen ist Tool-Hersteller unabhängig und könnte z.B. auch ohne Probleme statt für ASICs für FPGAs genutzt werden.



**Abbildung 4.32: Ablauf zur Ausgabe der Delay-Werte als Standard Delay Format (SDF).** Da das SDF-Format standardisiert ist, wird es von allen bewährten Tool-Herstellern als Ein- und Ausgabe Format für Timing Daten angeboten (sowohl bei ASIC als auch bei FPGA Tools). Hierdurch kann das durch den Alterungsflow neu berechnete Delay z.B. durch Tools zur statischen Timing Analyse, Verhaltenssimulation mit gealterten Delay-Werten oder zur Optimierung eingesetzt werden.

### Angepasster Industrieflow/Gesamtflow

In diesem Abschnitt werden die vorher vorgestellten und verwendeten Informationen zur Alterungssimulation genutzt. Diese Arbeit beschränkt sich auf die Betrachtung von NBTI. Es ist mit den entwickelten Tools und dem Entwurfsfluss soweit vorbereitet, dass weitere Alterungseffekte leicht einzubauen sind, wie dies bereits an anderer Stelle aufgeführt wurde. Hierzu muss nur das Alterungsmodell auf andere Effekte angepasst werden.

Der zusammenhängende Ablauf, der in diesem Kapitel einzeln vorgestellter Modelle, ist in Abbildung 4.33 dargestellt. Zur Übersichtlichkeit wurden Zwischenschritte in der Abbildung weggelassen. Diese Zwischenschritte wurden in den vorherigen Unterkapiteln näher erläutert. Ausgehend von einem Design, das auf Gatter- oder Registertransferebene vorhanden ist, wird eine Synthese durchgeführt und eine Netzliste des Designs erstellt. Die Netzliste wird anschließend platziert und verdrahtet (P&R). Aus den gewonnenen Daten und einer Verhaltenssimulation der Netzliste können dann die IR-Drop- und die Temperatursimulation durchgeführt werden.

<sup>13</sup>Glitch=ungewollter Schaltvorgang, der durch Delay Differenzen in einer Schaltung entstehen kann; Hazard=eine Schaltung, in der potenziell Glitches möglich sind.

Bei der Temperatursimulation muss zusätzlich noch ein Package mit Hilfe des Package-Editors angelegt werden. Diese Simulation der elektrothermischen Kopplung muss für jedes short-term Missionsszenario durchgeführt werden, das später für die Alterungssimulation genutzt werden soll. Angedeutet ist dies durch den gestrichelten Pfeil von Testbenches nach dem Missionsszenario-Editor in der Abbildung. Zusätzlich kann die Schaltung auf die kritischen Pfade reduziert werden, um die Anzahl der zu betrachtenden Transistoren zu verringern. Es gibt aber auch die Möglichkeit, eine Simulation aller Gatter des Designs durchzuführen. Anschließend wird mit dem Missionsszenario-Editor das long-term Szenario aus den einzelnen short-term Szenarien zusammengesetzt. Hierbei muss unter anderem entschieden werden, wie lange das System betrieben wird (Länge der Alterung) und wie sich die Umgebungstemperatur über die Zeit verändert. Als nächster Schritt wird die Alterungssimulation mit dem gewählten long-term Missionsszenario durchgeführt, in diesem Fall das NBTI-Modell, wie es im entsprechenden Kapitel beschrieben wurde. Der Schwellspannungsschaden wird anschließend mit dem Delay-Modell genutzt, um die Werte der einzelnen Gatter zu bestimmen. Das Ergebnis des Flows sind die gealterten Delays des Designs in einer SDF-Datei. Diese Datei kann, da sie standardisiert ist, mit einem beliebigen Industrietool weiterverarbeitet werden.

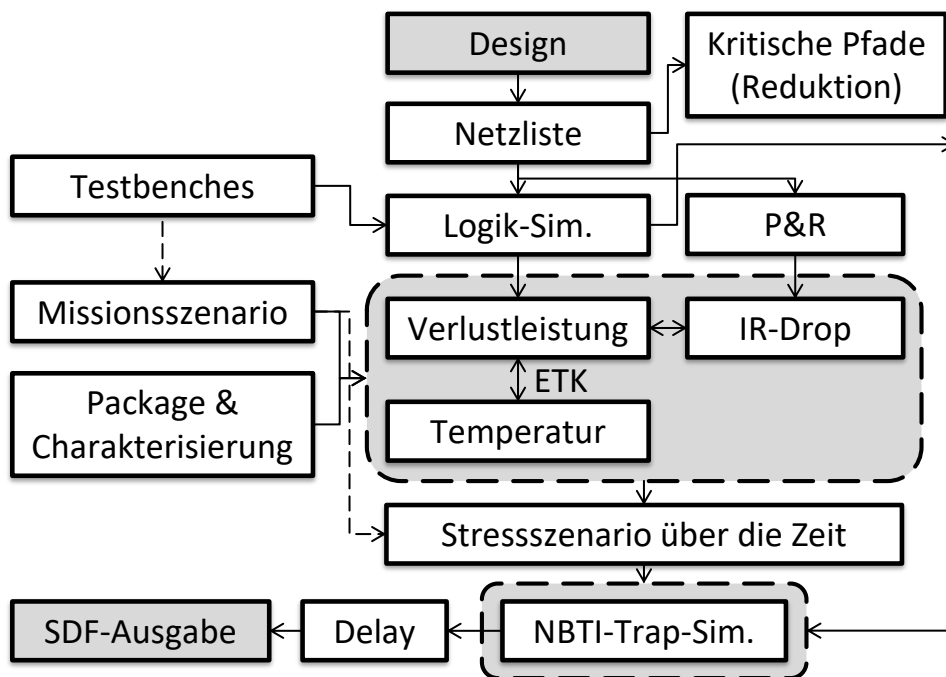


Abbildung 4.33: Alterungsflow mit den einzelnen Modellen, die in diesem Kapitel vorgestellt wurden. Ausgehend von den typischen Entwicklungsschritten, Synthese und Place & Route wird aus einem Missionsszenario ein Stressszenario abgeleitet. Hierbei wird die elektrothermische Kopplung (ETK) durch das Verlustleistungsmodell, das IR-Drop-Modell und die thermische Simulation durchgeführt.

### Konzept zur Toolausführung

Zu dem Gesamtkonzept dieser Arbeit gehört es, unterschiedliche Modelle zu entwickeln und diese in einem industriellen Toolflow zu integrieren. Hierbei werden in den Anwendungen die Programme in bestimmter Reihenfolge ausgeführt. Der Toolflow nutzt unterschiedliche Programme/Tools, wobei auch Bedingungen und Abhängigkeiten überprüft werden müssen. Hierfür wurde ein Konzept zur Ausführung und Überwachung der unterschiedlichen aufzuführenden Programme entwickelt. Dieses entwickelte Konzept basiert darauf, dass ein aufzuführender Programmflow als Petri-Netz dargestellt wird. Ein Petri-Netz ist ein endlicher Automat, der aus Transitionen und Stellen besteht. Die Transitionen und Stellen sind über gerichtete Kanten verbunden, wobei nur Transitionen mit Stellen und Stellen mit Transitionen verbunden werden können. Zusätzlich kann in jeder Stelle eine Marke gesetzt werden. Hat eine Transition an jeder Eingangsstelle eine Marke und sind alle Ausgangsstellen frei, so kann diese Transition schalten. Hierbei werden alle Marken an den Eingangsstellen entfernt und an jeder Ausgangsstelle wird eine Marke gesetzt. Hervorzuheben ist, dass Petri-Netze mit dieser Schaltbedingung der Transitionen Nebenläufigkeit abbilden. Transitionen können unabhängig voneinander schalten, sobald die Schaltbedingung erfüllt ist.<sup>14</sup>

Übertragen auf die Ausführung von vielen unterschiedlichen Programmen bedeutet dies, dass ein Programm als eine Transition und die Informationen (z.B. eine Datei), die das Programm benötigt, als Stelle aufgefasst werden kann. Beim Ausführen eines Programms müssen alle Eingabe-Informationen vorhanden sein (d.h. Marken gesetzt), damit das Programm ausgeführt werden kann. Auch ist es gewollt, dass ein nebenläufiges Ausführen der Programme durchgeführt wird. In Abbildung 4.34 a) ist diese Interpretation der Ausführung eines Programms als ein Petri-Netz dargestellt. In Abbildung 4.34 b) wird dies anhand eines Beispiels noch einmal veranschaulicht. Im Kapitel *Implementierung* wird näher auf das Petri-Netz-Konzept und die Umsetzung eingegangen.

## 4.9 Zuverlässigkeitsbewertung

### 4.9.1 Voraussetzungen

Das Ergebnis des Alterungsflows, der im letzten Unterkapitel vorgestellt wurde, ist das Delay einer gealterten Schaltung, das in einer SDF-Datei gespeichert wird. Mit Hilfe der Datei kann eine Static Timing Analysis (STA) durchgeführt werden. Diese STA liefert als Ergebnis für das gealterte Szenario das kritische Timing für diese Komponente.

### 4.9.2 Umsetzung

Um eine Bewertung der Zuverlässigkeit der Komponente durchzuführen, wird das ungealterte Timing der Komponente als Ausgangspunkt genommen. Es dient auch zur Überprüfung, ob das

<sup>14</sup>Weitere Eigenschaften von Petri-Netzen können in folgender Literatur nachgelesen werden [62][63]

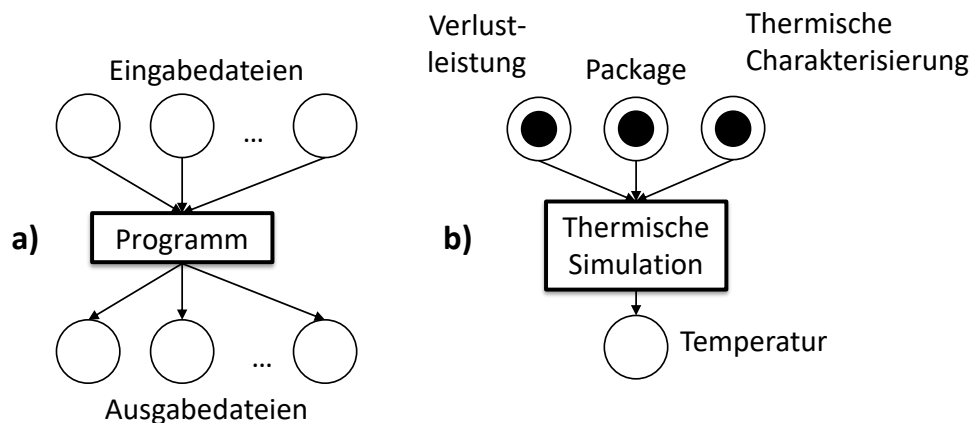


Abbildung 4.34: In Abbildung a) ist das Petri-Netz für ein beliebiges auszuführendes Programm dargestellt. Das Programm benötigt zur Ausführung eine bestimmte Anzahl an Eingabedateien und erzeugt eine beliebige Anzahl an Ausgabedateien. Überführt in ein Petri-Netz bedeutet dies, dass die Eingabe- und Ausgabedateien Stellen sind. Die Marken geben an, dass die Dateien vorhanden sind und das auszuführende Programm ist die Transition. Eine Transition kann nur ausgeführt werden, wenn alle Eingangsstellen mit Marken besetzt sind und alle Ausgabestellen leer sind. Abbildung b) zeigt ein Beispiel, bei dem die Temperatursimulation dargestellt ist. Die benötigten Eingangsdaten in Form von drei Dateien sind als drei Stellen (Verlustleistung, Package und Thermische Charakterisierung) dargestellt. Es gibt eine Stelle als Ausgang der Transition „Thermische Simulation“, dies ist die berechnete Temperatur. Damit eine Temperatursimulation durchgeführt werden kann, müssen alle drei Eingangsdateien vorhanden sein. Dies ist in dem Beispiel durch die drei Marken an den Eingangsstellen zu sehen.

Design der Spezifikation entspricht. Die Spezifikation bedeutet in diesem Fall, dass das Design alle Timing Vorgaben erfüllt und es mit einer vorgegebenen Taktfrequenz betrieben werden kann. Mit dem Alterungsflow können nun unterschiedliche Missionsszenarien getestet werden. Für unterschiedliche Szenarien, d.h. für unterschiedlich lange Laufzeiten oder andere Temperaturen werden unterschiedliche Alterungen durchgeführt. Das Ergebnis der einzelnen Alterungssimulationen sind unterschiedliche SDF-Dateien, die mit Hilfe einer STA das Timing der Komponente liefern. Für jedes der Szenarien wird das spezifizierte Timing überprüft. Das Gesamtergebnis ist eine Sammlung von unterschiedlichen Szenarien, bei denen die Komponente funktionstüchtig ist oder Timing-Probleme aufgetreten sind. In Abbildung 4.35 ist dieses Vorgehen dargestellt. Es wurden für unterschiedliche Zeiträume Alterungen simuliert. Aus den SDF-Dateien wird das Timing bestimmt und einige, in Rot dargestellte Komponenten und Instanzen, halten nach den Alterungen das spezifizierte Timing nicht mehr ein. Zusätzlich können, wie dies in der Abbildung dargestellt ist, auch verschiedene „Process Corners“ der Herstellungstechnologie der Komponente abgedeckt werden, um die Verteilung der Lebenserwartung abschätzen zu können. Im nächsten Abschnitt wird auf die Optimierungsmöglichkeiten eingegangen.

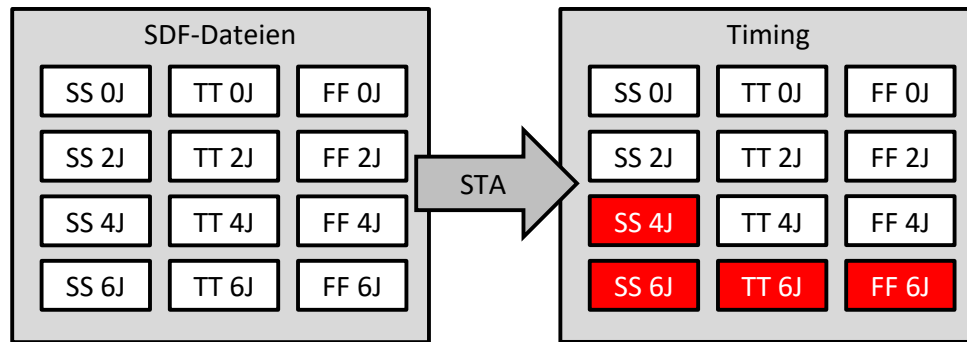


Abbildung 4.35: Systembewertung einer gealterten Komponente: Mit dem vorgestellten Alterungsflow werden für verschiedene Szenarien Alterungen simuliert. Mit Hilfe von einer STA wird aus den erhaltenen SDF-Dateien das Timing der Komponente berechnet. Anschließend wird überprüft, ob das Timing noch innerhalb des spezifizierten Timings liegt. In diesem Beispiel sind drei verschiedene „Process Corners“ der Transistoren dargestellt (slow/slow, typical/typical, fast/fast für NMOS/PMOS).

## 4.10 Zuverlässigkeitsoptimierung

### 4.10.1 Voraussetzungen

Anschließend können mit den Erkenntnissen zu den Timings der kritischen Pfade, Optimierungen des Designs durchgeführt werden. Die Optimierungsmöglichkeiten sind im nächsten Unterkapitel vorgestellt, wie sie in den Designflow integriert werden können.

### 4.10.2 Umsetzung

#### Systemoptimierung

In diesem Unterkapitel werden einige Möglichkeiten vorgestellt, die genutzt werden können, um das Design gegenüber Alterungsproblemen zu härten, so dass das Design auch in einem gealterten Zustand noch funktioniert. Zusätzlich wird in diesem Unterkapitel jeweils aufgezeigt, wie diese Optimierungsmöglichkeiten mit dem vorgestellten Designflow umgesetzt werden können. Die hier vorgestellten Ideen werden im Kapitel *Evaluation* einzeln getestet und dabei wird bewertet, wie gut sie letztendlich für die Alterungsoptimierung geeignet sind. Generell muss an dieser Stelle noch einmal erwähnt werden, dass diese Optimierungen nur für NBTI betrachtet werden und für andere Alterungseffekte auch andere Optimierungen sinnvoll sein können.

Wie im Kapitel *Grundlagen* schon herausgearbeitet wurde, sind für NBTI folgende Einflussgrößen besonders wichtig: Herstellungstechnologie, Temperatur und Versorgungsspannung des Systems und die Aktivität der PMOS-Transistoren. Die entwickelten Modelle und die Integration dieser in den Industrieflow ermöglichen es Maßnahmen abzuleiten, um detaillierte Informationen über die Ursachen des Alterungsproblems des untersuchten Designs zu erfahren. Dies bedeutet, dass einzelne Gatter identifiziert werden können, die besonders durch die Alterung betroffen sind. Hierbei können detaillierte Szenarien betrachtet werden, die in diesem Umfang noch mit



keinem anderen Ansatz umgesetzt werden konnten. Im Kapitel *Evaluation* werden die einzelnen Optimierungen exemplarisch für einen 32bit-Addierer angewendet.

### **Herstellungstechnologie**

Die erste Konsequenz aus der Technologieabhängigkeit von NBTI ist eine andere Technologie auszuwählen, die nicht so sensitiv auf den Alterungseffekt reagiert. Wie bereits im Kapitel *Grundlagen* aufgezeigt wurde, wird das Problem mit NBTI bei kleinen Herstellungsgrößen stärker. D.h. besonders Designs, die höheren Temperaturen und Spannungen ausgesetzt sind, wie z.B. Leistungselektronik, sollten weiterhin kleinere Herstellungstechnologien meiden. Normalerweise werden Leistungshalbleiter ohnehin in wesentlich größeren Herstellungsgrößen produziert. Es gibt aber zunehmend hochintegrierte Hybridhalbleiter, die aus Leistungselektronik- und Digitaltechnikteil bestehen.

### **Dynamic voltage and frequency scaling**

Unter „Dynamic Voltage and Frequency Scaling“ (DVFS) versteht man das dynamische Anpassen der Versorgungsspannung und der Frequenz eines Designs oder einer Komponente. Hierbei wird durch das Verringern der Taktfrequenz die Möglichkeit gegeben, die Spannung zu senken, da die kritischen Pfade, die die Taktfrequenz bestimmen, nun geringer sind. Dies ist nur möglich, wenn das System zu diesem Zeitpunkt eine geringere Rechenleistung anbieten muss. Ein Nebeneffekt von der Absenkung der Spannung und der Taktfrequenz ist, dass die dynamische Verlustleistung erheblich verringert wird, vergleiche hierzu die Formel der dynamischen Verlustleistung im Kapitel *Grundlagen*. Aus der Verringerung der Verlustleistung ergibt sich wiederum eine niedrigere Temperatur. D.h. durch DVFS werden zwei wichtige Einflussgrößen (Versorgungsspannung und Temperatur) für NBTI verringert, die einen erheblichen Einfluss auf die Alterung haben. Zum Test des Einflusses von DVFS auf NBTI gibt es ein Szenario im Kapitel *Evaluation*.

### **Power Gating**

Power Gating wird die Methode genannt, um einen Teil oder das gesamte Design auszuschalten und von der Versorgungsspannung vorübergehend, wenn die Schaltung nicht benötigt wird, zu trennen. Dies geschieht durch zusätzliches Hinzufügen von Transistoren zum Abschalten der Komponente. Der Einfluss von Power Gating auf die NBTI liegt dabei auf der Hand: Durch das Abschalten können sich die Transistoren regenerieren und der Schwellspannungsschaden nimmt hierbei ab. Zusätzlich nimmt die Gesamtverlustleistung des Systems ab und verringert so die Temperatur. In dieser Arbeit wird das Hinzufügen zusätzlicher Steuerlogik und das Hinzufügen der Power-Gating-Transistoren nicht berücksichtigt. Diese Arbeit beschränkt sich auf Auswirkungen auf die vorhandene Schaltung. Auch die durch Power Gating hinzugefügten Transistoren unterliegen der Alterung und können so zu neuen Alterungsproblemen führen.

### **Pipelining und Parallelisierung**

Zwei wichtige Entwurfstechniken bei der Entwicklung von digitalen Systemen, um die Rechenleistung zu erhöhen, sind Pipelining bzw. die Parallelisierung von Komponenten. Beim Pipelining werden zusätzlich Register in die Komponente eingebaut, um so den kritischen Pfad zu verkürzen. Man erreicht hierdurch eine Datenlatenz der Komponente, d.h. es dauert mehrere Takte, bis das Ergebnis zu den Eingaben am Ausgang erscheint. Durch die kürzeren kritischen Pfade kann die Komponente höher getaktet werden, wodurch sie einen höheren Datendurchsatz als die nicht gepipelinten Komponente erreicht.

Bei der Parallelisierung wird die Komponente mehrfach ausgeführt und kann gleichzeitig mit einer niedrigeren Frequenz und Spannung betrieben werden. Der Nachteil ist, dass weiterer Platz für die zusätzlichen Komponenten benötigt werden.

### **Pin Swapping**

Pin Swapping oder Pin Reordering[85] nutzt den Sachverhalt aus, dass die Eingänge eines Gatters unterschiedlich stark auf die Alterung reagieren können (Reihenschaltung von Transistoren), d.h. das gealterte Delay eines Gatters hängt auch von dem genutzten Eingang eines Gatters ab und nicht nur von den Alterungsparameter Versorgungsspannung, Temperatur und Signalwahrscheinlichkeit. Diese Eigenschaft wurde bereits im Unterkapitel Gatter-Modell und Transistor-Reduktion angesprochen. Zur Optimierung kann das schlechteste Delay bzw. der Eingang, der auf einem alterungsrelevanten kritischen Pfad liegt, mit einem besseren Eingang getauscht werden. Der Aufwand der Optimierung ist an dem Designflow recht gering, das Stressszenario und die Alterungssimulation werden nicht geändert, sondern es werden nur die Signalwahrscheinlichkeiten an den Eingängen vertauscht, dieser Eingriff betrifft also die Delay-Berechnung.

### **Temperatur- und Floorplan-Optimierung**

Die Temperatur ist eine entscheidende Größe für die meisten Alterungseffekte, wie dies auch für NBTI am Anfang dieses Kapitels gezeigt wurde. Es macht also durchaus Sinn, aus den Ergebnissen, die sich durch die Alterungssimulation ergeben, sich auch die Temperaturwerte der kritischen Gatter anzusehen und so gegebenenfalls eine Floorplan-Optimierung durchzuführen. Weitere Möglichkeiten zur Senkung der maximalen Temperatur sind zum einen das genutzte Package und zum anderen die genutzte Kühlung des Systems. Hier ist es z.B. sinnvoll von einer passiven auf eine aktive Kühlung zu wechseln. Durch ein Ändern des Floorplans oder der Temperatureigenschaften muss das gesamte Stressszenario sowie die Alterungssimulation mit Delay-Berechnung erneut durchgeführt werden.

## 4.11 Zusammenfassung

In diesem Kapitel werden die einzelnen, in dieser Arbeit entwickelten und genutzten Modelle zur Entwicklung eines Designflows zur Alterungssimulation, vorgestellt. Hierbei werden zuerst die einzelnen Teilmodelle vorgestellt, die man zur Simulation benötigt.

Ausgangspunkt ist ein Trap basiertes NBTI-Modell, bei dem man für einen Transistor die einzelnen Traps als eine RC-Schaltung simulieren kann. Zusätzlich werden die zu simulierenden Traps in einen long-term und short-term Teil unterteilt. Die long-term Traps können unter bestimmten Voraussetzungen mit Hilfe der Signalwahrscheinlichkeit, die an den Eingängen des alternden Transistors anliegen, simuliert werden. Bei dem short-term Traps reicht es in den meisten Fällen aus, den Worst-Case-Fall, d.h. dass alle Traps aktiviert sind, anzunehmen. Für das NBTI-Modell werden also die Signalwahrscheinlichkeit, die Temperatur und die Versorgungsspannung benötigt, diese Informationen, die zeitabhängig sind, nennt man Stressszenario. Dieses wiederum ist also der Ausgangspunkt für die Alterungssimulation. Es wird genutzt, um mit Hilfe des vorgestellten NBTI-Modells den Schwellspannungsschaden für jeden relevanten Transistor zu berechnen, woraus dann letztendlich das gealterte Delay für die Gatter errechnet wird.

Um ein Stressszenario zu erhalten, muss zuerst ein Missionsszenario definiert werden, das beschreibt, wie das System über die Laufzeit genutzt wird. Das Missionsszenario ist dann der Ausgang für die Simulation eines Stressszenarios mit Hilfe von Simulationsmodellen für die thermische Simulation, IR-Drop-Simulation, den Verlustleistungs-Modellen und der elektrothermischen Kopplung.

Das vorgestellte thermische Modell basiert auf der Methode der Green-Funktion und beschleunigt durch eine Charakterisierung des thermischen Systems die Berechnung. Zusätzlich wird für die thermische Simulation eine Beschreibung des zu simulierenden Systems benötigt. Hierzu wird ein Package-Modell vorgestellt.

Neben der Temperatur spielt auch die Versorgungsspannung bei NBTI eine Rolle, aus diesem Grund wird ein IR-Drop-Modell vorgestellt, dass es ermöglicht, Spannungsschwankungen der Versorgungsleitungen zu simulieren.

Zusätzlich wird ein Ablauf entwickelt, der die elektrothermische Kopplung abbildet. Hierbei wird die Wechselwirkung zwischen der Temperatur und der Verlustleistung berücksichtigt.

Nachdem die Transistoren gealtert sind, muss das neue Delay der Gatter berechnet werden. Hierzu wird ein Delay-Modell vorgestellt. Dabei wird besonders darauf geachtet, dass bei der Simulation der Alterung nur die Transistoren betrachtet werden, die auch relevant sind. Dabei kann die Anzahl der Gatter und Transistoren, z.B. durch die Kritische-Pfad-Reduktion, verringert werden.

Ein weiterer wichtiger Punkt in diesem Kapitel ist der Aufbau des Gesamtflows, also der Ablauf und das Verbinden der einzelnen Modelle. Ein zentraler Aspekt ist hierbei auch die Integration in einen typischen in der Industrie anwendbaren Designflow.

Es wird auf die Zuverlässigkeitsbewertung eingegangen, um aufzuzeigen, wie die Ergebnisse weiterverarbeitet werden und welche Erkenntnisse aus den Ergebnissen gewonnen werden können.

Zum Abschluss des Kapitels wird vorgestellt, wie die Ergebnisse genutzt werden können, um das Design zu optimieren.

Im folgenden Kapitel *Implementierung* werden die hier vorgestellten Modelle umgesetzt. Dabei liegt ein Schwerpunkt darauf, die Modelle auf eine parallele Ausführung zu optimieren. Dies ist besonders von Interesse, da heutige Hardwarearchitekturen aus parallelen Recheneinheiten bestehen.

---

## Implementierung

---

In diesem Kapitel wird erläutert, wie die im vorherigen Kapitel vorgestellten Konzepte und Modelle umgesetzt werden. Hierbei wird zum einen betrachtet, wie die benötigten Informationen für die einzelnen Modelle erhalten werden können und zum anderen wird auf ihre Implementierung eingegangen. Wie bereits bei den einzelnen Modellen beschrieben, wurde bei den Konzepten darauf geachtet, dass die Modelle sich besonders für die Parallelisierung eignen und so eine ausreichende Performance für den gesamten Alterungsflow bieten. Im folgenden Abschnitt wird die Programmiersprache und das Modell hinter OpenCL vorgestellt, das zum Verständnis der nächsten Unterkapitel benötigt wird. Anschließend werden die Implementierung der einzelnen Modelle aus dem letzten Kapitel erläutert. Hierbei wird dieselbe Reihenfolge bei der Implementierung der Modelle gewählt wie im Kapitel *Konzepte und Modelle*.

### 5.1 OpenCL

Die Umwandlung der Verlustleistung in Abwärme hat sich seit dem Jahr 2005 bei den CPUs als eine physikalische Grenze herausgestellt. Beschränkt durch die Kosten ergab sich hierbei eine Grenze der Taktfrequenz eines Einzelkernprozessors bei etwa 4 GHz. Die Folge dieser Grenze war der Wechsel von einzelnen CPU-Kernen zu Mehrkernprozessoren.

In den letzten Jahren hat sich auch neben den CPUs immer mehr die Nutzung von Grafikkarten (GPUs - Graphics Processing Unit) zur Berechnung von parallelen Aufgaben durchgesetzt. Im Gegensatz zu CPUs, die mittlerweile auch immer mehr parallele Berechnungseinheiten aufweisen, sind GPUs von vornherein für die parallele Berechnung entwickelt worden. Ursprünglich wurden diese GPUs eigentlich für Computerspiele entwickelt. Beide große Grafikkartenhersteller, AMD und NVIDIA, haben mittlerweile ihre Produkte auch auf den Supercomputing Markt ausgerichtet und bieten hierfür auch speziell angepasste Lösungen an. Besonders bei der wissenschaftlichen

Berechnung kommt es nicht nur auf die Berechnung von Daten in einfacher Genauigkeit (single precision - SP) an, sondern viele Probleme benötigen auch eine doppelte Genauigkeit (double precision - DP). Heutige GPUs haben einige Tausend Berechnungseinheiten auf einer Grafikkarte integriert und erreichen Rechenleistungen von mehr als zehn TeraFlops bei einfacher Genauigkeit. Die Rechenleistung für die doppelte Genauigkeit ist sehr stark von der GPU-Architektur abhängig und kann zum einen durch eine eigene DP-Hardware implementiert sein, zum anderen durch eine Zusammenschaltung der SP-Einheiten.

Jeder der beiden Hersteller bietet eigene Programmiersprachen und Entwicklungstools zur Programmierung ihrer GPUs an, diese sind bei AMD-Stream und bei NVIDIA-Cuda. Leider sind beide Sprachen nicht direkt kompatibel zueinander, d.h. man legt durch die Wahl der Hardware auch die Programmiersprache und Entwicklungstools fest. Bei der Wahl der Programmiersprache wurde in dieser Arbeit eine Alternative zu Steam und Cuda gewählt, die OpenCL heißt. OpenCL (Open Computing Language) ist eine im Jahre 2008 von der Khronos Group veröffentlichte herstellerunabhängige Sprache, die für die parallele Berechnung auf unterschiedlicher Hardware geeignet ist, d.h. die entwickelten Programme können sowohl auf CPUs als auch auf GPUs genutzt werden. Die einzige Voraussetzung ist ein OpenCL-Treiber für die Hardware<sup>1</sup>.

### 5.1.1 Compute Device

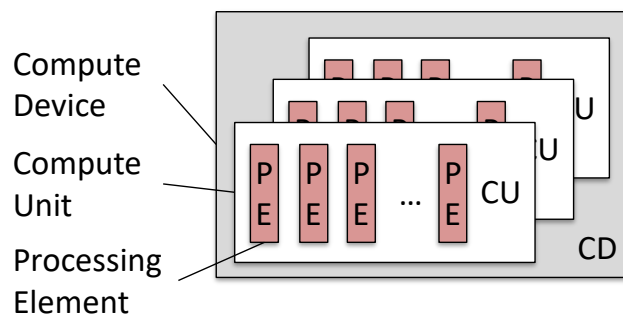
Da OpenCL herstellerunabhängig ist, ist es besonders wichtig, dass es ein einheitliches Modell für die Hardware als auch für die Programmierung dieser gibt. Dieses Modell wurde als OpenCL-Standard veröffentlicht und wird auch aktiv von vielen Herstellern gemeinsam weiterentwickelt. Die aktuelle Version ist 2.0. Je nach Hardware werden nicht immer die aktuellsten Versionen unterstützt. Das zentrale Element in OpenCL ist das Compute Device. Es beschreibt wie die Hardware auszusehen hat, die den OpenCL-Code ausführen soll. In Abbildung 5.1 ist der Aufbau des Compute Device dargestellt. Es besteht aus mehreren Compute Units, die wiederum in einzelne Processing Elements unterteilt sind.

Die Processing Units sind letztendlich die Hardwareeinheiten, die die Berechnung durchführen. Ein Beispiel für ein Compute Device (CD) ist ein Quad-Core-CPU, wobei die vier Kerne der CPU die Compute Units (CU) darstellen. Innerhalb der einzelnen CPU-Kerne gibt es dann mindestens ein Processing Element (PE), diese können je nach Hardware auch mehrere Einheiten sein.

Bei GPUs ist die Anzahl der vorhandenen PEs deutlich höher, da historisch betrachtet gerade die Bild-Verarbeitung hochgradig parallel bearbeitet werden kann. Im Anhang C ist der Aufbau einer CPU im Vergleich zu einer GPU genauer dargestellt.

---

<sup>1</sup>Für die schriftliche Ausarbeitung des Kapitels wurden unterschiedliche Quellen verwendet, siehe hierzu Literaturverzeichnis [37][55][65]



**Abbildung 5.1: Hierarchischer Aufbau eines OpenCL Compute Devices.** Ein Compute Device besteht aus mehreren Compute Units, die wiederum aus mehreren Processing Elements bestehen. Diese sind die Recheneinheiten, die letztendlich den Code ausführen. Die Anzahlen der einzelnen Einheiten unterscheiden sich abhängig vom Hersteller und des Produktes erheblich. Die Anzahl der PEs kann von einigen wenigen bis zu einigen Tausenden gehen; übernommen aus Quelle: [55].

### 5.1.2 Speicherhierarchie

Zum Aufbau der Berechnungseinheiten gehört auch der Aufbau der Speicherhierarchie. In Abbildung 5.2 ist der Aufbau der einzelnen Speicher eines OpenCL CUs zu sehen. Der Speicher ist in einen globalen Speicher für allgemeine Daten und in einen konstanten Speicher für nur lesbare Daten unterteilt. Die zu verarbeiteten Daten werden von der Hostanwendung, die die Berechnung auf dem CD startet, in diesen globalen/konstanten Speicher kopiert. Zusätzlich wird dieser globale, konstante Speicher auch gecacht, da er im Vergleich zu dem übrigen Speicher langsamer ist. Neben dem bisher vorgestellten Speicher gibt es noch einen lokalen und einen privaten Speicher. Der lokale Speicher kann von den jeweiligen PE einer CU gelesen und geschrieben werden, wohingegen andere CU nur auf ihren eigenen lokalen Speicher zugreifen können. Der private Speicher ist nur von den einzelnen PEs nutzbar. Wichtig ist an dieser Stelle, dass die Nutzung des unterschiedlichen Speichers aktiv vom Anwendungsentwickler im Programmcode gesteuert werden muss.

Von der Performance ist der globale Speicher am langsamsten und der private am schnellsten, wohingegen der globale Speicher einige GByte groß ist und der private nur einige KByte hat. Die Synchronisierung von Daten ist ein wichtiger Aspekt. Wie bereits beschrieben wurde, können PEs innerhalb einer CU sich über einen lokalen Speicher austauschen. Ein Datenaustausch zwischen einzelnen CU hingegen ist nur über den globalen Speicher möglich. Eine noch schlechtere Anbindung ist die Synchronisierung von mehreren CDs, da dies über die Hostanwendung durchgeführt werden müsste.

### 5.1.3 Ausführung

In diesem Unterkapitel wird vorgestellt, wie die entwickelten OpenCL-Anwendungen ausgeführt werden. Ein geschriebenes OpenCL-Programm wird in OpenCL als Kernel bezeichnet und auf den Compute Devices ausgeführt. Hierzu wird zunächst eine Hostanwendung benötigt, die die

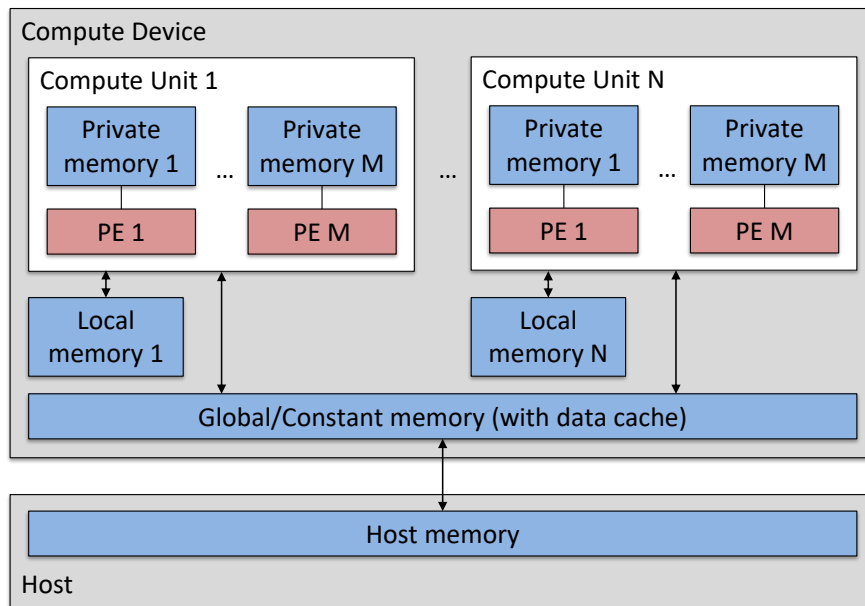


Abbildung 5.2: Aufbau der OpenCL-Speicherhierarchie. Von der Hostanwendung werden die Daten aus dem Hostspeicher in den globalen Speicher des Compute Device kopiert. Die einzelnen Compute Units können auf den globalen Speicher zugreifen. Sie können auch auf einen lokalen Speicher innerhalb einer Compute Unit zugreifen oder die einzelnen Processing Elemente können den privaten Speicher nutzen. Die Compute Units können nicht auf den lokalen Speicher der anderen CUs zugreifen, genauso wenig wie PEs nicht auf den privaten Speicher einer anderen PE zugreifen; übernommen aus Quelle: [55].

Verwaltung, Konfiguration und Kompilierung für die OpenCL Compute Devices erledigt. In Abbildung 5.3 ist der Ablauf zum Ausführen von OpenCL-Kernels dargestellt.

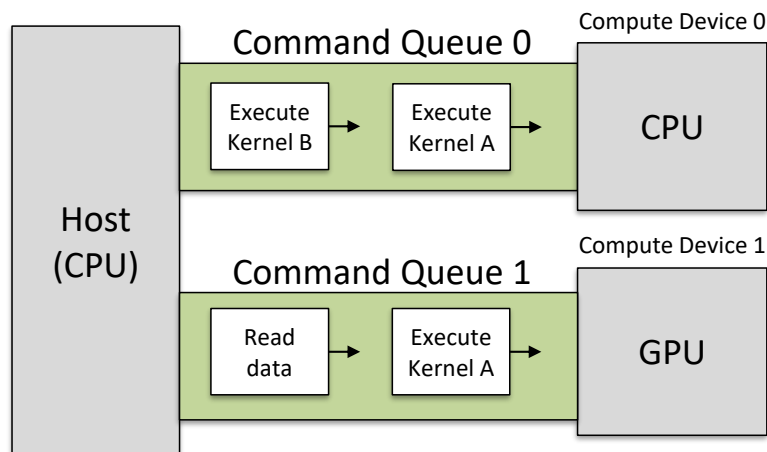


Abbildung 5.3: Beispiel des OpenCL Host Interfaces. Dieses Beispiel besteht aus einem Host, der zwei Compute Devices hat. Der Host ist typischerweise eine CPU, die Devices sind eine GPU und die CPU selbst. Die Hostanwendung baut jeweils eine Command Queue zu den CUs auf, über die Befehle wie z.B. das Kopieren von Daten oder das Ausführen von Kernels laufen; übernommen aus Quelle: [65].

Hierbei erstellt die Hostanwendung sogenannte Command Queues, die eine Verbindung



zwischen dem Host und den CDs herstellen. Durch diese Command Queue kann der Host Befehle an die CDs senden. Diese sind z.B. das Ausführen eines Kernels oder das Lesen und Schreiben in den globalen Speicher des CDs. Wichtig ist an dieser Stelle, dass es eine Queue ist und die Befehle auf den CDs in-order ausgeführt werden. D.h. der Host kann mehrere Befehle in die Queue senden, ohne wissen zu müssen, ob andere Befehle bereits ausgeführt werden. Die typische Reihenfolge der Queue-Befehle sind somit: Daten schreiben, Kernel ausführen und Daten lesen. Zusätzlich können auch mehrere Kernels ausgeführt werden, ohne neue Daten zu schreiben oder alte Daten aus dem CD zu lesen. Dieses Vorgehen ist dann sinnvoll, wenn durch diesen Schritt eine globale Synchronisierung innerhalb des Kernels vermieden werden kann. Zusätzlich ist noch auf eine Besonderheit in Abbildung 5.3 hinzuweisen: Der Host kann neben der Ausführung der Hostanwendung auch als CD genutzt werden. Dies wird in vielen Fällen genauso genutzt, wobei sich bei diesem Vorgehen das Lesen und Schreiben der Daten zum CD erheblich beschleunigt, da die Daten von der Hostanwendung bereits geladen wurden. D.h. hier hat die CPU einen erheblichen Vorteil gegenüber einer GPU, da hier erst die Daten von der Hostanwendung in den globalen Speicher der GPU geladen werden müssen und anschließend wieder gelesen werden. Dieser Sachverhalt wird später im Kapitel *Evaluation* erneut aufgegriffen.

#### 5.1.4 Workgroups und Workitems

Der ausgeführte Kernel auf einem PE wird als Workitem bezeichnet. Die Workitems werden durch Workgroups zusammengefasst. Workgroups werden einzelnen CUs zugewiesen. Je nach zu berechnenden Kernel und wie die Workgroup-Größe gewählt wird, hat die Wahl der Größe der Workgroup und Workitems einen sehr hohen Einfluss auf die Performance. Schon beim Entwickeln der Kernels muss darauf geachtet werden, das z.B. nur innerhalb einer Workgroup auf den gemeinsamen lokalen Speicher zugegriffen werden kann. Im Idealfall sind genauso viele Workitems und Workgroups vorhanden, wie in der Hardware CUs und PEs existieren. Dies ist typischerweise nicht der Fall, oft muss die Partitionierung an die Hardware durch Tests angepasst werden, um die optimale Performance zu erreichen. In Abbildung 5.4 ist ein Beispiel zur Auswirkung von der Workgroups- und Workitems-Größe abgebildet. Abgebildet sind zwei unterschiedliche Varianten a.) und b.) zur Auswirkung der Wahl der Workgroups und Workitems. Die Varianten a.) und b.) sind in der Abbildung jeweils mit einem gestrichelten Kasten umrandet. Beide Möglichkeiten sollen dasselbe Problem lösen und zwar die Addition von zwei Arrays A und B. Für dieses Problem gibt es unterschiedliche Möglichkeiten der Auswahl der Workgroups- und Workitems-Größe. Auf der rechten Seite im schwarz umrandeten Kasten ist ein einfacher Kernel dargestellt, der die Addition der zwei Arrays durchführen soll. Durch die Codezeile `get_global_id` wird die Id jedes einzelnen ausgeführten Workitem auf den PEs bestimmt und ermöglicht jedem ausgeführten Workitem auf einen anderen Array-Bereich zuzugreifen. Das Array reicht in beiden Fällen a.) und b.) von 0 bis 15 (siehe Abbildung kleine aneinandergereihte Kästchen). Im Beispiel a.) wird festgelegt, dass die Workgroup-Größe aus vier Workitems besteht, d.h. insgesamt werden vier Workgroups benötigt, um 16 Elemente (von 0 bis 15) zu berechnen. In dieser Variante wird

jeweils eine Workgroup auf einer CU mit jeweils vier PEs ausgeführt. Da die Hardware aus zwei CUs mit jeweils vier PEs besteht, ist das System voll ausgelastet. Im Beispiel b.) wird eine Workgroup-Größe von 16 Workitems gewählt (0 bis 15) und diese wird ebenfalls mit der gleichen Hardware wie im ersten Beispiel a.) ausgeführt. Da hierbei nur eine Workgroup vorhanden ist, wird sie nur auf einer CU Unit berechnet. Bei dieser Wahl der Workgroup-Größe wird also nur 50% der Hardware ausgelastet.

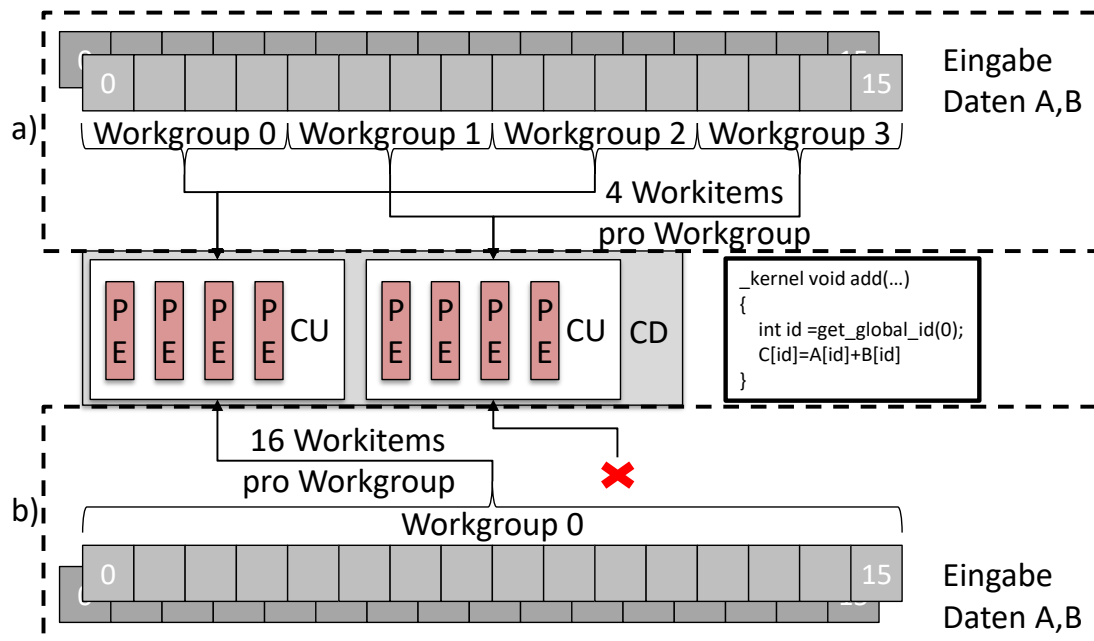


Abbildung 5.4: Auswirkungen der Workgroup- und Workitems-Größe.

## 5.2 NBTI-Simulation

In diesem Abschnitt werden die Umsetzung des vorgestellten NBTI-Modells aus dem Kapitel *Konzepte und Modelle* erläutert. Zuerst wird das Erstellen der Zeitkonstanten  $\tau$  erläutert und anschließend die Umsetzung des NBTI-Modells.

### 5.2.1 Zeitkonstanten und TCAD-Simulation

Durch TCAD-Simulationen (Technology Computer Aided Design) können die Zeitkonstanten  $\tau$  für die Transistoren, die für die NBTI-Simulation benötigt werden, gewonnen werden. Die Parameter dieser TCAD-Simulationen basieren auf echten Silizium-Messungen der Traps[61]. Bei diesen Messungen werden die Transistoren durch ein Stresssignal belastet und anschließend wird die Heilung der Transistoren gemessen. Das Ergebnis der TCAD-Simulation sind pro Trap zwei Dateien für zwei unterschiedliche Temperaturen, um die Temperaturabhängigkeit der Zeitkonstanten zu erhalten. Der Inhalt jeder einzelnen Trap-Datei sind die Werte für  $\tau_e$  und  $\tau_c$  für unterschiedliche Stressspannungen. Je nach gewählter Transistor Technologie kann die Anzahl an Traps pro Transistor bei einigen Tausend liegen. Der Inhalt einer solchen Trap-Datei mit der Dateierweiterung ".crv" ist im Anhang B.0.2 dargestellt.

### 5.2.2 Ablauf

Die NBTI-Simulation wird so durchgeführt, wie der Ablauf im Kapitel *Konzepte und Modelle* vorgestellt wurde. Mit Hilfe der Gleichungen 4.4, 4.5 und 4.6 lassen sich die Zustände der einzelnen Traps der Transistoren berechnen. Hierbei können die einzelnen Traps unabhängig voneinander berechnet werden. Nur die Berechnung des Schwellspannungsschadens benötigt die Rechenergebnisse aller Traps (Gleichung 4.5), d.h. die Berechnung der Trap-Wahrscheinlichkeiten ist besonders stark parallelisierbar. Daher bietet sich die Implementierung in einer Programmiersprache zur Parallelisierung von Berechnungen wie OpenCL besonders an. Bei eigenen Tests hat sich herausgestellt, dass eine Berechnung bei einfacher Genauigkeit (Single Precision (SP), 32 Bit Floating Point) zu erheblichen Rundungsfehlern beim Simulieren der Traps führt. Daraus ergibt sich, dass sich die Simulation zwar sehr stark parallelisieren lässt, allerdings dabei die Anforderung an die Hardware hat, dass die Berechnungen mit doppelter Genauigkeit (Double Precision (DP), 64 Bit Floating Point) durchgeführt werden müssen. Da nicht jede OpenCL-Hardware DP-Berechnungseinheiten hat, bzw. gerade die Performance von DP sehr stark bei unterschiedlicher Hardware ist, muss darauf bei einer späteren Ausführung der Simulation geachtet werden.

Anhand dieses vorgestellten Ablaufs wurde das NBTI-Modell umgesetzt, diese Ausführung wird im nächsten Unterkapitel zusammengefasst.

### 5.2.3 Simulation

In Abbildung 5.5 ist die entwickelte NBTI-Simulation des Stressszenarios zu sehen. Hierbei werden in der Vorverarbeitung die Trapdaten eingelesen und für die Berechnung der Traps vorbereitet. Bei dieser Vorverarbeitung wird die Anzahl der Stressspannungen für die  $\tau_e$  und  $\tau_c$  Werte auf das Stressszenario angepasst. Typischerweise haben die Spannungen eine Schrittweite der  $\tau$  Werte von 0.1 Volt, also eine zu grobe Auflösung, die durch Interpolation vergrößert werden muss. Außerdem wird in diesem Bearbeitungsschritt der Ausschnitt der Werte auf den genutzten Wertebereich des Stressszenarios eingegrenzt. Hat das Stressszenario z.B. eine maximale Versorgungsspannung von 1 Volt, dann wird der Wertebereich für unterschiedliche Stressspannungen darauf eingeschränkt, also von 0 bis 1 Volt. Gleichzeitig wird die benötigte Auflösung z.B. in 0.01 Volt Schritten interpoliert. Dieses Vorgehen verringert zum einen die Menge der  $\tau$  Werte auf die die im passenden Stressspannungsbereich liegen und zum anderen erspart die kleinere Schrittweite die Interpolation in OpenCL, da eine höhere Auflösung als 0.01 Volt bei der Stressspannung nicht benötigt wird. Ein Beispiel dazu: Für einen Spannungsbereich von 0 bis 1 Volt bei 0.01 Volt Schritten, sind dies 200  $\tau_e$  und  $\tau_c$  Werte. Hat die Transistor-Technologie z.B. 1000 Traps pro Transistor, so müssen insgesamt 200000  $\tau$  Werte für die Berechnung gespeichert werden. Bei der Berechnung der Traps in OpenCL können direkt die passenden  $\tau_e$  und  $\tau_c$  Werte aus dem Speicher genutzt werden.

Die Berechnung der Temperaturabhängigkeit der  $\tau$  Werte wird durch eine Interpolation in der OpenCL Implementierung umgesetzt. Hierzu bestehen die  $\tau$  Werte aus Daten für zwei unterschiedliche Temperaturen, die dann zur Interpolation genutzt werden.

Im nächsten Schritt in Abbildung 5.5 wird, wie bereits im Kapitel *Konzepte und Modelle* vorgestellt, aus dem Stressszenario die maximale Temperatur, Spannung und Signalwahrscheinlichkeit ermittelt. Mit diesen Werten wird für die Traps eines Transistors eine NBTI-Simulation durchgeführt, um anschließend die Traps auszusortieren, die keinen oder nur einen geringen Beitrag zu dem Schwellspannungsschaden bei diesem Stressszenario haben. An dieser Stelle kann die Genauigkeit des NBTI-Modells, die mit den reduzierten Traps erreicht wird, frei gewählt werden und so kann auch die Performance durch eine Verringerung der Traps pro Transistor weiter gesteigert werden. Die reduzierte Trapliste wird anschließend in short-term und long-term Traps unterteilt. Mit der long-term Trapliste wird dann für jeden Transistor und dem Stressszenario eine NBTI-Simulation durchgeführt. Diese Berechnung erfolgt in OpenCL. Hierbei werden die Daten für die  $\tau$  Werte und das Stressszenario benötigt, die aus den Spannungswerten, den Temperaturen und der Signalwahrscheinlichkeit des Transistors bestehen.

Im Anschluss werden die Schwellspannungsschäden für die short-term Traps zu den long-term Schäden dazugerechnet. Das Ergebnis ist der Schwellspannungsschaden für jeden einzelnen Transistor des Stressszenarios. Die Schwellspannungsschäden werden dann anschließend für die Delay-Simulation genutzt.

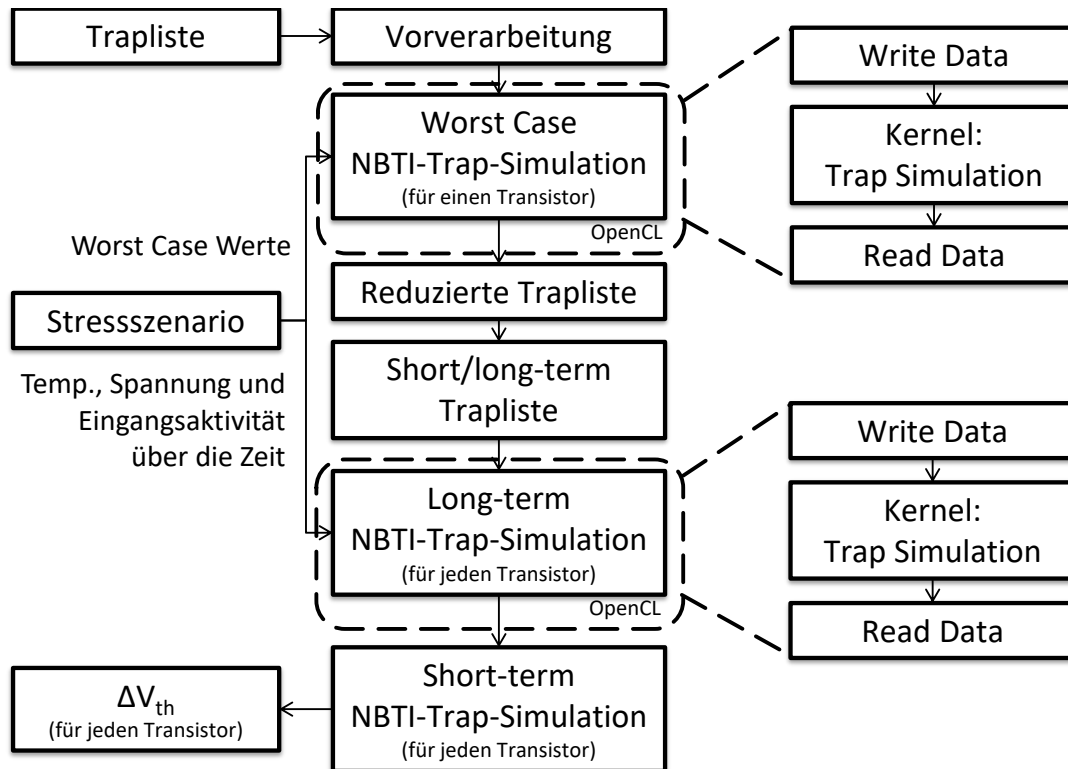


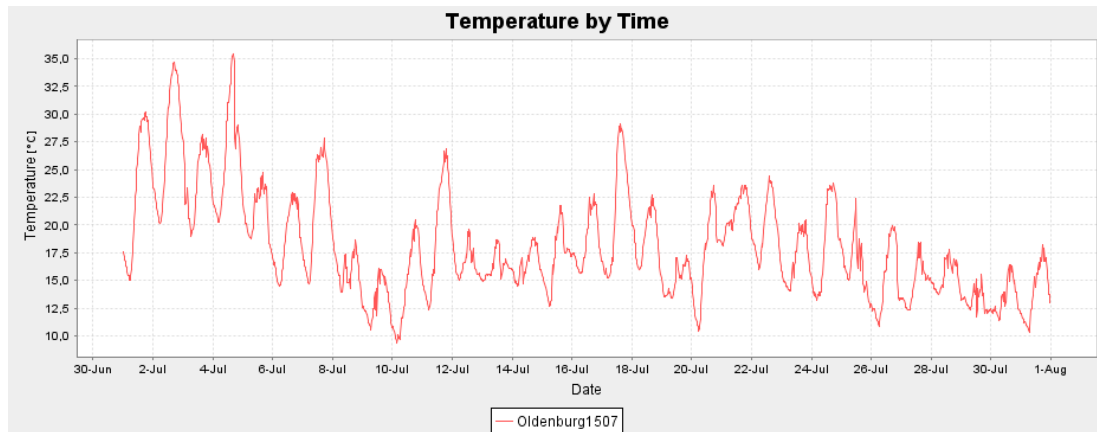
Abbildung 5.5: Simulation der NBTI-Alterung mit Hilfe des vorgestellten NBTI-Modells. Die Kernidee basiert darauf, die Trapanzahl pro Transistor massiv zu verringern und gleichzeitig die Parallelität des vorgestellten NBTI-Modells für die Trap-Simulation auszunutzen. Die NBTI-Trap-Simulationen finden mit Hilfe von OpenCL statt. Die einzelnen Schritte wurden aus Abbildung 4.11 übernommen

### 5.3 Missions- und Stressszenario

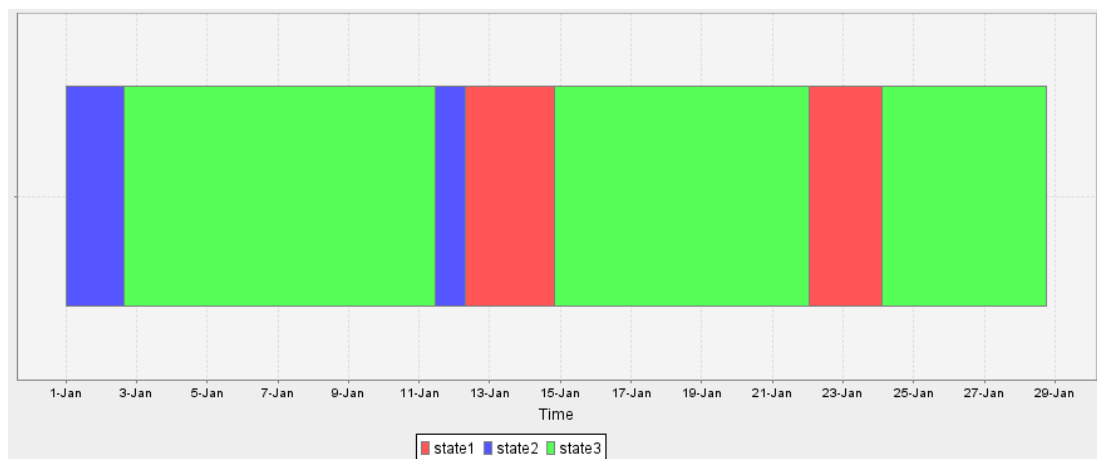
Das Missionsszenario wurde im Kapitel *Konzepte und Modelle* durch die Anforderungen des Alterungsmodells definiert. Es besteht, wie in Gleichung 4.9 definiert wurde, aus den Zuständen, die das System einnehmen kann, und der Umgebungstemperatur über die Zeit. Im Laufe dieser Arbeit ist ein Missionsszenarien-Editor entstanden, der im Anhang B.0.9 zu sehen ist. Mit diesem Editor ist es möglich, aus einzelnen Zuständen des Systems und Temperaturverläufen, ein Missionsszenario zu definieren. Hierfür wird die Umgebungstemperatur über die Zeit eingegeben bzw. aus vorhanden Datensätzen geladen. Dies kann z.B., wie dies in Abbildung 5.6 dargestellt ist, die gemessene Außentemperatur von Oldenburg im Juli 2015 sein. An dieser Stelle können auch mehrere gespeicherte Datensätze verbunden und bearbeitet werden.

Als nächstes wird der zeitliche Verlauf der Zustände des zu simulierenden Systems eingegeben. Auch hier können vorhandene Datensätze verbunden und bearbeitet werden. Ein Beispiel ist in Abbildung 5.7 dargestellt, bei dem mehrere Zustände und der zeitliche Verlauf erstellt wurden.

Abschließend erfolgt die Verbindung der Temperaturen und der Zustände zu dem Missionsszenario. In diesem Schritt wird angegeben, wie lange das Missionsszenario laufen soll, also wie das System mit diesen Zuständen altern soll. Ist die gewünschte Laufzeit länger als



**Abbildung 5.6: Erzeugter Temperaturverlauf für ein Missionsszenario. Dieser wird auch im Anhang B.0.9 dargestellt. Dieser Temperaturverlauf wurde mit dem Missionsszenario-Editor erstellt. Die Temperatur ist die gemessene Außentemperatur von Oldenburg im Juli 2015. Quelle der Daten [58]**



**Abbildung 5.7: Zeitlicher Ablauf der Zustände eines Systems. Dargestellt sind drei unterschiedliche Zustände durch unterschiedliche Farben.**

der zuvor beschriebene Temperaturverlauf oder die Abfolge der Zustände, so werden diese zeitlichen Verläufe solange wiederholt, bis sie dem gewünschten Alterungszeitraum entsprechen. In Abbildung 5.8 ist die Verbindung aus dem vorherigen Temperatur- und Zustandsbeispiel dargestellt. Bei diesem Beispiel wurde ein Missionsszenario von zwei Monaten gewählt, d.h. der zuvor definierte Temperatur- und Zustandsverlauf wird zweimal wiederholt. Zusätzlich kann es bei der Verbindung der Temperaturen und Zustände sinnvoll sein, die Anzahl der Temperaturen zu reduzieren, da z.B. zu viele Temperaturdaten vorhanden sind. Hier bietet es sich an, die Durchschnittstemperatur für jeden der Zustände zu berechnen und diesen als Temperaturverlauf zu nutzen.

Der Missionsszenario-Editor wurde eigentlich zur Bearbeitung und Erstellung der Missionsszenarien entwickelt. Er kann auch ohne weiteres als alternativer Weg zur direkten Erstellung bzw. Bearbeitung von Stressszenarien genutzt werden, ohne anschließend eine Temperatur,

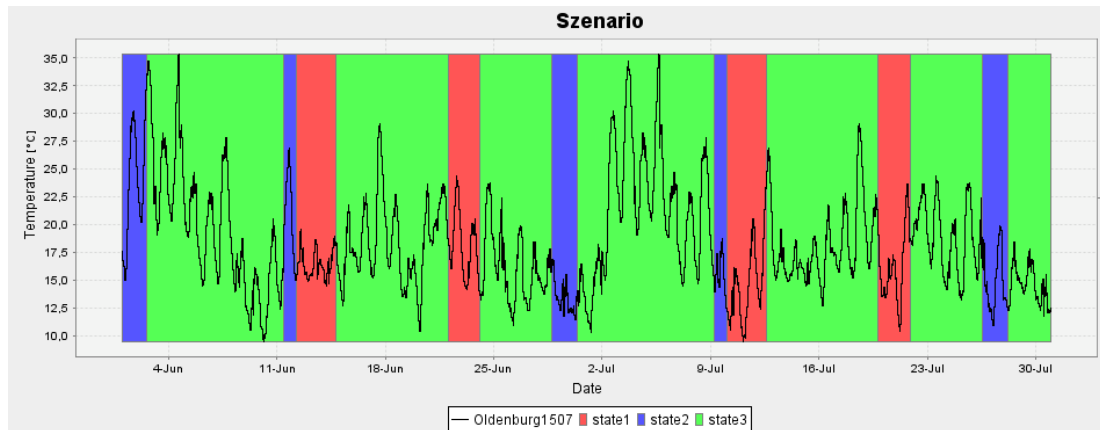


Abbildung 5.8: Abgebildet ist ein Missionsszenario, das aus den Temperaturen und dem Zustandsverlauf aus den Abbildungen 5.6 und 5.7 entstanden ist. Die Laufzeit wurde auf zwei Monate festgelegt, d.h. da der Temperatur- und Zustandsverlauf ursprünglich einen Monat umfasst, wurde genau dieser Verlauf bis zum Laufzeitende wiederholt.

IR-Drop und elektrothermische Kopplung durchzuführen. Da ein Stressszenario für NBTI, wie dies in Gleichung 4.10 definiert wurde, aus der Versorgungsspannung, der Temperatur und der Signalwahrscheinlichkeit besteht, so kann im Editor statt der Umgebungstemperatur direkt die Temperatur des zu simulierenden Systems eingegeben werden. Die Zustände bestehen dann wie vorher schon aus der Signalwahrscheinlichkeit und z.B. aus einer vorgegebenen Versorgungsspannung. In Abbildung 5.9 ist der Ausschnitt eines Stressszenarios dargestellt.

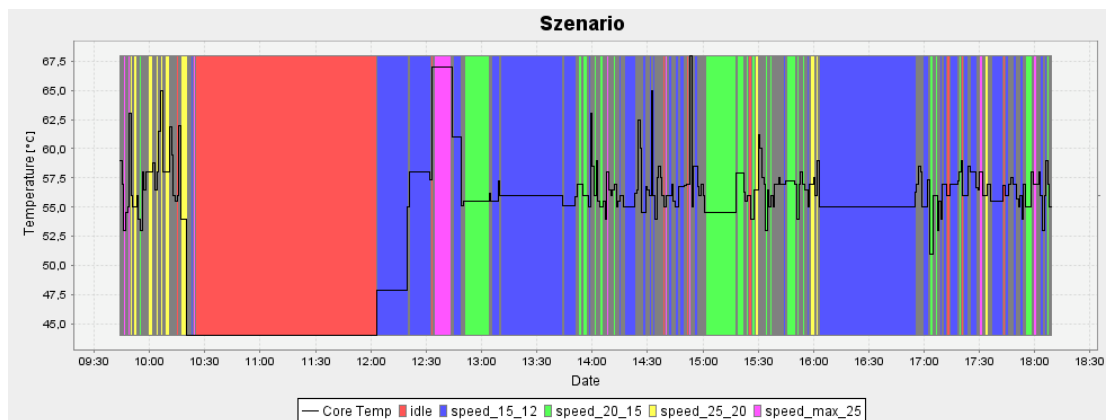


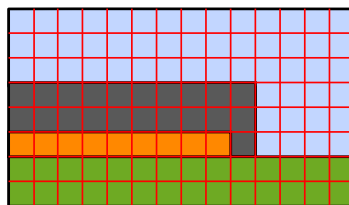
Abbildung 5.9: Erstelltes Stressszenario mit dem Missionsszenario-Editor. Statt einer Umgebungstemperatur wird direkt im Editor die Stresstemperatur angegeben. Die Temperatur wurde, z.B. durch eine Mittelwertbildung, an die Anzahl der Werte so angepasst, dass pro Zustand eine Temperatur vorhanden ist. Es wird für die Alterungssimulation keine Simulation eines Missionsszenarios durchgeführt, sondern direkt mit dem Stressszenario eine Alterungssimulation.

## 5.4 Thermische Simulation und Package

Die Voraussetzung, um eine thermische Simulation durchführen zu können, sind das Package, die Verlustleistung und die Position der Verlustleistung. Mit dem vorgestellten Package-Modell und dem thermischen Modell wird in diesem Abschnitt die Implementierung der Modelle und der Ablauf zur Anwendung dieser vorgestellt. Begonnen wird mit der Erstellung eines Packages und der Rasterung. Anschließend wird auf die thermische Charakterisierung und die eigentliche thermische Simulation eingegangen.

### 5.4.1 Package-Rasterung

Das Package ist die physikalische Beschreibung der Materialien und die Position des zu simulierenden Systems. Die Materialien bestehen aus den physikalischen Eigenschaften Wärmeleitfähigkeit  $\kappa$ , Dichte  $\rho$  und Wärmekapazität  $C_p$ , wie dies bereits im Kapitel *Konzepte und Modelle* beschrieben wurde. Das Package wird mit Hilfe eines CAD-Tools (Computer Aided Design) entworfen. Hierzu wurde in dieser Arbeit neben eines einfachen Material-Editors auch ein Plug-in für das OpenSource-CAD-Programm Blender entwickelt. Dieses Plug-in erweitert Blender mit der Möglichkeit, Objekten Materialien mit den physikalischen Eigenschaften zuzuweisen. Zusätzlich bietet das Plug-in die Möglichkeit, das entworfene Package als XML-Datei zu ex- und importieren. Im Anhang B.0.6 ist der einfache Material-Editor und die Nutzung des Plug-ins dargestellt. In einer weiteren Abbildung im Anhang B.0.6 sind Beispiele für erstellte Packages dargestellt. Dieses erstellte, kontinuierlich beschriebene Package, muss für die thermische Charakterisierung gerastert werden. Es entsteht hierbei eine diskrete Package-Beschreibung. Je nach Wahl der Rasterung kann es zu neuen Metamaterialien kommen, wie dies im Kapitel *Konzepte und Modelle* vorgestellt wurde. Der Ablauf der Rasterung sieht folgendermaßen aus: Für eine vorgegebene Rastergröße wird das kontinuierliche Package analysiert. Bildlich gesprochen wird das Package in kleine Würfel (Rasterblöcke) geschnitten, dabei wird eine Rastergröße vorgegeben. Dies ist in Abbildung 5.10 dargestellt.

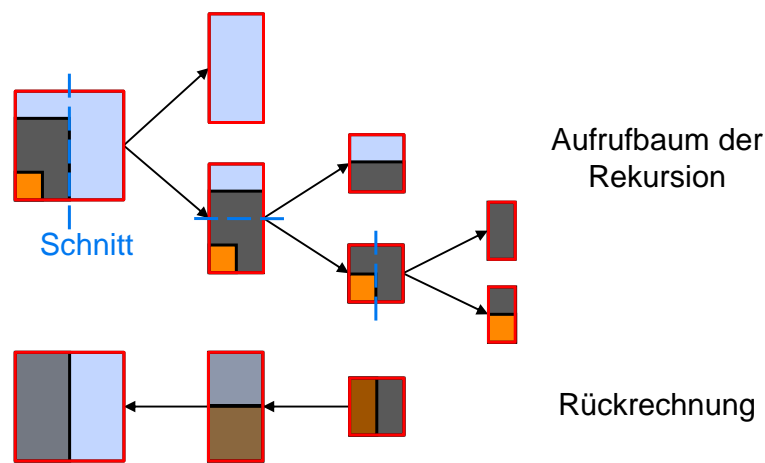


**Abbildung 5.10:** Diskretisierung eines kontinuierlich beschriebenen Package. In roter Farbe sind die Rasterblöcke dargestellt. Die Darstellung wurde aus der Abbildung 4.14 übernommen.

Besteht nun ein Rasterblock nur aus einem Material, so werden die physikalischen Eigenschaften aus der kontinuierlichen Beschreibung übernommen. Besteht der Rasterblock aber aus mehr als einem Material, so müssen über die Gleichungen 4.11, 4.12, 4.13 und 4.14 die neuen



physikalischen Eigenschaften berechnet werden. Hierbei können die Berechnungen, wie dies bereits das Beispiel aus dem Kapitel *Konzepte und Modelle* gezeigt hat, schnell komplex werden. Daher bietet es sich an, folgende, aus Implementierungssicht, einfache Lösung zu wählen: Besteht der Rasterblock aus zwei Materialien, die parallel oder in Reihe liegen, so kann das neue Material aus den obigen Gleichungen berechnet werden. Besteht der Rasterblock aber aus mehr als zwei Materialien, bzw. aus einer komplexen Materialverteilung, so wird die Berechnung rekursiv durchgeführt. D.h. dieser Rasterblock wird solange in kleinere Rasterblöcke zerschnitten, bis wieder eine einfache Berechnung möglich ist. Danach wird das einfache Ergebnis an die vorher komplexe Berechnung zurückgegeben, um diese wiederum zu einer einfachen Berechnung zu machen. Implementiert ist dies durch eine Rekursion. In Abbildung 5.11 ist dieses Vorgehen anhand des komplexeren Beispiels aus dem Kapitel *Konzepte und Modelle* dargestellt.



**Abbildung 5.11:** Beispiele zur Berechnung der Materialeigenschaften eines Rasterblocks, der aus mehreren Materialien besteht. Der Rasterblock wird so oft unterteilt, bis die neu entstandenen Blöcke einfach zu berechnen sind. Dies wird durch eine Rekursion durchgeführt. In der Abbildung ist der Aufrufbaum, der bei der Berechnung des komplexen Beispiels aus Abbildung 4.15 entsteht, dargestellt. Die Blätter des Baumes sind mit den Gleichungen aus dem Kapitel *Konzepte und Modelle* direkt berechenbar. Die Rückrechnung des Aufrufbaums ist im unteren Bereich der Abbildung dargestellt und verdeutlicht, wie sich durch Rekursion das komplexe Beispiel in einfache Rechnungen unterteilt.

Das während dieser Arbeit entstandene Programm zur Rasterung des Packages, der Package-Slicer, ist im Anhang B.0.7 näher erläutert. Es liest das exportierte Package im XML-Format ein, das mit Hilfe von Blender entworfen werden kann und speichert das gerasterte Package in einem XML-Format. Zu beiden Formaten, der kontinuierlichen und diskreten Package-Beschreibung, ist ein Beispiel im Anhang B.0.6 und B.0.7 zu finden.

## 5.4.2 Position der Komponenten

Als nächstes wird für die thermische Simulation die Position der einzelnen Komponenten innerhalb des Chips benötigt. Hierbei gibt es bei dem vorgestellten thermischen Modell keine Einschränkung auf rein 2-dimensionale Systeme, die nur eine aktive Transistorebene haben. Mit

dem genutzten Modell können ohne Einschränkung auch Multilayer-Systeme simuliert werden, die mehrere aktive Transistorebenen haben.

Außerdem müssen bei der thermischen Simulation die Komponenten nicht zwangsläufig den Registertransferkomponenten eines Designs entsprechen, sondern es kann sich an dieser Stelle anbieten, z.B. größere Registertransferkomponenten weiter zu unterteilen oder kleinere zusammenzufassen. Dieses Vorgehen des Aufteilens macht an den Stellen Sinn, an denen eine höhere Auflösung der Temperaturen benötigt wird, bzw. die Zusammenfassung, wenn bei bestimmten Komponenten keine detailliertere räumliche Auflösung der Temperaturen gebraucht wird. Zusätzlich sollte auch nicht außer Acht gelassen werden, dass mit steigender Anzahl an Komponenten der Rechenaufwand der thermischen Simulation ansteigt.

Ausgehend von einer Netzliste auf Gatterebene ist es möglich mit Hilfe von einer LEF<sup>2</sup>-Datei ein Floorplaning mit Industrietools durchzuführen, z.B. kann hierfür das Industrietool Encounter von Cadence genutzt werden. Das Ergebnis ist ein platziertes und geroutetes Design, das in einer DEF<sup>3</sup>-Datei gespeichert werden kann. Die Positionsinformationen zu den Komponenten können über die DEF-Datei oder direkt aus dem Floorplaner, z.B. über eigene TCL-Skripte (Anhang B.0.1) in eine XML-Datei exportiert werden.

Die Positionen der Komponenten werden zur thermischen Simulation zwar zwingend benötigt, aber es muss an dieser Stelle nicht zwangsläufig die Informationen aus den Industrietools genutzt werden, da diese Informationen erst zu einer sehr späten Designphase zur Verfügung stehen. D.h. auch eine Flächenabschätzung der Komponenten und ein damit durchgeführtes frühes Floorplanning auf Systemebene reicht für die Durchführung einer thermischen Simulation mit den in dieser Arbeit vorgestellten Modellen aus.

### 5.4.3 Thermische Charakterisierung

Um eine thermische Simulation mit dem vorgestellten Modell durchführen zu können, wird eine Charakterisierung des Packages benötigt. Dies bedeutet, es müssen die thermischen Impulsantworten der einzelnen Komponenten des Systems bestimmt und für die spätere Simulation gespeichert werden. Hierbei wird diese Charakterisierung für jede Komponente des Systems durchgeführt. Durch dieses Vorgehen werden Probleme<sup>4</sup>, die andere thermische Ansätze haben, die auf den Green-Funktionen basieren, durch die einzelnen Charakterisierungen gelöst. In Abbildung 5.12 ist der Ablauf zur thermischen Charakterisierung dargestellt. Der erste Schritt ist die Package-Rasterung, wie sie im vorherigen Unterkapitel beschrieben wurde. Das Ergebnis ist das gerasterte Package, mit diesem wird anschließend eine thermische Simulation durchgeführt, bei dem jede einzelne Komponente mit einem Impuls angeregt wird und die Impulsantworten für die einzelnen Komponenten gespeichert werden.

<sup>2</sup>LEF=Library Exchange Format: beinhaltet Informationen zu den einzelnen Gattern einer Gatterbibliothek

<sup>3</sup>DEF=Design Exchange Format: beinhaltet das komplette physikalische Layout eines Designs

<sup>4</sup>Andere bereits Veröffentlichte Ansätze haben Probleme mit Randwerten des Packages (Randwertproblem), aber auch mit nicht homogenen Materialien innerhalb der aktiven Ebene wie z.B. durch Silizium-Durchkontaktierungen (through-silicon via, TSV). Für diese Probleme zu lösen müssen bei anderen Ansätzen Ausgleichsrechnungen durchgeführt werden[87].

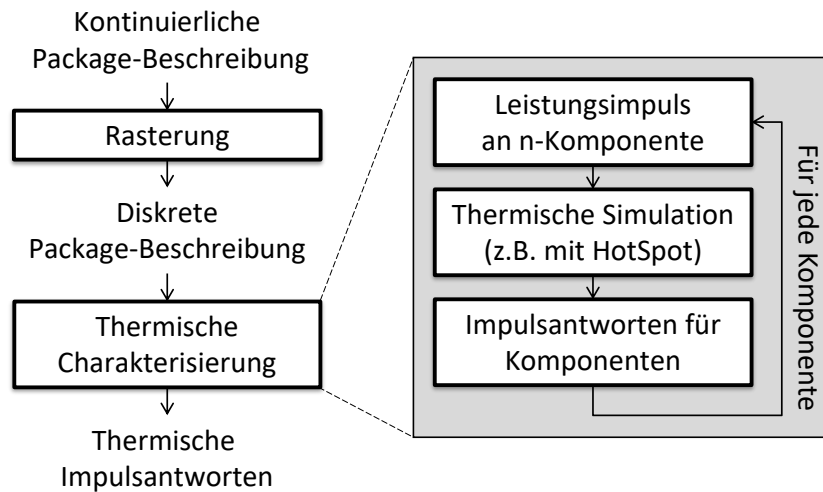


Abbildung 5.12: Dargestellt sind die Vorverarbeitungsschritte, die für die thermische Simulation benötigt werden: Die Rasterung des Packages und die Charakterisierung des Packages. Von einer kontinuierlichen Package-Beschreibung, die mit einem CAD-Tool erzeugt wurde, wird durch Rasterung eine diskrete Package-Beschreibung erstellt. Anschließend wird eine thermische Charakterisierung des Packages durchgeführt. Hierbei werden für jede Komponente, von der später die Temperatur benötigt wird, die Impulsantworten simuliert. Dies wird mit Hilfe einer thermischen Simulation durchgeführt, in dieser Arbeit wird hierzu das thermische Simulationstool HotSpot verwendet.

Der Ablauf der thermischen Charakterisierung ist auch noch einmal an einem Beispiel in Abbildung 5.13 dargestellt. In diesem Beispiel wird die Charakterisierung für ein System mit vier Komponenten durchgeführt.

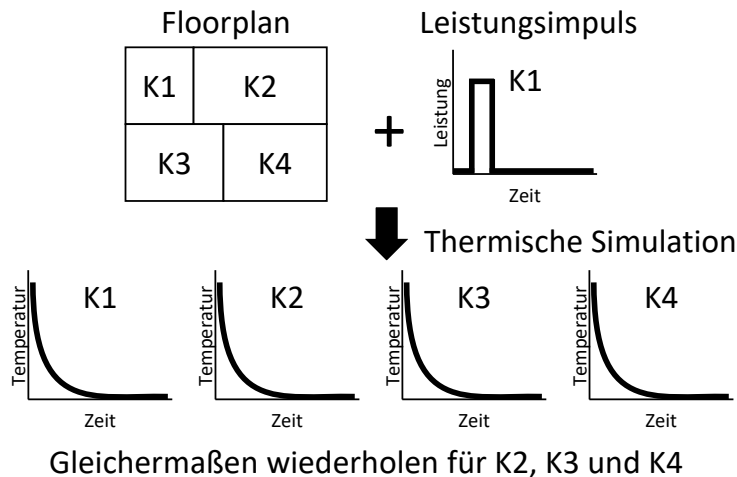


Abbildung 5.13: Beispiel einer thermischen Charakterisierung bei einem System mit vier Komponenten. Für jede Komponente wird eine thermische Simulation mit einem Leistungsimpuls durchgeführt und die Impulsantworten werden jeweils gespeichert. Dies bedeutet für das Beispiel, dass vier thermische Simulationen durchgeführt werden müssen und insgesamt 16 Impulsantworten gespeichert werden.

Für die Simulation der Impulsantworten wird in dieser Arbeit das thermische Simulationstool HotSpot eingesetzt[75]. Das Tool ist Open Source und wird als Standardtool im wissenschaftlichen

Bereich genutzt. Um es mit dem entwickelten, vorgestellten Package-Modell zu verwenden, wurde es um eine Einlese-Funktion für das gerasterte XML-Package-Format erweitert. Im nächsten Abschnitt geht es darum, wie die Charakterisierung genutzt wird, um eine thermische Simulation durchzuführen.

#### 5.4.4 Thermisches Modell

Mit der im letzten Abschnitt vorgestellten thermischen Charakterisierung, der Verlustleistung der einzelnen Komponenten und der Gleichung 4.19 aus dem Kapitel *Konzepte und Modelle* kann nun die thermische Simulation durchgeführt werden.

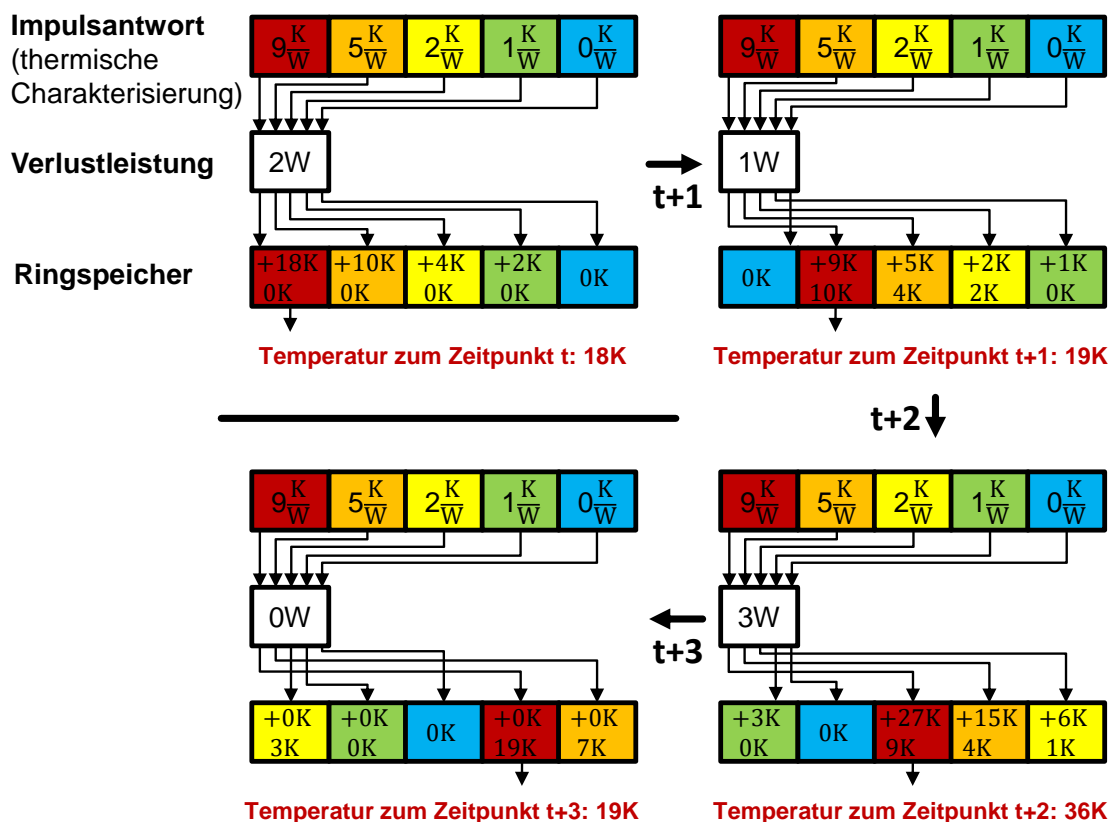


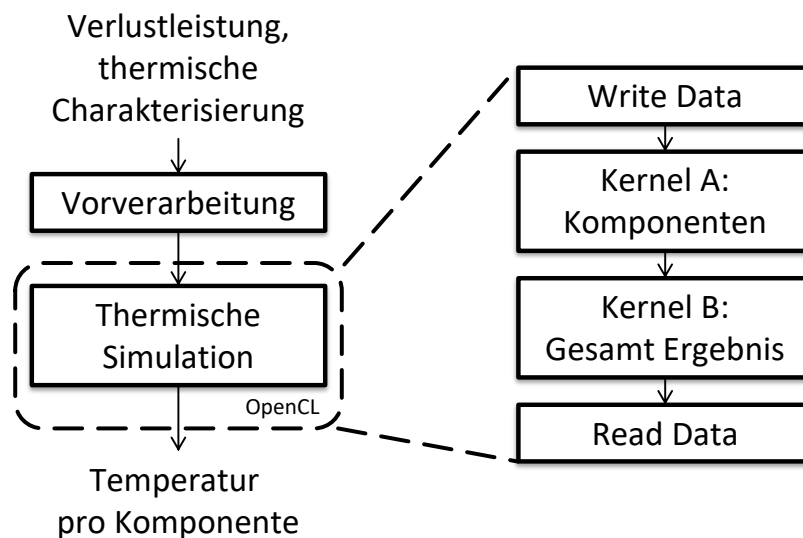
Abbildung 5.14: Beispiel zur Umsetzung der thermischen Simulation: In diesem Beispiel wird die Temperatur einer Komponente berechnet. Als Eingabe wird die Impulsantwort und die Verlustleistung der Komponente benötigt. Ausgehend von der Impulsantwort wird jeder Wert der Impulsantwort mit dem Verlustleistungswert multipliziert und zu den vorhandenen Werten in den Ringspeicher addiert. Das Temperaturergebnis der Komponenten liegt an der Startspeicheradresse, die im ersten Zeitschritt an der ersten Stelle im Ringspeicher liegt. Bei jedem Zeitschritt bewegt sich die Startspeicheradresse im Ringspeicher einen Schritt weiter. Außerdem wird der zeitlich gesehene älteste Wert (Startspeicheradresse - 1) im Ringspeicher auf Null gesetzt. In der Abbildung ist die Berechnung für vier Zeitschritte dargestellt, die Verlustleistung ist für diese Zeitschritte  $2W$ ,  $1W$ ,  $3W$  und  $0W$ . Die Ergebnistemperaturen für diese Zeitschritte sind  $18K$ ,  $19K$ ,  $36K$  und  $19K$ .

Das Vorgehen der Berechnung wird anhand des Beispiels in Abbildung 5.14 genauer veranschaulicht. In diesem Beispiel ist die Berechnung der Temperatur für eine Komponente dargestellt. Dabei wird jeder Wert der thermischen Impulsantwort mit der aktuellen Verlustleistung der Komponente multipliziert und in einen Ringspeicher abgelegt. Hierbei werden bereits gespeicherte Temperaturen aus vorherigen Zeitschritten, die im Ringspeicher liegen, zu den Berechnungen addiert. Dieser Schritt berechnet, wie die Verlustleistung die Komponente zum aktuellen Zeitpunkt und in den zukünftigen Zeitschritten erwärmt.

Zusätzlich gibt es einen Zeiger, der auf die Position der aktuellen Temperatur im Ringspeicher zeigt. Nachdem die Berechnungen für diesen Zeitschritt abgeschlossen sind, wird die Temperatur, die an der Zeiger-Position gespeichert ist, als aktuelle Temperatur der Komponente ausgegeben. Anschließend wird der Wert an dieser Zeiger-Position auf Null gesetzt. Dieser Schritt spiegelt den Sachverhalt wieder, dass eine Verlustleistung nur eine endliche Zeit eine Erwärmung des Systems in der Zukunft hervorruft und der älteste Temperaturwert im Ringspeicher auf Null gesetzt wird. Als letzter Schritt rückt die Position des Zeigers um eine Stelle weiter.

Dieses Vorgehen der Temperaturberechnung wird für jeden weiteren Zeitschritt wiederholt.

In diesem Beispiel wurde nur die Eigenerwärmung der Komponente berechnet. Besteht das System aus mehr als einer Komponente, so muss bei jedem Zeitschritt zusätzlich noch die Erwärmung durch die übrigen Komponenten mit eingerechnet werden. In diesem Beispiel wurde also nur der erste Teil der Gleichung 4.19 vor dem Summenzeichen berechnet, da es nur eine Komponente gab.



**Abbildung 5.15:** Abgebildet ist der Ablauf der thermischen Simulation. Die eigentliche Berechnung wird mit OpenCL durchgeführt. Die Verlustleistung und die thermische Charakterisierung werden in der Hostanwendung geladen und vorverarbeitet, d.h. in die richtige Datenstruktur gebracht. Anschließend werden die Daten an das Compute Device übertragen.

Der gesamte Vorgang zur Durchführung der thermischen Simulation ist in Abbildung 5.15 dargestellt. In einer Vorverarbeitung werden die Daten für die Simulation aufbereitet und dann

auf das OpenCL Compute Device übertragen.

Anschließend wird Kernel A auf dem Device ausgeführt, hierbei wird die Berechnung aus Gleichung 4.19 verwendet. Der Kernel teilt die Berechnung auf  $N$  Workitems auf, wobei  $N$  die Anzahl an Komponenten ist. Jeder der Workitems benutzt einen Teil des Speichers als Ringspeicher, in dem die Ergebnisse der Temperaturberechnungen zwischengespeichert werden, wie dies bereits im Beispiel erläutert wurde. Wie im Beispiel gibt es auch einen globalen Zeiger, der auf die aktuelle Position im Speicher zeigt, an dem die Temperatur zum aktuellen Zeitschritt liegt. Nach der Berechnung der Temperaturen wird Kernel B ausgeführt, dieser wird nur durch ein Workitem ausgeführt. Der Kernel hat die Aufgabe, die aktuellen Temperaturen aus dem Speicher auszulesen und anschließend den aktuellen Wert, auf den der globale Zeiger zeigt im Ringspeicher jeder Komponente auf Null zu setzen. Als letztes wird der globale Zeiger um eine Stelle im Speicher verschoben.

## 5.5 Simulation der Verlustleistung

In diesem Unterkapitel wird die Berechnung der Verlustleistung erläutert. Das Berechnen der Verlustleistung wird auf die Gatterebene beschränkt. Die Arbeit leistet dabei keinen neuen Beitrag und nutzt hier bereits bekannte Vorgehensweisen[24].

### 5.5.1 Dynamische Verlustleistung

Zur Berechnung der dynamischen Verlustleistung wird, wie in Abbildung 5.16 dargestellt, eine Schaltungssimulation mit Hilfe von Testbenches durchgeführt. Das Ergebnis der Schaltungssimulation ist die Signalaktivität der Schaltung, die typischerweise in einer Value Change Dump (VCD) Datei gespeichert wird. Das Format speichert die Schaltaktivitäten im ASCII-Format. Hierbei können sehr schnell große Datenmengen entstehen. Neben dem VCD-Dateiformat gibt es noch das Switching Activity Interchange Format (SAIF). Hierbei werden die Informationen aus zeitlichen Informationen, d.h. zu welcher Zeit ändern Signale ihre Werte, zu statischen Informationen, d.h. wie häufig ist ein Signal in einem bestimmten Zustand. In der SAIF-Datei stehen auch die Signalwahrscheinlichkeiten, die für die Alterungssimulationen benötigt werden. Eine Konvertierung von VCD zu SAIF wird von den meisten Schaltungssimulatoren angeboten. Mit der Angabe zur Versorgungsspannung, Schaltfrequenz und der Lastkapazität kann aus Gleichung 2.4 aus dem Kapitel zu den Grundlagen dann die dynamische Verlustleistung berechnet werden.

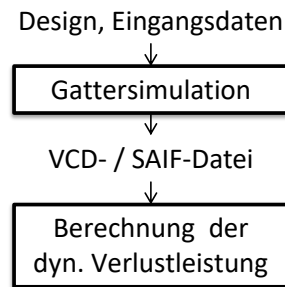


Abbildung 5.16: Berechnung der dynamischen Verlustleistung, wie sie mit Industriertools umgesetzt wird. Aus dem Design und dem charakteristischen Eingangssignal wird eine Gattersimulation durchgeführt. Anschließend kann aus den erhaltenen Schaltungsaktivitäten die dynamische Verlustleistung berechnet werden.

### 5.5.2 Statische Verlustleistung

Für die Berechnung der statischen Verlustleistung wird, wie diese in der Abbildung 5.17 zu sehen ist, eine SPICE-Simulation für die einzelnen Gattertypen einer Gatterbibliothek durchgeführt, um eine Lookup-Tabelle zu erzeugen. Mit Hilfe der Lookup-Tabelle wird dann bei der Berechnung eine Interpolation durchgeführt, um die statischen Verlustleistung der Gatter zu erhalten.

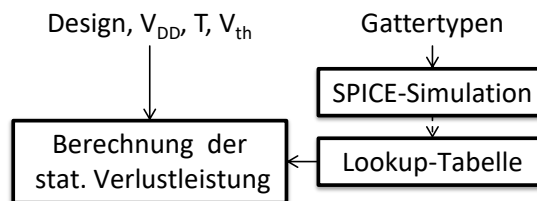


Abbildung 5.17: Berechnung der statischen Verlustleistung. Für jeden Gattertyp der Gatterbibliothek wird eine SPICE-Simulation durchgeführt, um eine Lookup-Tabelle zu erstellen. Diese Charakterisierung muss nur einmalig für eine Gatterbibliothek durchgeführt werden. Die Berechnung der statischen Verlustleistung ist dann ein Interpolieren der Daten aus der Lookup-Tabelle.

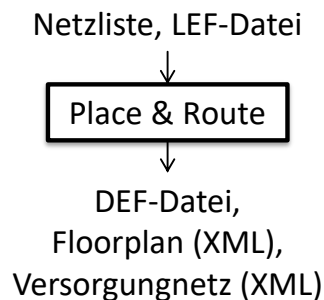
## 5.6 IR-Drop Simulation

Ausgehend von dem vorgestellten Superpositionsansatz im Kapitel *Konzepte und Modelle* wird in diesem Unterkapitel die Umsetzung dieses Konzepts mit Hilfe von OpenCL genauer vorgestellt.

### 5.6.1 Voraussetzungen

Für die Berechnung des IR-Drop wird die Position der Gatter (Floorplan), der Aufbau des Versorgungsnetzes mit den Informationen zum physikalischen Aufbau der einzelnen Metallisierungsebenen benötigt. Hierzu gehören auch die Angaben zum Widerstand der Leitungen: Widerstand pro Länge / Querschnitt und die Verlustleistung der Gatter. In Abbildung 5.18 ist ein Beispiel zur Gewinnung der Informationen, die für die Berechnung des IR-Drop benötigt

werden, dargestellt. Nicht abgebildet ist die Verlustleistung, da diese im letzten Unterkapitel behandelt wurde.



**Abbildung 5.18:** Erstellung der benötigten Informationen für die Berechnung des IR-Drop. Aus der Netzliste und den Layout-Informationen, die in einer LEF-Datei für eine Gatterbibliothek stehen, kann das Design platziert und geroutet werden. Die Ergebnisse können als eine DEF-Datei ausgegeben werden. Zusätzlich wurden Exportskripte entwickelt, die eine einfachere Weiterverarbeitung ermöglichen, da der Floorplan und das Versorgungsnetz als XML ausgegeben werden können.

Hierbei werden die Daten mit Hilfe eigener Exportskripte aus den Daten des Industrietools Encounter von der Firma Cadence gewonnen<sup>5</sup>. Die erhaltenen Informationen werden für die spätere Weiterverarbeitung in XML-Dateien gespeichert. Dies ist zum einen eine Floorplan-Datei mit der Position der Gatter und zum anderen eine Datei, in der die geometrischen Informationen zum Versorgungsnetz stehen. Im Anhang B.0.3 ist der Aufbau und Inhalt der Export XML-Dateien, die in dieser Arbeit entstanden sind, dargestellt. Zusammen mit der Verlustleistung der Komponenten können damit dann die verschiedenen Versorgungsspannungen der Komponenten berechnet werden.

### 5.6.2 Ablauf

Durch den Überlagerungssatz (Superpositionsansatz) können die einzelnen Ströme durch die Widerstände unabhängig voneinander berechnet werden. Dies ergibt den Ablauf, wie er in Abbildung 5.19 dargestellt ist. In einem Vorverarbeitungsschritt werden die Daten, die für die Berechnung benötigt werden, aufbereitet. In diesem Schritt müssen die Leitungswiderstände berechnet werden. Hierzu werden mit Hilfe des Floorplans die Abstände zwischen den einzelnen Komponenten ermittelt und daraus die Widerstände. Außerdem werden die Widerstände und die Verlustleistung in eine passende Datenstruktur für OpenCL gebracht. Diese Schritte geschehen in der OpenCL-Hostanwendung, die anschließend die OpenCL-Berechnung startet.

In Abbildung 5.19 ist auf der rechten Seite der Ablauf der OpenCL-Berechnung dargestellt. Zuerst werden die Daten von der Hostanwendung an das Compute Device gesendet und

<sup>5</sup>Eine andere Herangehensweise ist das Parsen der Formate LEF/DEF, diese enthalten ebenfalls die benötigten Informationen. Der Aufwand, die LEF/DEF Dateien einzulesen, ist deutlich größer als die Exportskripte. Der Austausch von Daten durch XML-Dateien macht eine spätere Anpassung an andere Tools deutlich einfacher. Als Ersatz der Exportskripte kann auch ein LEF/DEF zu XML-Konverter entwickelt werden. Ein Ziel dieser Arbeit ist es unter anderem, dass die einzelnen, entwickelten Module einfach erweiterbar und austauschbar sind, dies gilt auch für die genutzten Datei Formate. Dies ist mit XML als Ein-/Ausgabe Format deutlich besser durchführbar.



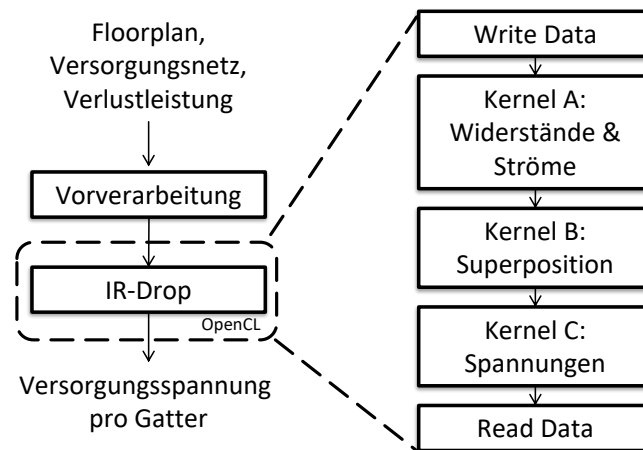


Abbildung 5.19: Ablauf der IR-Drop-Berechnung mit Hilfe von OpenCL. Ausgehend von dem Floorplan, dem Versorgungsnetz und der Verlustleistung wird der IR-Drop berechnet. In einem Vorverarbeitungsschritt werden die Daten für die Berechnung aufbereitet, hierbei werden unter anderem aus dem Aufbau des Versorgungsnetzes die Widerstände berechnet. Die benötigten Daten werden dann an das OpenCL Compute Device übertragen. Anschließend werden drei Kernels gestartet. Kernel A berechnet die Widerstände und Ströme, wie dies in Abbildung 4.20 im Kapitel *Konzepte und Modelle* erläutert wurde. Kernel B führt die Superposition durch und Kernel C berechnet den IR-Drop. Anschließend werden die Ergebnisse an den Host übertragen.

anschließend die Kernels zur Berechnung gestartet. Bei der Entwicklung des Kernels hat es sich herausgestellt, dass es für die Performance technisch sinnvoller ist, die Berechnung, statt nur mit einem Kernel durchzuführen, auf drei aufzuteilen. Der erste Kernel berechnet für die Teilschaltungen zuerst die Ersatzwiderstände und anschließend die Ströme, wie es in dem Kapitel *Konzepte und Modelle* in den Gleichungen 4.23, 4.24 und 4.25 angegeben ist. Hierbei kann der Kernel bei  $N$  Gattern auf insgesamt  $N$  Workitems parallel ausgeführt werden. Der nächste Kernel führt die Superposition aus. Er berechnet aus den Teilströmen den Gesamtstrom durch jeden der Leitungswiderstände. Hierbei berechnet der Kernel, die Ströme für  $V_{DD}$  als auch  $V_{SS}$ , wodurch der Kernel auf  $N + 1^6$  Workitems ausgeführt wird. Der letzte Kernel berechnet die Spannung, mit der die einzelnen Gatter versorgt werden. Dies wurde mit Gleichung 4.27 und 4.28 beschrieben. Die Ausführung wird hier wieder auf  $N$  Workitems aufgeteilt. Anschließend können die Ergebnisse aus dem Compute Device zurück an die Hostanwendung übertragen werden, dort werden dann die Ergebnisse weiterverarbeitet.

Der Ansatz wurde an dieser Stelle auf der Gatterebene erläutert, es ist aber auch möglich, die Gatterstrukturen zusammenzufassen und den Ansatz auf der Ebene der Komponenten zu nutzen. Hierdurch wird zwar keine exakte Simulation pro Gatter durchgeführt, es wird aber ein Mittelwert des IR-Drops gebildet, der für eine Abschätzung auf Komponentenebene geeigneter ist.

Im Kapitel *Evaluation* werden die Ergebnisse mit SPICE-Simulationen verglichen und die

<sup>6</sup> $N + 1$  da es bei dem Versorgungsnetz  $2 * N + 2$  Widerstände gibt und im Kernel die Ströme durch zwei Widerstände berechnet werden.

Performance der Kernels wird auf unterschiedlichen Compute Devices getestet. Besonders durch die gute Aufteilung auf viele Workitems kann eine gute Beschleunigung durch Grafikkarten(GPU) erwartet werden. Im nächsten Abschnitt wird auf die Kopplung zwischen der thermischen Verlustleistung und IR-Drop-Simulation eingegangen.

## 5.7 Elektrothermische Kopplung

Wie im Kapitel *Konzepte und Modelle* dargestellt wurde, bildet die elektrothermische Kopplung den Zusammenhang zwischen dem thermischen Verhalten und der Verlustleistung. Zusätzlich können andere Effekte, wie z.B. der IR-Drop noch Wirkung auf die Kopplung haben. In den vorherigen Abschnitten wurden die Implementierungen zu der thermischen Simulation, die Verlustleistungsberechnung und der IR-Drop vorgestellt. Im Pseudo Code 5.1 wird nun dargestellt, wie die einzelnen Modelle zusammen berechnet werden. Hierzu wird für jeden neuen Zustandswechsel aus dem definierten Missionsszenario zuerst die neue Verlustleistung und dann mit Hilfe der Verlustleistung der IR-Drop berechnet. Diese Berechnung wird solange wiederholt, bis sie nahezu konvergiert. Dann wird mit Hilfe der konvergierten Verlustleistung eine thermische Simulation durchgeführt. Da das Ergebnis der Berechnungen über einen längeren Zeitraum gehen kann, wird an dieser Stelle das Ergebnis z.B. nur zu bestimmten Zeitschritten oder bei definierten Änderungen (Stress Event) der Temperatur oder der Versorgungsspannung durchgeführt. Das hierbei gespeicherte Ergebnis ist das bereits definierte Stressszenario. Diese Schritte werden nun für die thermische Simulation so lange wiederholt, bis zeitlich der nächste Zustand kommt. Hierbei hat die thermische Simulation eine vorgegebene interne zeitliche Auflösung (`internal_time_step`). Dieser Sachverhalt spiegelt das gewünschte Verhalten eines zu simulierenden Systems wider, dass sich die Kopplung der Verlustleistung/IR-Drop deutlich schneller ändert als die Verlustleistung/Temperatur. Die Temperatur hat also eine Trägheit gegenüber der Verlustleistung und benötigt deutlich länger für eine Anpassung der Temperatur. Die Verlustleistungs- und IR-Drop-Berechnung findet nur bei Änderung der Temperatur oder bei einem Wechsel in einen neuen Zustand statt. Im nächsten Abschnitt wird die Umsetzung der Delay-Berechnung vorgestellt.

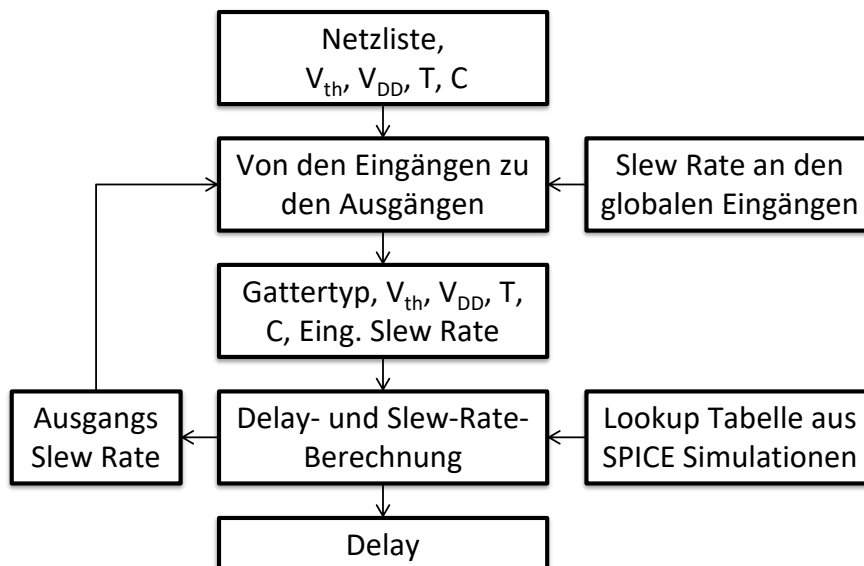
```
foreach (state in mission_scenario) {
  calc_power()
  while (!duration_reached) {
    if (temp_changed or state_changed) {
      while (!power_conv) {
        calc_ir_drop()
        calc_power()
      }
    }
    calc_temperature()
    if (stress_event) {
      store_stress_scenario()
    }
    internal_time_step()
  }
}
```

**Auflistung 5.1: Pseudo Code der elektrothermischen Kopplung.** Für jeden Zustand wird zu Beginn die Verlustleistung als Startwert bestimmt. Anschließend werden abwechselnd der IR-Drop und die neu entstandene Verlustleistung berechnet, bis das Ergebnis der Verlustleistung konvergiert ist. Mit dieser wird dann eine thermische Simulation durchgeführt. Die Ergebnisse der Simulationen, d.h. die Temperatur und die Versorgungsspannung können entweder zu festen Zeitpunkten oder bei bestimmten Ereignissen (`stress_event`) gespeichert werden, wie z.B.: Die Temperatur hat sich um ein Kelvin verändert. Die Berechnung wird solange wiederholt, bis der Zeitraum des simulierten Zustandes zu Ende ist und in den nächste Zustand gewechselt wird.

## 5.8 Delay-Berechnung

### 5.8.1 Umsetzung

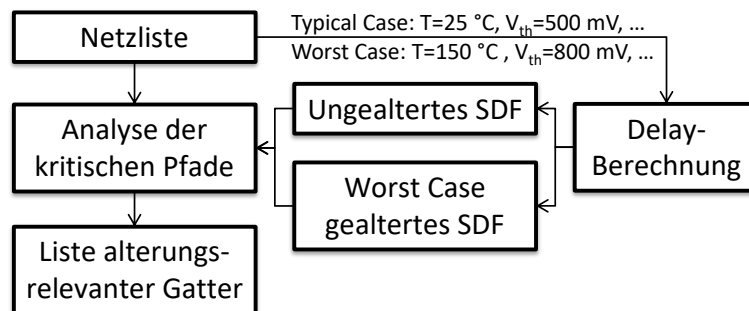
Im Kapitel *Konzepte und Modelle* wurde bereits herausgearbeitet, dass die relevanten Parameter, von dem das Delay abhängt, der Gattertyp, die Schwellspannung  $V_{th}$ , die Versorgungsspannung  $V_{DD}$ , die Temperatur  $T$ , die Lastkapazität  $C$  und die Slew Rate (Flankensteilheit) ist. Ausgehend von diesen Parametern, wird aus mehreren SPICE-Simulationen eine Lookup-Tabelle für die unterschiedlichen Gatter und den Parameter erstellt. Aus dieser Lookup-Tabelle kann dann zur Berechnung neuer Delay-Werte eine Interpolation aus den Werten aus der Lookup-Tabelle durchgeführt werden. Bei diesem Vorgehen wird auch die Slew Rate an den Ausgängen der Gatter berechnet, die sich durch die Alterung ebenfalls verändert. Da das Delay eines Gatters von der Eingangs Slew Rate abhängt, muss bei der Berechnung der gealterten Delays die Ausgangs Slew Rate des vorherigen Gatters bekannt sein. D.h. die gealterten Delays müssen von den globalen Eingängen zu den Ausgängen einer Schaltung berechnet werden, um jeweils die benötigte Slew Rate des vorherigen Gatters zu erhalten. In Abbildung 5.20 ist der Ablauf der Berechnung der Delays und Slew Rates dargestellt. Im Anhang B.0.5 ist ein Beispiel einer Netzliste im XML-Format dargestellt, die mit Hilfe von eigenen TCL Skripten in dem Synthesetool erzeugt wurde. In dieser XML-Datei werden nach der Erstellung im Synthesetool die Informationen aus der Alterungssimulation eingetragen und anschließend wird diese Datei zur Delay-Berechnung genutzt.



**Abbildung 5.20:** Berechnung neuer Delay-Werte. Die Berechnung beginnt bei den globalen Eingängen und errechnet für jedes Folgegatter das neue Delay und die Slew Rate. Die Slew Rate wird dann bei den Folgegattern als Eingangs Slew Rate genutzt. Das Ergebnis der Berechnung ist das Delay für jedes Gatter, das später als SDF ausgegeben werden kann.

### 5.8.2 Kritische Pfade

Da die Anzahl der Gatter, die bei der Alterungssimulation betrachtet werden muss, eine direkte Auswirkung auf die Dauer der Alterungssimulation hat, ist es von besonderem Interesse, die Anzahl der möglichen Gatter, die betrachtet werden müssen, gering zu halten. Hierzu wurde bereits im Kapitel *Grundlagen* ein Verfahren zur Reduktion der zu betrachtenden Gatter vorgestellt. Aus diesem Ansatz ist ein Tool entstanden, das im Anhang B.0.8 dargestellt ist. Es liest die Netzliste des zu reduzierenden Designs ein und baut daraus den Timing Graph auf. Hierzu wird eine ungealterte SDF-Datei und eine durch Worst-Case-Alterung erzeugte SDF benötigt. Das Vorgehen ist in Abbildung 5.21 dargestellt.



**Abbildung 5.21:** Analyse der kritischen Pfade, um die Anzahl der Gatter, die bei der Alterungssimulation betrachtet werden müssen zu reduzieren. Ausgehend aus einer Netzliste wird eine ungealterte SDF-Datei und eine mit Worst-Case-Annahmen gealterte SDF-Datei erzeugt. Mit diesen zwei Timing-Dateien kann eine Analyse der kritischen Pfade durchgeführt werden. Das hierbei genutzte Verfahren wurde im Kapitel *Grundlagen* vorgestellt.

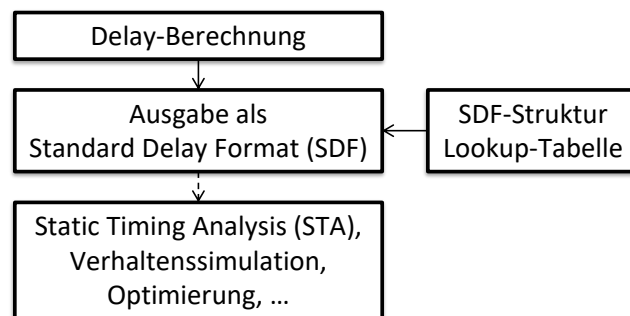
Diese beiden SDF-Dateien können direkt mit dem vorgestellten Weg zum Berechnen der Delays aus dem vorherigen Unterkapitel und dem Erzeugen von SDF-Dateien im nächsten Kapitel erstellt werden. Das Ergebnis der Analyse ist eine reduzierte Liste von Gattern, die auf möglichen kritischen Pfaden liegen und daher für die Alterung relevant sind.

## 5.9 Designflow

### 5.9.1 Aus- und Weitergabe der Ergebnisse – SDF

Ausgehend von der Delay-Berechnung werden in die SDF-Datei die Delay-Werte für jedes Gatter mit den passenden Eingangs- und Ausgangspins eingetragen. In Abbildung 5.22 ist der Ablauf dargestellt. Für die Erstellung einer SDF-Datei wird der genaue Aufbau der Datei benötigt, der in einer Lookup-Tabelle für die unterschiedlichen Gatter abgelegt ist.

Wie dies bereits im Kapitel *Konzepte und Modelle* zu den SDF-Dateien erwähnt wurde, muss eine SDF-Datei nicht alle Gatter einer Schaltung enthalten, sondern nur die, bei der das Timing angepasst werden soll. Bei einer Static Timing Analysis (STA) bedeutet dies, dass nur die Gatter, die auf einem kritischen Pfad liegen, in der gealterten SDF-Datei stehen. Beim Laden der SDF-Datei im STA-Tool werden die internen Delay-Werte in der Datenbank durch die neuen aus der SDF-Datei überschrieben. Wichtig an dieser Stelle ist, dass die Instanz-Namen mit den Gattern aus der Netzliste übereinstimmen und eindeutig sind, da über die Instanz-Namen die Zuweisung in die Industriertools durchgeführt wird.



**Abbildung 5.22:** Ausgabe der Delay-Werte als eine SDF-Datei. Ausgehend aus den berechneten Delay-Werten kann eine SDF-Datei erzeugt werden. Die SDF-Datei beinhaltet die Delay-Werte von den Eingangspins zu den Ausgangspins jedes einzelnen Gatters. Für die Erstellung des SDF wird eine Lookup-Tabelle benötigt, in der die von den Industriertools erwartete Struktur einer SDF-Datei für jeden Gattertyp eingetragen ist.

### 5.9.2 Angepasster Industrieflow/Gesamtflow

In Abbildung 5.23 ist der Designflow dargestellt. Ausgehend von einem typischen Industrieflow wird mit dem Missionsszenario-Editor ein Szenario für die Alterung erstellt. Aus der thermischen Simulation, der Verlustleistung und dem IR-Drop wird das Stressszenario simuliert. Für die thermische Simulation wird ein Package benötigt, das vorher mit dem Package-Editor erstellt werden kann. Das Stressszenario ist dann mit der Netzliste, bzw. den kritischen Pfaden, Ausgangspunkt für die NBTI-Simulation. Anschließend wird eine Delay-Berechnung aus den Ergebnissen aus der NBTI-Simulation durchgeführt. Die Delay-Werte können in einem letzten Schritt als SDF-Datei ausgegeben werden. Die SDF-Datei kann dann für die Weiterverarbeitung in den Industriertools genutzt werden.

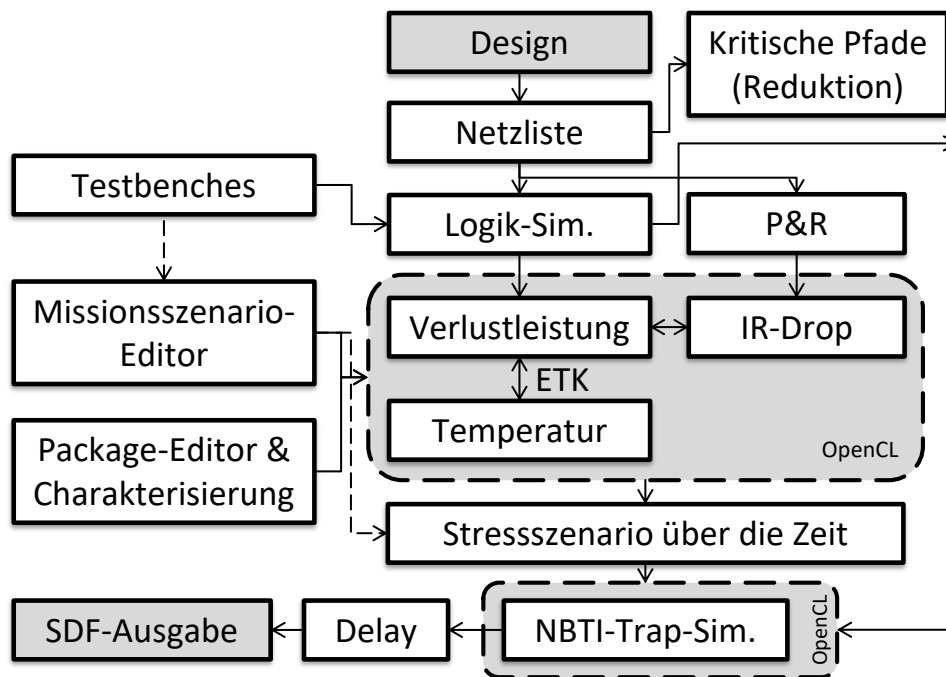


Abbildung 5.23: Überarbeiteter Designflow aus Abbildung 4.33. Im Vergleich zu dem bereits vorgestellten Flow ist hier zusätzlich der Package-Editor und der Missionsszenario-Editor eingetragen.

### 5.9.3 Konzept zur Toolausführung

Im Kapitel *Konzepte und Modelle* wurde ein Konzept vorgestellt, wie die Ausführung von unterschiedlichen Programmen mit Hilfe von Petri-Netzen durchgeführt werden kann. Dies hat den Hintergrund, dass in dieser Arbeit viele kleine Programme durch die vorgestellten Modelle entstanden sind und diese in einem Gesamtflow ausgeführt werden sollen. Hierbei liegt ein Augenmerk auf dem Austausch der Daten, der in den meisten Fällen mit Hilfe von XML-Dateien durchgeführt wird. Dies ermöglicht eine einfache Erweiterung der Dateinhalte, aber auch eine Erweiterung der Programme durch neue oder andere Alterungseffekte, Optimierungen, andere Industrietools, usw.

Zusätzlich haben aber auch die Industrietools eine Restriktion, so dass sie nur auf bestimmten Rechnern mit Lizenzen laufen können. Hinzu kommt hier noch, dass es bestimmte Anforderungen an die OpenCL Compute Devices gibt, die in einem Rechner vorhanden sein muss, wie z.B. die Unterstützung von Berechnungen mit doppelter Genauigkeit.

Aus diesen Anforderungen ist das Konzept, das in Abbildung 5.24 dargestellt ist, entstanden.

Der in dieser Arbeit entstandene Editor ist im Anhang B.0.10 dargestellt. Er erlaubt es, Ablaufpläne von Programmen zu erzeugen und diese lokal auszuführen. Das Konzept zur Server-seitigen Ausführung ist eine Erweiterung des Editors, es befindet sich in einem frühen Entwicklungsstadium. Eine besondere Schwierigkeit an diesem Client/Server Konzept ist es, u.a. auch für eine ausreichende Sicherheit bei der Kommunikation und bei der Ausführung von fremden Programmen zu sorgen. Derzeit besteht der Server-seitige Teil nur aus einem Prototyp,

der als Consolen-Anwendungen ohne grafische Benutzeroberfläche ausgeführt wird.

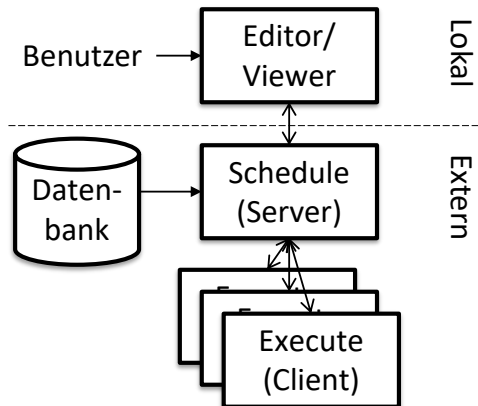


Abbildung 5.24: Vorstellung der Umsetzung des Petri-Netz-Konzepts, bei dem unterschiedliche Programme mit vorgegebener Reihenfolge und unter Berücksichtigung von Abhängigkeiten ausgeführt werden sollen. Der Benutzer gibt in einen Editor die Ausführungsreihenfolge, die Verbindung der einzelnen Tools und die Eingabe-/Ausgabedaten der Programme an. Sind alle Programme lokal vorhanden, kann die Ausführung lokal auf dem Rechner ausgeführt werden. Die Erweiterung des Konzeptes ist eine Server-seitige Ausführung. Hierzu ist ein zentraler Server nötig, der das Scheduling übernimmt und die Programmausführung auf die passenden Clients verteilt. Der Server hat außerdem eine Datenbank, in der Informationen zu den einzelnen Clients gespeichert sind, z.B. welcher Client bestimmte Lizenzen hat oder auch welche Hardware/Performance vorhanden ist.



## 5.10 Zusammenfassung

In dem Kapitel *Implementierung* werden die im Kapitel *Konzepte und Modelle* vorgestellten Modelle umgesetzt. Zu Beginn gibt es eine Einführung in die verwendete Programmiersprache OpenCL, insbesondere konzentriert sie sich auf die für diese Arbeit relevanten Kernelemente. OpenCL ist insofern eine besondere Programmiersprache, da sie Berechnungen massiv parallel auf unterschiedlicher Hardware ausführen kann. Die Umsetzung des NBTI-Modells wird anhand von OpenCL durchgeführt, auf diesem Modell liegt auch der Schwerpunkt der Arbeit. Dabei wird zum einen auf die Erzeugung der NBTI-Technologie-Daten, die zur Simulation benötigt werden, eingegangen. Zum anderen wird der Ablauf der NBTI-Simulation genauer analysiert. Ein zentraler Punkt der Umsetzung des NBTI-Modells ist, die Anzahl der Traps auf ein Minimum zu reduzieren und ihre Berechnung parallel mit Hilfe von OpenCL auszuführen. Durch dieses Vorgehen der parallelen Ausführung der Traps und der Reduktion wird eine massive Beschleunigung erreicht. Neben der Umsetzung des NBTI-Modells geht es auch um das umgesetzte Modell der thermischen Simulation und des Package-Modells. Da für das thermische Modell eine genaue Charakterisierung des Packages nötig ist, wird zuerst der Ablauf der Entstehung eines Packages und die Schritte, die zur Package-Charakterisierung benötigt werden, beschrieben. Die Charakterisierung erfolgt mit dem thermischen Tool HotSpot. Anschließend wird die thermische Simulation erläutert, die mit OpenCL umgesetzt wird. Auch hier wird, wie beim NBTI-Modell, darauf geachtet, die zu verarbeiteten Daten und die Berechnung parallel auszuführen, um die Vorteile von OpenCL auszunutzen. Der nächste Schritt besteht in der Gewinnung der Verlustleistung mit industriellen Tools. Die Verlustleistung wird zur thermischen Simulation und zur Berechnung des IR-Drops benötigt. Das IR-Drop-Modell wird wie die NBTI- und thermische Simulation mit OpenCL umgesetzt. Eine weitere wichtige Umsetzung ist die elektrothermische Kopplung, hierbei wird aufgezeigt, wie das Missionsszenario mit dem thermischen Modell, der Verlustleistung und dem IR-Drop in ein Stressszenario umgewandelt wird. Weiterhin wird die Berechnung der Delay-Werte über eine Interpolation aus Lookup-Tabellen behandelt, sowie der Ansatz zur Reduktion der kritischen Pfade. Bei der Umsetzung der Designflows wird auf die SDF-Ausgabe eingegangen, auf die Anpassung des Industrieflows durch Erweiterung der in dieser Arbeit entstandenen Modelle und es wird aufgezeigt, wie die Umsetzung zu der Ausführung und Überwachung von Programmen mit Hilfe von Petri-Netzen arbeitet. Im folgenden Kapitel *Evaluation* werden die Umsetzungen der *Konzepte und Modelle* mit Hilfe anderer, vergleichbarer Modelle sowie gleichzeitig die Annahmen des Kapitels *Konzepte und Modelle* in ihren einzelnen Schritten überprüft.



In diesem Kapitel werden die in dieser Arbeit entwickelten Konzepte und Modelle überprüft. Hierzu wird zuerst die Modellbewertung vorgestellt, in dieser werden die Fehlerbewertung und die genutzte Evaluationsplattform erläutert. Bei der Evaluation werden zuerst die einzelnen Modelle NBTL, die thermische Simulation, IR-Drop und die kritischen Pfade betrachtet. Anschließend werden komplexere Beispiele mit Ergebnissen vorgestellt.

## 6.1 Modellbewertung

### 6.1.1 Fehlerbewertung

Da bei der Evaluation die Ergebnisse zwischen verschiedenen Methoden, die Auswirkungen auf die Ergebnisse durch Änderungen der simulierten Szenarien haben und z.B. die Nutzung unterschiedlicher Herstellungstechnologien untersucht werden, wird in dieser Arbeit der relative Fehler zwischen dem ursprünglichen und dem neuen Ergebnis betrachtet. Da die Aussagekraft des absoluten Fehlers eher gering ist, wird nur an einigen Stellen ein Beispiel der absoluten Zahlen gegeben. Daraus ergeben sich zur Bewertung der Evaluation in den Unterkapiteln folgende Fehlerbeurteilungen: Der relative Fehler<sup>1</sup> ist das Verhältnis aus der Abweichung vom idealen Ergebnis. Er ist wie in Gleichung 6.1 definiert. In der Gleichung ist  $\Delta x$  der absolute Fehler von dem Sollwert  $x$  dargestellt.

$$\delta x = \frac{\Delta x}{x} = \frac{x_0 - x}{x} = \frac{x_0}{x} - 1 \quad (6.1)$$

---

<sup>1</sup>Taschenbuch der Mathematik, Bronstein[6], (16.210)

Zusätzlich werden der relative Maximal- und Minimalerfehler<sup>2</sup>, wie er in Gleichung 6.2 und 6.3 angegeben ist, genutzt.

$$\delta x_{max} = \max(\delta x_1, \dots, \delta x_n) \quad (6.2)$$

$$\delta x_{min} = \min(\delta x_1, \dots, \delta x_n) \quad (6.3)$$

Eine weitere hierzu passende Angabe ist der Mittelwert<sup>3</sup> aus den relativen Fehlern und die relative Standardabweichung<sup>4</sup> des relativen Fehlers, wie sie in den Gleichungen 6.4 und 6.5 gegeben sind.

$$\bar{x}_{rel} = \frac{1}{n} \sum_{i=1}^n \delta x_i \quad (6.4)$$

$$\sigma_{rel} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (\delta x_i - \bar{x}_{rel})^2} \quad (6.5)$$

### 6.1.2 Evaluationsplattform

Zur Evaluation der Performance der vorgestellten Modelle, die in OpenCL umgesetzt wurden, werden in dieser Arbeit drei verschiedene OpenCL Compute Devices getestet, die in Tabelle 6.1 dargestellt sind. In der Tabelle werden für eine Intel CPU und zwei GPUs von AMD und NVIDIA die Anzahl der Processing Units und die Performance für einfache und doppelte Genauigkeit aufgeführt. Die unterschiedlichen Devices werden in folgenden Evaluationen durch ihren Herstellernamen abgekürzt, wobei hier immer das aufgeführte Modell in der Tabelle gemeint ist. Die Messungen der Ausführungszeiten der einzelnen Devices, die in den Evaluationsteilen durchgeführt werden, beziehen sich immer auf die Übertragung der Daten zum OpenCL Device, die Durchführung der Berechnung (Ausführen der OpenCL Kernels) und dem Übertragen der Ergebnisse. Auf die Besonderheiten zu OpenCL wurde im Kapitel zur Implementierung hingewiesen.

Name	Intel i5-3570K	AMD RX 480	NVIDIA Titan Black
Type	CPU	GPU	GPU
Compute Units	4	36	15
Processing Units	-	2304	2880
einfache Genauigkeit [GFlops] <sup>5</sup>	30	5161	5120
doppelter Genauigkeit [GFlops] <sup>5</sup>	15	323	1700

**Tabelle 6.1:** Dargestellt sind die in dieser Arbeit genutzten OpenCL Devices. Es werden eine 4 Kern CPU und zwei GPUs von unterschiedlichen Firmen genutzt.

<sup>2</sup>Taschenbuch der Mathematik, Bronstein[6], (16.212)

<sup>3</sup>Taschenbuch der Mathematik, Bronstein[6], (16.120)

<sup>4</sup>Taschenbuch der Mathematik, Bronstein[6], (16.121)

<sup>5</sup>Gemessen wurden die OpenCL Performance mit dem Benchmark tool „Flops“. Hierbei wird ein Kernel ausgeführt, der nur aus 32bit/64bit MAD (Multiply and Add) Befehlen besteht. D.h. die Zahlen geben an, wie viele MAD Befehle pro Sekunde ausgeführt werden können (GFlops= 10<sup>9</sup> floating point operations per

### 6.1.3 Technologiedaten zur Evaluation

Für die Evaluation wurden die NBTI-Daten von dem Forschungsprojekt MoRV[61] genutzt. Diese Trapdaten bestehen aus TCAD-Simulation[67] für 16 nm und 45 nm Transistoren. Die Transistor-Daten basieren auf den SPICE-Modellen von Predictive Technology Model (PTM)[72], die für die Delay-Berechnung genutzt werden. Zusätzlich zu den Transistoren wird noch eine Gatterbibliothek benötigt. Dies ist die 45 nm NanGate Open Cell Library[56]. Da für die 16 nm Transistoren keine passende Gatterbibliothek zur Verfügung stand, wurden aus den 16 nm PTM-Transistoren in Anlehnung an die vorhandenen 45 nm NanGate-Gattern eigene 16 nm Gatter gebaut. Da der Aufwand eine komplette Gatterbibliothek aufzubauen zu zeitaufwendig ist, beschränkt sich die Evaluation auf die Nutzung der drei Gattertypen NOR und NAND mit jeweils zwei Eingängen und einem Inverter (INV). Die Wahl dieser drei Gattertypen ist darauf zurückzuführen, dass das genutzte Synthesetool als Voraussetzung mindestens den Gattertyp Inverter und einen weiteren verknüpfenden Gattertyp verlangt, obwohl aus mathematischer Sicht für eine Boolesche Algebra die Operation NOR oder NAND ausreichen würden<sup>6</sup>. Um die Gesamtzahl der Gatter weiter zu senken und um auch für die Alterung zusätzlich beide PMOS-Transistor-Verschaltungen (parallel/seriell) zu haben, wurde die Gattertypen auf NOR, NAND und INV festgelegt.

### 6.1.4 Genutzte Tools in der Evaluation

Für die Evaluation werden zum Erstellen des Designs die Industrietools Synopsys Design Compile[71] für die Synthese und Cadence Encounter[7] für das Floorplaning genutzt. Neben diesen Tools wird auch das Open Source SPICE Tool NGSPICE[57] und HSPICE[7] von Synopsys genutzt. Zur Evaluation der thermischen Simulation wird HotSpot[75] verwendet.

### 6.1.5 Genutzte Designs in der Evaluation

Zur Überprüfung des industriellen Einsatzes der entwickelten Modelle und des Toolflows werden in der Evaluation unterschiedliche digitale Schaltungen eingesetzt. Hierzu werden die häufig von wissenschaftlichen Veröffentlichungen verwendeten Schaltungen aus dem ISCAS-Benchmark[23] genutzt, um die Auswirkungen der Alterung auf die kritischen Pfade zu untersuchen. Weiterhin wird für die verschiedenen Optimierungsmöglichkeiten die Alterung eines 32bit-Addierers betrachtet. Für die Überprüfung, ob auch größere Designs mit den vorgestellten Modellen und den Industrietools durchgeführt werden können, wird ein großes Industriedesign simuliert. Dabei handelt es sich um einen LEON 3 32bit-Prozessor von Aeroflex Gaisler[1].

---

second). Die Ergebnisse geben einen Eindruck, wie viele MAD Operationen im Idealfall ausgeführt werden können. Diese Leistung ist aber nicht auf die Performance von echten Kernels übertragbar, da hier nicht nur die MAD Performance eine Rolle spielt, sondern auch z.B. die Speicherperformance. Weitere Daten im Anhang C.

<sup>6</sup>Taschenbuch der Mathematik, Bronstein[6], Boolesche Algebren und Schaltalgebra (Definition 5.7.1)

## 6.2 Überprüfung der einzelnen Modelle

### 6.2.1 NBTI-Modell

In diesem Kapitel wird das in dieser Arbeit vorgestellte NBTI-Modell evaluiert. Hierzu dienen die im vorherigen Abschnitt vorgestellten Gatter Daten sowie die bereits erwähnten Trapdaten zu den Transistoren der Gatter.

#### Short-term und long-term Unterteilung

Im Kapitel zu den NBTI-Modellen wurde im Abschnitt „Short- und long-term“ die Unterteilung der NBTI-Traps in zwei Klassen vorgestellt. Die erste Klasse sind die short-term Traps, die sehr schnell auf die Änderung der Stresszustände reagieren. Die zweite Klasse sind long-term Traps, die deutlich länger benötigen, bis sich der Einfluss eines Stresszustand bemerkbar macht. Der Ablauf der Unterteilung der Traps wurde genauer in der Abbildung 4.9 vorgestellt. Da die short-term Traps schnell reagieren, müssen diese nicht über den gesamten Zeitraum mitberücksichtigt werden, sondern es reicht aus, dass diese erst zum Ende der Simulation mit den long-term Traps simuliert werden. Die Überprüfung der Unterteilung der Traps und den damit verursachten Fehlern soll in diesem Abschnitt untersucht werden.

Für die Evaluation wurden zufällige Stressszenarien mit unterschiedlichen Laufzeiten erzeugt. Anschließend wurden die Traps für diese Szenarien in short- und long-term unterteilt und mit dem gegebenen Stressszenario eine Alterung durchgeführt. Die Grenze der Unterteilung der Traps wurde auf 1% der Gesamtlaufzeit (Simulationszeitraum der Alterung) festgelegt, d.h. die Traps, die innerhalb des kurzen Zeitraums einen Einfluss auf den Schwellspannungsschaden haben, werden als short-term Traps klassifiziert. In Tabelle 6.2 sind die Ergebnisse der Simulation abgebildet. Die Ergebnisse der Unterteilung, bei dem die long-term Traps über den gesamten Zeitraum und die short-term Traps nur für 1% der Zeit simuliert werden, werden mit den Ergebnissen verglichen, bei dem alle Traps gleich lange simuliert wurden. Bei jedem der Alterungszeiträumen wurden jeweils 100 unterschiedliche Stressszenarien genutzt, womit dann die Fehlerbewertung, die am Anfang des Kapitels vorgestellt wurde, angewendet wurde. Außerdem ist in der Tabelle angegeben, wie viele der Traps als short-term Traps klassifiziert wurden. Bei den Ergebnissen ist zu erkennen, dass die Unterteilung der Traps und das verkürzte Simulieren der short-term Traps zu einem sehr geringen Fehler führt. Dabei reicht die Anzahl der short-term Traps von 34% bis 51% der gesamten Traps. Aus diesem Ergebnis kann geschlossen werden, dass die Annahmen, die zur Unterteilung geführt haben, korrekt sind. Aus der Unterteilung ergibt sich ein erheblicher Geschwindigkeitszuwachs, da die zu simulierenden Traps auf die long-term Traps beschränkt werden können. Hierbei können der Rechenaufwand durch die Zusammenfassung der Logiksignale auf die Signalwahrscheinlichkeiten beschleunigt werden. In diesen Beispielen war die Beschleunigung der Simulation mehr als 30 mal so schnell als eine Simulation bei der die Traps nicht unterteilt wurden. Zum einen können die short-term Traps wie in diesem Beispiel am Ende der Lauzeit der long-term Traps simuliert werden. Zum anderen ist es sinnvoll bei den

short-term Traps den Worst Case anzunehmen, d.h. das alle short-term Traps ihren maximalen Schwellspannungsschaden verursachen. Dies spiegelt den Sachverhalt wieder, das den Designer eines Systems interessiert, wie das System nach zehn Jahren im Worst-Case-Fall sich durch Alterung verhält und nicht, wie der exakte Zustand jedes einzelnen Traps nach zehn Jahren aussieht, der sich aber nach 30 Sekunden durch die short-term Traps wieder ändern kann. Das Auslassen der short-term Trap-Simulation beschleunigt die Berechnung noch einmal deutlich, was in den weiteren Evaluationsteilen noch gezeigt wird.

Alterungszeitraum	Short-term Traps	$\delta x_{max}$	$\delta x_{min}$	$\bar{x}_{rel}$	$\sigma_{rel}$
1 Woche	51%	-0.02%	-0.13%	-0.07%	0.02%
1 Monat	44%	0.14%	-0.20%	-0.07%	0.08%
1 Jahr	39%	0.43%	-0.34%	0.02%	0.26%
10 Jahre	34%	0.57%	-0.38%	0.04%	0.32%

**Tabelle 6.2:** Dargestellt sind die Evaluationsergebnisse von unterschiedlichen NBTI-Trap-Simulationen, bei denen die der Traps in short- und long-term Traps unterteilt sind. Für jede der unterschiedlichen Alterungszeiträume wurden 100 Stressszenarien mit unterschiedlichen Signalwahrscheinlichkeiten durchgeführt. Die Stressszenarien wurden für eine Woche, einen Monat, ein Jahr und zehn Jahre durchgeführt. Die Anteile der short-term Traps, die in der zweiten Spalte dargestellt sind, sinken zum Vergleich mit den long-term Traps bei längeren Simulationszeiten, da die Gesamtzahl der betrachteten Traps bei einer längeren Simulation zunehmen. Diese Zunahme der long-Term Traps wird in Tabelle 6.4 im nächsten Abschnitt erläutert.

### Reduktion der Traps

In diesem Teil wird die Reduktion der Traps untersucht. Im Kapitel *Konzepte und Modelle* wurde im Abschnitt „Reduktion der Traps“ 4.1.2 vorgestellt, dass die NBTI-Traps deutlich reduziert werden können, da ein großer Teil der Traps aufgrund von zu geringer Stressspannung oder der Temperatur bei bestimmten Szenarien keinen Einfluss auf die Schwellspannung haben. Das Vorgehen wurde in Abbildung 4.10 erläutert. Um die nicht benötigten Traps löschen zu können, muss eine Worst-Case-Simulation bei der maximalen Temperatur und Stressspannung, die in dem gewählten Stressszenario vorkommt, durchgeführt werden. Diese Simulation dauert weniger als 60 Sekunden und kann die Anzahl der Traps für die eigentlichen Stressszenarien deutlich reduzieren. In Tabelle 6.3 sind Ergebnisse von Worst-Case-Simulationen für unterschiedliche Temperaturen und Stressspannungen bei den genutzten 16 nm und 45 nm Technologien dargestellt.

Wie zu erkennen ist, kann bei einem Stressszenario mit einer maximalen Temperatur von 350 K und einer maximalen Stressspannung von 0.8 V 68% bei 16 nm und 81% bei 45 nm der Traps gelöscht werden. Bei höheren maximalen Temperaturen und Stressspannungen haben mehr Traps eine Auswirkung auf die Schwellspannung, dies führt zu einer Reduktion der Traps um 23% und 40% bei 450 K und 1.2 V. Bei der Auswahl der Traps kann frei gewählt werden, welcher Fehler durch die nicht mehr berücksichtigten Traps noch akzeptabel ist. Je nach Fehlergrenze können dann mehr Traps bei dem Stressszenario vernachlässigt werden.

In Tabelle 6.4 wurden noch zusätzlich die Auswirkungen des Alterungszeitraums auf die An-

Temperatur	Stressspannung					
	0.8V		1V		1.2V	
	16 nm	45 nm	16 nm	45 nm	16 nm	45 nm
350 K	67%	81%	62%	78%	52%	72%
400 K	54%	72%	44%	64%	32%	54%
450 K	38%	56%	28%	47%	21%	38%

**Tabelle 6.3: Reduktion der NBTI-Traps auf nur noch relevante, die einen Einfluss auf die Schwellspannung haben. In der Tabelle sind die Traps angegeben, die nicht mehr für ein bestimmtes Stressszenario, das über zehn Jahre läuft, berücksichtigt werden müssen. D.h. z.B. für ein Stressszenario bei dem maximal 1 V und 400 K bei 45 nm vorkommen, müssen nur noch  $100\% - 64\% = 36\%$  der ursprünglichen Traps simuliert werden. Hierzu wurde eine Worst-Case-NBTI-Simulation für unterschiedliche Stressspannungen sowie unterschiedliche Temperaturen für die 16 nm und 45 nm Technologie durchgeführt. Anschließend werden bei den Ergebnissen der Simulation die Traps aussortiert, die einen sehr geringen Beitrag zu dem Schwellspannungsschaden haben. Die Grenze, ab denen ein Trap aussortiert wird, kann je nach gewünschter Genauigkeit gewählt werden. Bei dem abgebildeten Ergebnissen wurde die Grenze so gewählt, dass der Fehler durch die reduzierten Traps stets kleiner als 1% ist.**

zahl der reduzierten Traps evaluiert. Hierbei ist es so, dass bei längerer Laufzeit mehr Traps Auswirkungen auf die Schwellspannung haben und so nicht mehr vernachlässigt werden können. So können bei einem Stressszenario bei 1 V, 400 K maximal Temperatur und einer Laufzeit von einer Woche 46% der Traps bei der Simulation nicht betrachtet werden. Bei einer längeren Laufzeit von zehn Jahren sind es nur noch 28% der Traps, die keine Auswirkungen haben. D.h. über eine längere Laufzeit werden mehr Traps aktiviert, die bei einer kürzeren Laufzeit noch keine Rolle spielen. Die Anzahl der long-term Traps nimmt bei längeren Laufzeiten zu. Diese Zunahme der long-term Traps wurde schon in der Erklärung der Tabelle 6.2 angesprochen.

Temperatur	Alterungszeitraum							
	1 Woche		1 Monat		1 Jahr		10 Jahre	
	16 nm	45 nm	16 nm	45 nm	16 nm	45 nm	16 nm	45 nm
350 K	71%	84%	69%	83%	65%	80%	62%	78%
400 K	62%	78%	58%	76%	51%	70%	44%	64%
450 K	46%	66%	42%	61%	34%	53%	28%	47%

**Tabelle 6.4: Reduktion der NBTI-Traps. In der Tabelle sind die Traps, die keinen oder nur noch einen geringen Einfluss auf ein Szenario haben, abhängig von der Temperatur und der Laufzeit, abgebildet. Bei diesen Simulationswerten wurde die Stressspannung auf 1 V festgelegt. D.h. die zehn Jahresspalte entspricht der mittleren Spalte aus Tabelle 6.3.**

### Reduktion der Stresszustände

In diesem Kapitel soll überprüft werden, welche Auswirkungen die Anzahl der Zustände des Stressszenarios auf die Alterungssimulation haben. Im Kapitel zur Implementierung wurde im Abschnitt zur elektrothermischen Kopplung bzw. im Pseudo Code 5.1 schon darauf hingewiesen, dass nicht für jede Temperaturänderung ein zusätzlicher Zustand erstellt werden muss, sondern es erst ab einer bestimmten Änderung der Temperatur sinnvoll ist. Um die Möglichkeit der



Reduktion der Zustände zu überprüfen, wurden 1000 zufällige Stressszenarien erstellt, die jeweils 100000 Zustände haben. Bei jeder dieser Szenarien wurden anschließend die Zustände zusammengefasst, bei denen sich die Temperatur nicht über einen festgelegten Wert ändert. Zusätzlich werden aber maximal 50 Zustände, die aneinander liegen, zusammengefasst, dies soll verhindern, dass, falls sich die Temperatur über einen langen Zeitraum nur langsam ändert, keine großen Fehler durch das Zusammenfassen entstehen. Bei dem Zusammenfassen der Zustände werden die Parameter der Zustände entsprechend verrechnet, so werden aus zwei unterschiedlichen Signalwahrscheinlichkeiten eine neue berechnet, wobei hierbei die zeitlichen Längen der Zustände addiert werden. Hat z.B. ein Zustand eine Signalwahrscheinlichkeit (Einswahrscheinlichkeit) von 80% bei einer 100 Sekunden Zustandslänge und ein zweiter Zustand 50% bei 200 Sekunden, so hat der neue Zustand eine Signalwahrscheinlichkeit von 60% bei einer Laufzeit von 300 Sekunden.

Die Ergebnisse der Simulation sind in der Tabelle 6.5 dargestellt. Hierbei sind in der ersten Spalte die Temperaturschwellen angegeben bis zu denen die Zustände zusammengefasst werden. D.h. in der dritten Spalte stehen die Ergebnisse für den Fall, dass hintereinander liegende Zustände solange zusammengefasst werden, bis der nächste Zustand eine Temperaturänderung von 2 K zum Vergleich zum ersten Zustand hat oder eine maximale Anzahl an zusammengefassten Zuständen erreicht ist.

Temperaturschwelle	Anzahl der Zustände	$\delta x_{max}$	$\delta x_{min}$	$\bar{x}_{rel}$	$\sigma_{rel}$
0 K	100000	0%	0%	0%	0%
1 K	∅ 32918	0.3%	-0.2%	-0.02%	0.06%
2 K	∅ 13789	-0.01%	-0.6%	-0.1%	0.1%
5 K	∅ 3543	-0.1%	-1.1%	-0.3%	0.2%
10 K	∅ 2109	-0.2%	-2.0%	-0.6%	0.3%

**Tabelle 6.5: Reduktion der Zustände von Stressszenarien.** In der Tabelle sind die Ergebnisse verschieden starker Reduktionen der Zustände von zufällig generierten Stressszenarien. Das Kriterium zur Reduktion ist die Änderung der Temperatur zwischen den Stressszenariozuständen. Zustände des Stressszenarios werden zusammengefasst, bis der nächste Temperaturwert die Temperaturschwelle von x K überschritten hat. Insgesamt wurden pro Temperaturschwelle 1000 zufällige Stressszenarien getestet.

Das Ergebnis zeigt, dass selbst bei einer starken Reduktion der Stresszustände bei einem Schwellwert von 10 K der größte relative Fehler von 2% erreicht wird. Der gemittelte relative Fehler hat einen Wert von 0.6%. Durch die Reduktion der Zustände wird die Alterungssimulation um den Faktor 47 beschleunigt, da nur noch etwa 2100 Zustände anstatt der 100000 Zustände simuliert werden müssen.

Auffällig ist außerdem, dass es durch die Reduktion zu einer Unterschätzung der Alterung kommt. Dies ist an den negativen Fehlern zu erkennen. Es passiert aufgrund der Zusammenfassung der Temperaturen durch eine Durchschnittstemperatur der Zustände, da höhere Temperaturen zu einer stärkeren Alterung führen als niedrigere Temperaturen. Eine alternative Möglichkeit

zur Zusammenfassung der Temperaturen ist, dass anstatt einer Durchschnittstemperatur die maximale Temperatur der Zustände genommen wird. In Tabelle 6.6 sind die Ergebnisse dargestellt, bei der die maximale Temperatur der zusammengefassten Zustände genutzt wird. Dies führt zu einer Überschätzung der Alterung, bei ähnlichen relativen Fehlerwerten verglichen mit Tabelle 6.5. Auch hier zeigt sich, dass mit einem relativen maximalen Fehler von 5.4% eine beachtliche Beschleunigung der Simulation erhalten wird. Eine weitere Lösungsmöglichkeit ist bei der Berechnung zu berücksichtigen, wie die Temperatur in die Alterungsberechnung einfließt: Die Temperatur geht dabei exponentiell in die Alterungsberechnung ein. Hierbei ist das Problem, dass jeder Trap eine andere Temperaturabhängigkeit hat. D.h. es kann nicht eine einzelne neue Temperatur durch das Zusammenfassen berechnet werden, sondern viele für jeden einzelnen Trap. Da ein Transistor aus mehreren Tausend Traps bestehen kann, ist der Aufwand zu hoch.

Temperaturschwelle	Anzahl der Zustände	$\delta x_{max}$	$\delta x_{min}$	$\bar{x}_{rel}$	$\sigma_{rel}$
0 K	100000	0%	0%	0%	0%
1 K	∅ 32918	0.9%	-0.03%	0.4%	0.2%
2 K	∅ 13789	2.5%	-0.3%	1.0%	0.6%
5 K	∅ 3543	5.8%	-0.5%	2.3%	1.4%
10 K	∅ 2109	8.5%	0.5%	3.3%	1.9%

**Tabelle 6.6: Reduktion der Stresszustände durch ein Zusammenfassen von aufeinander folgender Zustände abhängig von einer Temperaturschwelle. In der Tabelle werden die Ergebnisse von 1000 zufälligen Stressszenarien, bei denen die Zustände reduziert wurden, mit den nicht reduzierten Werten verglichen. Im Gegensatz zu Tabelle 6.5 wird bei dem Zusammenfassen der Zustände die maximale Temperatur der Zustände übernommen. Die Ergebnisse zeigen am Mittelwert, dass durch die maximalen Temperaturen beim Zusammenfassen eine Überschätzung des Schwellenspannungsschadens entsteht.**

### Vergleich mit anderen Modellen

Um das in dieser Arbeit entwickelte NBTI-Modell mit anderen Modellen zu vergleichen, werden die Daten aus der Arbeit von Eilers[14] genutzt, in der ebenfalls ein NBTI-Modell entwickelt und ausführlich gegen andere Modelle evaluiert wurde. In Tabelle 6.7 sind die Performance-Werte von unterschiedlichen NBTI-Modellen für eine Simulation von einem PMOS-Transistor aus den Daten von Eilers dargestellt. Die Daten geben an, wie viel Zeit eine Simulation eines Stressszenarios für drei unterschiedliche Modelle benötigt. Die Modelle sind der Phasenraum, das CET- und das Trap-Modell. Da nur Daten für eine Laufzeit von einer Woche bei Eilers vorhanden waren, wurden die Laufzeiten für ein Jahr und zehn Jahre aus den gegebenen Daten extrapoliert. Das Phasenraum-Modell benötigt für eine Alterungssimulation für einen Transistor 28 Minuten bei einer Signalperiode von 10 Hz. Im Vergleich hierzu benötigt das CET-Modell 22 Tage, das Trap-Modell benötigt für das gleiche Szenario 452 Tage Rechenzeit. Dies zeigt, dass sowohl das CET-Modell als auch das Trap-Modell nicht in der Lage sind, in einer annehmbaren Zeit einen Transistor altern zu lassen. Selbst das Phasenraum-Modell benötigt 28 Minuten, wenn man dies auf ein größeres Design umrechnet, das aus mehreren Hundert Transistoren besteht, dauert diese Simulation viel zu lange. Bei 100 Transistoren würde die Berechnung ca. 46 Stunden

brauchen.

Modell	Signalperiode [Hz]	Laufzeit der Alterungssimulation für		
		1 Woche	1 Jahr	10 Jahre
Phasenraum-Modell	1	3,29 s	2,9 min*	29 min*
	10	3,22 s	2,8 min*	28 min*
CET-Modell	1	6,1 min	5,3 h*	53 h*
	10	61,1 min	2,2 T*	22 T*
Trap-Modell	1	13,2 h	28,7 T*	287 T*
	10	20,8 h	45,2 T*	452 T*

**Tabelle 6.7: Geschwindigkeitsvergleich zwischen unterschiedlichen NBTI-Modellen. Quelle der Daten ist eine andere Arbeit[14]. Für das Szenario gilt: eine Woche Stressszenario bei einer konstanten Temperatur 350 K, Versorgungsspannung 0.8 V und Signalwahrscheinlichkeit 85%. Die Signalperiode ist 1 Hz und 10 Hz. Die Werte für ein Jahr und zehn Jahre sind aus den Daten einer Woche extrapoliert und gerundet. s= Sekunde, min=Minute, h=Stunde, T=Tag, \*=extrapoliert.**

Da das in dieser Arbeit entwickelte Vorgehen auf dem Trap-Modell basiert, ist dieses der Vergleichswert zwischen den Angaben von Eilers aus der Tabelle 6.7 und den Werten aus meinen eigenen Berechnungen, die in Tabelle 6.8 dargestellt sind. In dieser Tabelle 6.8 ist das Ergebnis des in dieser Arbeit entwickelten NBTI-Modell abgebildet. Ausgangspunkt ist das in OpenCL umgesetzte Trap-Modell, es zeigt bei beiden Szenarien, 1 Hz und 10 Hz Signalperiode, eine Performance, die abhängig von dem Alterungszeitraum steigt. D.h. für eine Simulation der Signalperiode von einem Hertz über eine Woche müssen insgesamt  $60 * 60 * 24 * 7 = 604800$  Signalperiode simuliert werden. Jede Signalperiode besteht aus einem Low- und High-Pegel, dies führt insgesamt zu einer Anzahl von 1.2 Millionen Simulationsschritten für die Trap-Simulation. Entsprechend liegt die Anzahl der Simulationschritte für 10 Hz für eine Woche bei 12 Millionen. Daraus ergibt sich eine extrapolierte Simulationszeit von 28,7 Stunden für eine Simulation des 10 Hz Beispiels über den Zehnjahreszeitraum. Dies zeigt schon eine erhebliche Beschleunigung zu dem gegebenen Wert von Eilers aus der Tabelle 6.7 mit 452 Tagen. Nur die Implementierung in OpenCL ergibt schon eine Beschleunigung um den Faktor 378. Weiterhin ist das Phasenraum-Modell 62-fach schneller als das Trap-Modell in einer OpenCL-Umsetzung. Wendet man jetzt die Unterteilung in short- und long-term Szenario an, dann beschleunigt sich die Berechnung erheblich. Eine Berechnung eines Transistors auf zehn Jahre dauert nun nur noch drei Sekunden. Dieser Wert ist abhängig von der Unterteilung, wie lange ein long- bzw. short-term Szenario ist. Aus diesen Gründen sind die Berechnungen für eine Woche und ein Jahr kürzer als für zehn Jahre. Der letzte Schritt, um das Modell noch weiter zu beschleunigen, ist, das short-term Szenario nicht mit zu simulieren und an dieser Stelle den Worst Case als short-term Szenario anzunehmen. Es wird also nur das long-term Szenario simuliert und kann erheblich reduziert werden, wie dies in den vorherigen Evaluationsteilen gezeigt wurde. Dies führt zu einer Ausführungszeit, die kleiner ist als 0.01 Sekunden. Dieser Wert hängt stark von dem Szenario ab, d.h. wie häufig können einzelne Zustände des Stressszenarios zusammengefasst werden. Dies wurde schon im vorherigen Abschnitt durch die Reduktion der Stresszustände besprochen. Dieses selbst entwickelte Modell ermöglicht es auch, größere Designs mit mehreren Tausend Transistoren altern zu lassen, was

mit dem anderen Modell nicht möglich ist.

Modell	Signalperiode [Hz]	Laufzeit der Alterungssimulation für		
		1 Woche	1 Jahr	10 Jahre
Trap OpenCL - alle Traps	1	19,5 s	17,1 min	3,1 h
	10	3,3 min	2,9 h	28,7 h*
Trap OpenCL - short/long-term	1	0,3 s	0,3 s	3 s
	10	0,3 s	0,3 s	3 s
Trap OpenCL - long-term & Worst Case short-term	1	< 0.01 s	< 0.01 s	< 0.01 s
	10	< 0.01 s	< 0.01 s	< 0.01 s

**Tabelle 6.8: Geschwindigkeitsvergleich zwischen unterschiedlichen NBTI-Modellen. Für das Szenario gilt: eine Woche Stressszenario bei einer konstanten Temperatur, Versorgungsspannung und Signalwahrscheinlichkeit. Die Signalperiode ist 1 Hz und 10 Hz. Fehler < 0.4%. s=Sekunde, min=Minute, h=Stunde, \*=extrapoliert.**

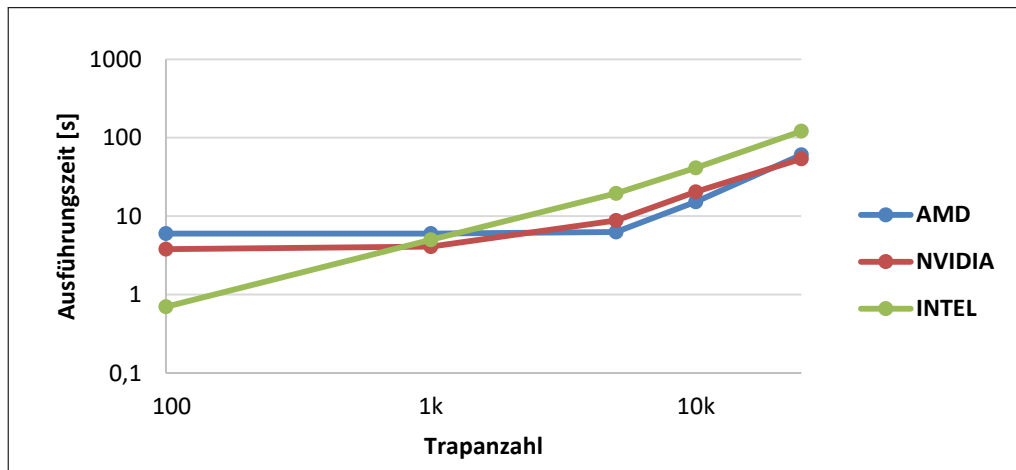
In Tabelle 6.9 sieht man nun die einzelnen Modelle, die ersten beiden Modelle sind aus der Arbeit von Eilers übernommen, während das dritte Modell das eigene darstellt. Als Referenz ist das unoptimierte Trap-Modell zugrunde gelegt. Der Phasenraum zeigt einen maximalen quadratischen Mittelwert von 4.7%. Im Vergleich hierzu zeigt das CET-Modell schon deutlich größere Fehler von 15.3% . Im Vergleich hierzu zeigt das von mir entwickelte optimierte Trap-Modell, bei dem das long-term Szenario über den gesamten Alterungszeitraum simuliert wird und das short-term Szenario nur am Ende, einen deutlich niedrigeren quadratischen Mittelwert von unter 0.5% . Der Vergleich mit den anderen Modellen zeigt, dass die von mir entwickelte optimierte Trap-Simulation einen sehr geringen Fehler aufweist und gleichzeitig zu einer erheblichen Performance des ursprünglichen Trap-Modells führt. Keine der anderen Modelle ermöglicht es, größere Designs in einer annehmbaren Zeit zu simulieren, wo hingegen sie im Vergleich zum Trap-Modell auch noch zusätzlich einen deutlich größeren Fehler aufweisen.

Modell	Zeitraum			Signalperiode		Signalwahr.		
	1T	1W	1M	1Hz	10Hz	10%	50%	90%
Phasenraum [%]	3.8	4.3	4.2	4.0	4.2	4.7	4.4	3.0
CET [%]	5.4	6.7	14.9	8.5	11.2	15.3	5.0	6.0
Trap OpenCL [%] short/long-Term	0.3	0.5	0.2	0.4	0.3	0.4	0.2	0.4

**Tabelle 6.9: Vergleich von unterschiedlichen NBTI-Modellen für verschiedene Stressszenarien.**

### OpenCL-Performance

In diesem Abschnitt wird die Performance der eigenen Methode, die in OpenCL umgesetzt wurde, gemessen. In Abbildung 6.1 ist die Performance der Trap-Simulation für eine Million Stresszustände für unterschiedliche Trapanzahlen im Vergleich zur Ausführungszeit dargestellt.



**Abbildung 6.1: Performance der Trap-Simulation für unterschiedliche Trapanzahlen bei einer Million Stresszustände.**

Für 100 Traps ist die Intel CPU etwa fünfmal schneller als die beiden GPUs. Dies ändert sich ab etwa 1000 Traps, dann sind die GPUs schneller als die CPU. Im Maximum liegen hier die GPUs um den Faktor drei vor der CPU. Zu sehen ist hier zum einen, dass die GPUs eine bestimmte Overhead haben, der besonders bei einer geringen Trapanzahl eine Auswirkung hat und erst ab einer Größe von etwa 5000 steigt die Ausführungszeit mit der Trapanzahl an. Dieses Verhalten ist bei der CPU nicht zu sehen. Zum anderen erreichen die GPUs nur einen geringen Vorteil gegenüber der CPU, obwohl sie zumindest aus der Spezifikation eine deutlich bessere Performance bieten sollten. Die Temperatursimulation zeigt hier deutlich bessere Performance auf der GPU, was im nächsten Abschnitt vorgestellt wird. Grund für das schlechte Abschneiden der GPU ist der Sachverhalt, dass die Daten anwendungsbedingt ungeordnet im Speicher liegen. Dies liegt daran, dass je nach Szenario, abhängig von der aktuellen Temperatur und Spannung, andere Trapdaten aus dem Speicher geladen werden müssen. GPUs haben einen breiten Speicherbus, der zwischen 256 bis zu 512bit breit sein kann, um die nötigen Daten für die Recheneinheiten zu laden. Idealerweise sollten die zu ladenden Daten geordnet im Speicher liegen, um eine bessere Performance mit GPUs erreichen zu können. Dies ist für die Trap-Berechnung leider nicht möglich. Von daher ist, obwohl die Rechenleistung der GPUs höher ist, die CPU fast gleich schnell bei der Berechnung.

In der nächsten Abbildung 6.2 ist die Performance bei unterschiedlicher Anzahl von Stresszuständen gegenüber der Ausführungszeit dargestellt. Diese Messung wurde bei einer Anzahl von 5000 Traps durchgeführt. Hierbei ist die Ausführungszeit linear von der Anzahl der zu simulierenden Stresszuständen abhängig. Die unterschiedlichen OpenCL Compute Devices haben in diesem Fall nur einen geringen Einfluss. 100 Stresszustände benötigen 0.01 Sekunden und bei

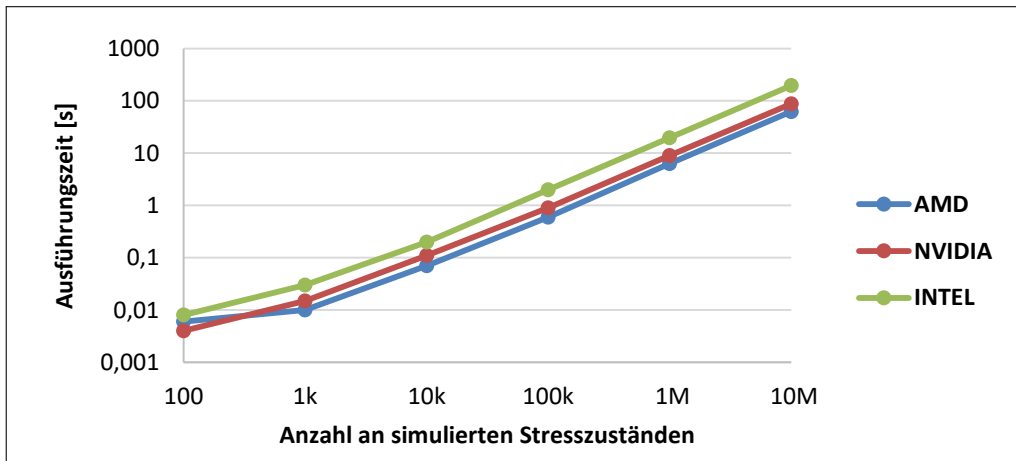


Abbildung 6.2: Performance der Trap-Simulation bei unterschiedlicher Anzahl von Stresszuständen bei einer konstanten Anzahl von 5k Traps.

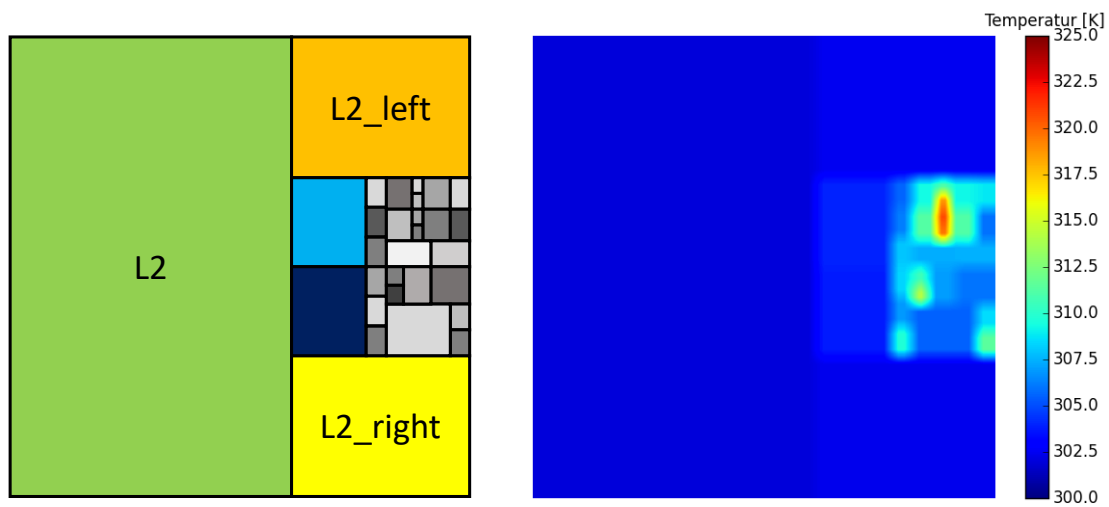
zehn Millionen Zuständen etwa 100 Sekunden.

Insgesamt erreicht die OpenCL-Implementierung eine ausreichende Performance, um Trap-Simulationen auch für komplexe Szenarien durchzuführen. Durch die vorgestellte Unterteilung der Traps und der Zusammenfassung von Zuständen ergibt diese eine erhebliche Beschleunigung, um auch deutlich größere Designs betrachten zu können.

## 6.2.2 Thermisches Modell

### Vergleich mit HotSpot

In diesem Abschnitt wird das entwickelte Modell mit HotSpot verglichen. Hierbei wird, wie bereits zuvor beschrieben, eine thermische Charakterisierung mit Hilfe von HotSpot durchgeführt. Durch diese Charakterisierung kann eine thermische Simulation des thermischen Modells durchgeführt werden. Das genutzte Evaluationsbeispiel ist ein Alpha-EV6-Prozessor, der aus 30 Komponenten besteht[66]. Der Floorplan des Prozessors ist in Abbildung 6.3 auf der linken Seite dargestellt. Auf der rechten Seite der Abbildung ist das Ergebnis zu einem Zeitpunkt einer thermischen Simulation abgebildet, bei der zu diesem Zeitpunkt die Recheneinheiten des Prozessors besonders warm sind, wohingegen die Caches eher eine niedrige Temperatur aufweisen.



**Abbildung 6.3:** Linke Seite: Floorplan des Alpha-EV6-Prozessors mit 30 Komponenten. Rechte Seite: Ergebnis einer thermischen Simulation zu einem Zeitpunkt der transienten Simulation. Das Ergebnis zeigt eine Erwärmung im Bereich der Recheneinheiten des Prozessors.

Mit diesem Prozessor wurde eine 3000 Sekunden lange thermische Simulation mit HotSpot und dem eigenen entwickelten Modell durchgeführt. Für die Simulationsschrittweite der eigenen Methode wurde eine Sekunde gewählt. Der relative Fehler ist in Abbildung 6.4 dargestellt. Die Fehler und die Rechenzeit sind in Tabelle 6.10 eingetragen. Der maximale relative Fehler lag bei 0.61% und der minimale bei -0.52%. Der Mittelwert des relativen Fehlers liegt bei 0.12%. Die Rechenzeit von HotSpot lag bei etwa vier Stunden und die Simulation mit OpenCL bei 0.7 Sekunden. Die thermische Simulation hat einen vernachlässigbaren Fehler und trotzdem eine erhebliche Beschleunigung von mehr als 20000.

Simulationsschritte	Rechenzeit		$\delta x_{max}$	$\delta x_{min}$	$\bar{x}_{rel}$	$\sigma_{rel}$
	HotSpot	OpenCL				
3000	14368 s $\approx$ 4 h	0.7 s	0.61%	-0.52%	0.12%	0.09%

**Tabelle 6.10:** Simulation des Alpha-EV6-Prozessors mit 30 Komponenten.

In Abbildung 6.6 ist die Ausführungszeit im Vergleich zur Komponentenanzahl der OpenCL-

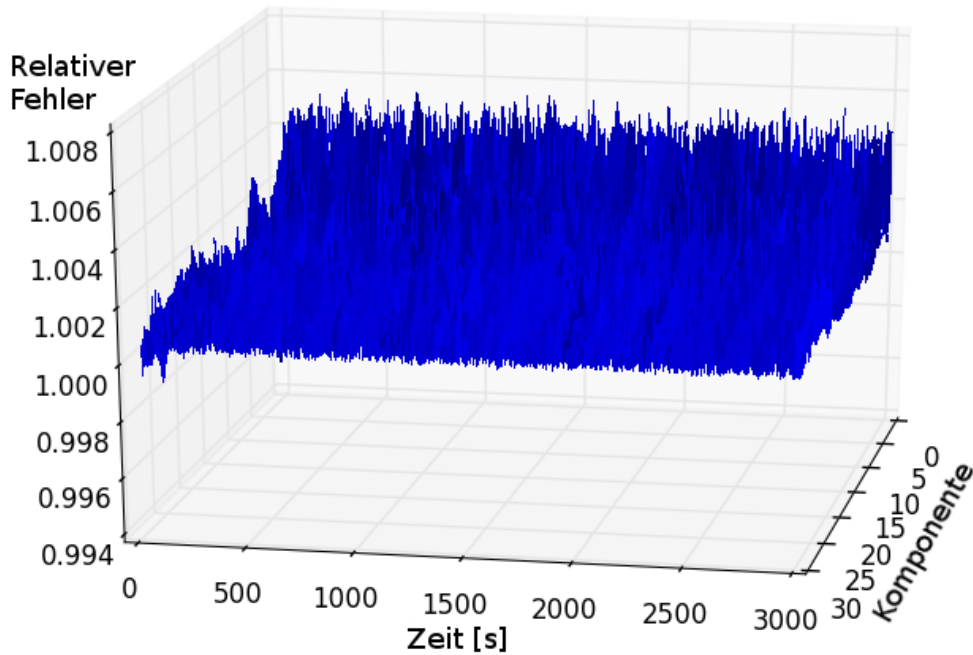


Abbildung 6.4: Relativer Fehler zwischen HotSpot und dem vorgestellten Modell. Die Simulationszeit ist 3000 Sekunden bzw. 50 Minuten. Das thermisch simulierte System besteht aus 30 Komponenten. Der Floorplan des simulierten Systems ist in Abbildung 6.3 dargestellt.

Implementierung bei 10000 Zeitschritten dargestellt. Es zeigt sich, dass die Intel CPU bei einer Anzahl bis zu 128 Komponenten schneller ist als die beiden GPUs. Hier ist die CPU im Maximum etwa dreimal schneller als die GPUs. Beide GPUs liefern eine ähnliche Performance und erreichen im Maximum eine 73-fache Beschleunigung gegenüber der CPU. In diesem Fall ist wie bei der Trap-Simulation zu sehen, dass erst ab einer größeren Anzahl an parallelen Berechnungen (Komponentenanzahl) die GPU schneller ist als die CPU. Im Gegensatz zur Trap-Simulation gibt es bei dieser Implementierung keine Probleme mit dem Speicherzugriff, die GPU zeigt hier deutliche Vorteile bei der parallelen Berechnung.

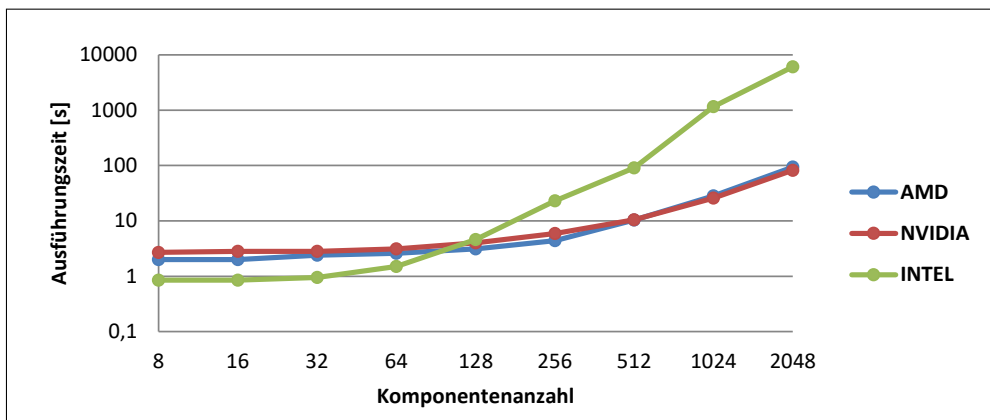


Abbildung 6.5: Performance des thermischen Modells für 10k Zeitschritte.

Zusätzlich wurde noch die Abhängigkeit zwischen der Ausführungszeit und der Anzahl der



Simulationsschritte für 32 Komponenten evaluiert. In Abbildung 6.6 sind die Ergebnisse für die drei OpenCL Devices dargestellt. Alle drei Devices haben eine lineare Abhängigkeit zwischen der Ausführungszeit und der Anzahl der Simulationsschritte. Für zehn Simulationsschritte benötigt die CPU etwa 0.001 Sekunden und für zehn Millionen Schritte 100 Sekunden.

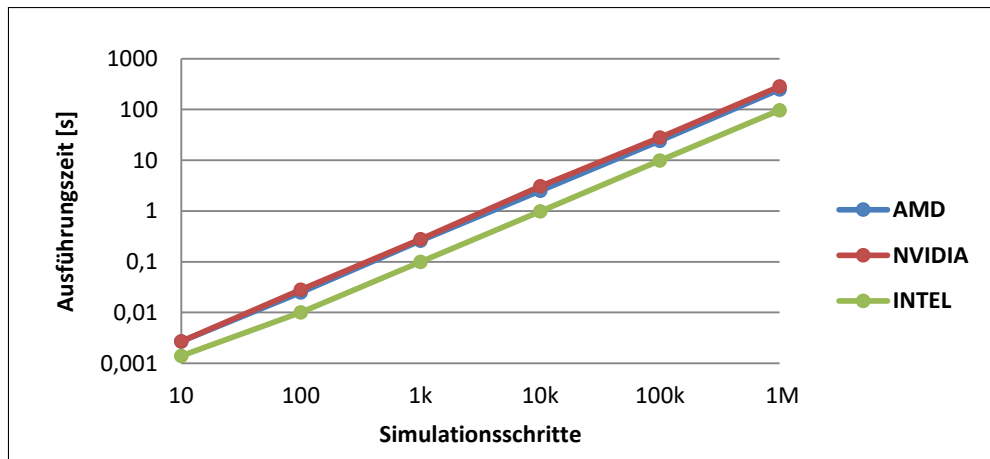


Abbildung 6.6: Performance des thermischen Modells für 32 Komponenten.

### 6.2.3 IR-Drop-Modell

In diesem Evaluationsteil wird das entwickelte IR-Drop-Modell evaluiert. Die erste Evaluation ist die Überprüfung der Berechnungsergebnisse mit SPICE-Simulationen. Es wurden 100 Schaltungen mit jeweils einer Größe von 100k Gattern erstellt. Hierzu wurden typische Werte für die Leitungswiderstände und die Verlustleistung durch Zufallszahlen generiert. Das Ergebnis ist eine so geringe Abweichung zwischen den Werten von SPICE und des Superpositionsansatzes, dass auf eine Angabe des Mittelwerts und der Standardabweichung an dieser Stelle verzichtet wird. Der maximale relative Fehler in den simulierten Schaltungen ist kleiner als  $10^{-5}$  und beschränkt sich auf Genauigkeitsfehler durch die Zahlendarstellung, sowie der eingestellten Toleranzen bei den SPICE-Simulationen.

Die Zeit, die für die Simulationen eines Versorgungsnetzes von SPICE und OpenCL benötigt wurde, ist in Tabelle 6.11 dargestellt.

Name	Performance für 100k Gatter
NGSPICE	24.03 s
GPU AMD	0.16 s

Tabelle 6.11: Geschwindigkeitsvergleich zwischen GPU- und SPICE-Simulation bei 100k Gatter.

In Abbildung 6.7 ist die OpenCL-Performance der drei unterschiedlichen Devices dargestellt. Hierbei sind unterschiedlich große Gatterzahlen und auf der anderen Achse die Laufzeit für einen Berechnungsdurchlauf aufgetragen. Zu sehen ist hierbei, dass die GPUs im Vergleich zur CPU einen bestimmten Overhead haben und erst ab etwa 4000 Gatter die GPUs schneller sind. Bei

einer Anzahl von 100 Gattern ist die CPU 25 mal schneller als die GPUs. Im Gegensatz hierzu erreichten die GPUs ihre Vorteile bei 100000 Gattern mit einer Beschleunigung um Faktor 65 gegenüber der CPU.

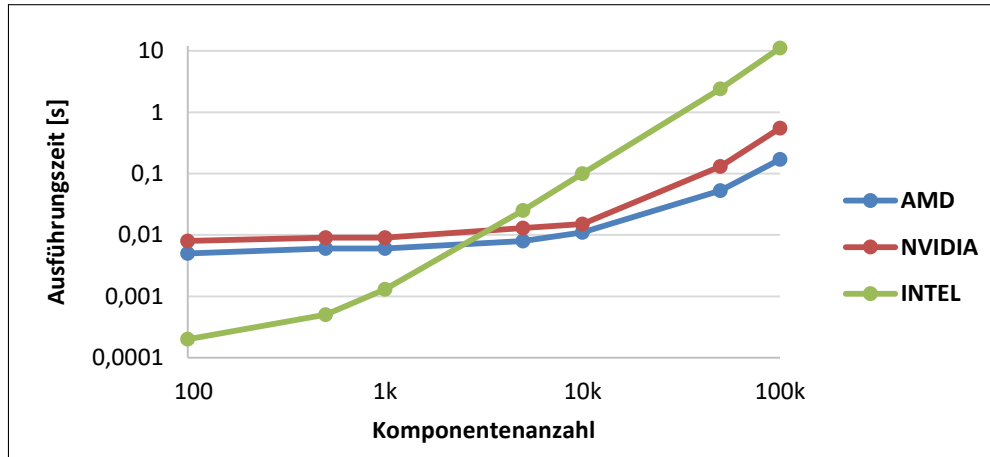


Abbildung 6.7: Performance der IR-Drop-OpenCL-Implementierung.

### IR-Drop-Gatter-Reduktion

Da die Fehler des Modells im Vergleich zu SPICE-Simulationen einen so geringen Wert haben, bietet es sich an, die Gatter für die Simulation zusammenzufassen und so bei einem annehmbaren Fehler die Berechnung zu beschleunigen. Hierzu wurden bereits vorgestellte Evaluationsbeispiele mit den 100k Gattern genutzt, um hieran die Reduktion der Gatter zu evaluieren. Die Ergebnisse sind in Tabelle 6.12 aufgelistet. In der Tabelle sind die Laufzeit und die relativen Fehler im Vergleich zu den SPICE-Simulationen dargestellt. Erst ab einer Größe von 100 Gattern steigt der maximale und minimale relative Fehler auf eine zu beachtende Größe von 2.4% und -2.3%. Dies führt zu einer erheblichen Verringerung des Rechenaufwandes, da die zu berechnende Gatteranzahl erheblich verringert werden kann.

Gatteranzahl	Laufzeit OpenCL (Intel)	$\delta x_{max}$	$\delta x_{min}$	$\bar{x}_{rel}$	$\sigma_{rel}$
100k	11.1 s	0%	0%	0%	0%
50k	2.8 s	0.006%	-0.007%	-0.0001%	0.002%
10k	0.1 s	0.03%	-0.03%	-0.0001%	0.01%
1k	0.002 s	0.26%	-0.26%	0.00001%	0.1%
100	0.0002 s	2.4%	-2.3%	0.01%	1.1%
10	0.0002 s	26.0%	-20.7%	0.9%	10.6%

Tabelle 6.12: IR-Drop-Reduktion.

### 6.2.4 Delay-Modell

In diesem Abschnitt wird das Delay-Modell evaluiert, das in Kapitel 5.8 und 4.7 vorgestellt wurde. Das Delay-Modell berechnet aus fünf Parametern das Delay und die Slew Rate eines Gatters. In dieser Arbeit wird - wie in der Einleitung dargestellt - eine 16 nm und eine 45 nm Technologie genutzt. Um den Aufwand gering zu halten, werden nur die Gatter INV, NOR und NAND genutzt. Aus Lookup-Tabellen werden die Delays und die Slew Rates der Gatter berechnet. Für die drei Gatter gibt es jeweils vier verschiedene Tabellen, d.h. für Slew Rate und Delay in je zwei Technologien. Die Lookup-Tabellen werden zur Interpolation der Delays und der Slew Rates genutzt. In Tabelle 6.13 sind die Parameter und die genutzten Bereiche, in denen die Parameter genutzt werden, dargestellt.

Parameter	Symbol	Bereich
Temperatur	$T$	[-25 °C, 150 °C]
Versorgungsspannung 45 nm	$V_{DD}$	[0.9 V, 1.25 V]
Versorgungsspannung 16 nm	$V_{DD}$	[0.75 V, 0.95 V]
Lastkapazität	$C_{load}$	[2 fF, 100 fF]
Slew Rate	$SR$	[0 ps, 100 ps]
Schwelspannungsschaden	$\Delta V_{th}$	[0 V, 0.3 V]

**Tabelle 6.13: Delay-Parameter**

Zur Überprüfung des Modells wurde für jedes Gatter 100000 zufällige Parameter erstellt. Mit diesen Parametern wurde dann das Delay und die Slew Rate mit SPICE simuliert und eine Berechnung mit Hilfe der Lookup-Tabellen durchgeführt. In Tabelle 6.14 sind die Ergebnisse mit den bekannten Fehlermaßen für die Delay-Berechnung dargestellt. Bei den Ergebnissen ist zu sehen, dass die maximalen Fehler sich im Bereich von 7% bis zu 20% bewegen und der minimale Fehler im Bereich von -2% bis 9%. Der Fehler von 20% ist groß, aber die Delay-Berechnung wird nicht nur für ein Gatter durchgeführt, sondern für ganze Komponenten, die aus mehreren tausend Gattern bestehen. Von daher ist der Mittelwert und auch die Standardabweichung an dieser Stelle ausschlaggebend. Der Mittelwert liegt bei der 16 nm Gattern unter 0.5% und bei den 45 nm unter 2.5%. Die Standardabweichung liegt bei 16 nm bei unter 2.5% und bei 45 nm unter 4.2%. Die Ergebnisse liegen allesamt im annehmbaren Bereich von unter 5%. Auffällig ist an dieser Stelle, dass die Fehler für die 45 nm größer sind als für die 16 nm. Dies ist zum einen auf die Wahl der Werte, die in der Lookup-Tabelle gespeichert sind, zurückzuführen und zum anderen aber auch darauf, dass hier nicht nur zwei verschiedene Strukturgrößen verglichen werden, sondern dass die 16 nm Gatter aus FinFET-Transistoren aufgebaut sind und andere Eigenschaften als die 45 nm Transistoren haben.

Gattertyp	$\delta x_{max}$		$\delta x_{min}$		$\bar{x}_{rel}$		$\sigma_{rel}$	
	16 nm	45 nm	16 nm	45 nm	16 nm	45 nm	16 nm	45 nm
INV	6.8%	13.3%	-8.6%	-3.8%	0.5%	1.2%	2.5%	2.5%
NAND	7.5%	13.5%	-7.7%	-1.7%	0.03%	2.3%	1.0%	2.8%
NOR	11.5%	20.2%	-9.1%	-3.8%	0.03%	2.5%	0.9%	4.2%

**Tabelle 6.14: Delay**

In Tabelle 6.15 sind die Evaluationsergebnisse für die Slew-Rate-Berechnung dargestellt. Auch bei diesen Ergebnissen ist zu erkennen, dass die für 45 nm maximalen Fehler größer sind als die für 16 nm. Hier sind ebenfalls der Mittelwert und die Standardabweichung von größerer Bedeutung. Wie bei den Delay-Werten auch sind die Ergebnisse besonders bei den 16 nm Gatter von unter 0.1% annehmbar.

Gattertyp	$\delta x_{max}$		$\delta x_{min}$		$\bar{x}_{rel}$		$\sigma_{rel}$	
	16 nm	45 nm	16 nm	45 nm	16 nm	45 nm	16 nm	45 nm
INV	7.8%	15.8%	-15.0%	-1.5%	0.01%	1.9%	2.1%	3.5%
NAND	10.1%	15.9%	-15.3%	-0.8%	0.1%	3.1%	1.4%	3.9%
NOR	10.9%	21.1%	-13.4%	-1.0%	0.08%	3.5%	1.3%	5.2%

**Tabelle 6.15: Slew Rate**

Neben der Genauigkeit der Delay-Berechnung ist auch die Dauer der Berechnung wichtig, da eine Komponente aus einer großen Anzahl aus Gattern bestehen kann und ggf. unterschiedliche Szenarien für eine Komponente getestet werden sollen, die zu vielen neuen Berechnungen der Gatter-Delays führen. In Tabelle 6.16 ist die Zeitdauer für die Berechnung von 1000 Gattern angegeben. Da die 45 nm und 16 nm Gatter auf unterschiedlichen Transistor-Modellen basieren, mussten unterschiedliche SPICE-Simulatoren genutzt werden. NGSPICE wurde für die 45 nm Gatter genutzt und da es OpenSource ist, konnte das 45 nm Transistor-Modell sowie die daraus gebauten Gatter so erweitert werden, dass zusätzliche Parameter für die Alterung vorhanden sind (siehe hierzu auch Anhang D). Bei dem 16 nm Gatter musste HSPICE als Simulator genutzt werden, da es hier nicht möglich war, das Transistor-Modell anzupassen. Die Lösung für dieses Problem war, dass für jeden Transistor eine eigene Modell-Datei mit den Alterungsparametern für jede Simulation erzeugt werden musste. Dieser Sachverhalt führt zu dem Ergebnis, dass HSPICE 11 mal länger für die Simulation benötigt als NGSPICE. Die Interpolation aus den Lookup-Tabellen benötigt 8.7 Sekunden für 1000 Gatter und ist etwa 13 mal schneller als NGSPICE. Hinzuweisen ist an dieser Stelle, dass NGSPICE und HSPICE aufgrund ihrer Lizenzen auf einem speziellen Server laufen und die Interpolation hingegen auf einem lokalen Rechner. Die Implementierung der Interpolation wird zurzeit nur auf einem Prozessorkern ausgeführt. Die Berechnung der Gatter kann in einer späteren Erweiterung auf beliebig viele Kerne aufgeteilt werden.

Name	Performance für 1k Gatter
NGSPICE 45 nm <sup>7</sup>	113 s
HSPICE 16 nm <sup>8</sup>	1246 s
Lineare Regression <sup>9</sup>	8.7 s

**Tabelle 6.16: Delay-Berechnung, Performance SPICE vs. Linearer Regression.**

<sup>7</sup>AMD Opteron 6376 1.4 GHz

<sup>8</sup>AMD Opteron 6376 1.4 GHz

<sup>9</sup>Intel 3570K 3.4 GHz

### 6.2.5 Kritische Pfade

In dieser Arbeit wurde an mehreren Stellen darauf hingewiesen, dass es nicht nur einen kritischen Pfad in einer Komponente geben kann, sondern dass sich der kritische Pfad durch die Alterung ändern und so ein anderer Pfad mit anderen Gattern kritisch werden kann. Um zu überprüfen, ob der kritische Pfad sich tatsächlich ändert, wurden zehn unterschiedliche Schaltungen aus dem ISCAS-Benchmark[23] untersucht. Die Namen der Schaltungen und auch die Ergebnisse der Untersuchung sind in Tabelle 6.17 dargestellt. Die Anzahl der Gatter reichen von 7 bis zu 2826 Gatter. In einem ersten Schritt wurden alle Gatter mit einem Worst-Case-Szenario simuliert, d.h alle Gatter der Schaltung altern mit der gleichen maximalen Temperatur, Versorgungsspannung und der maximalen Stressaktivität. Das Ereignis ist in der dritten Spalte dargestellt. Wie zu erwarten war, gibt es für keinen der kritischen Pfade eine Änderung, da alle Gatter gleich stark gealtert sind und die Delays der Gatter sich gleich verändert haben.

Schaltung	Gatter Anzahl	Änderung des kritischen Pfades		
		T,V=max p=max	T,V=max p=var	T,V=var p=var
c17	7	Nein	Nein	Nein
c432	170	Nein	Ja	Ja
c499	558	Nein	Ja	Nein
c880	395	Nein	Ja	Nein
c1355	533	Nein	Ja	Ja
c1908	396	Nein	Nein	Nein
c2670	707	Nein	Ja	Ja
c3540	883	Nein	Nein	Nein
c5315	1670	Nein	Ja	Ja
c6288	2826	Nein	Ja	Ja

**Tabelle 6.17:** Veränderung durch Alterung der kritischen Pfade von zehn unterschiedlichen Schaltungen aus dem ISCAS-Benchmark. T=Temperatur, V=Versorgungsspannung, p=Signalwahrscheinlichkeit, max=maximum, var=variabel.

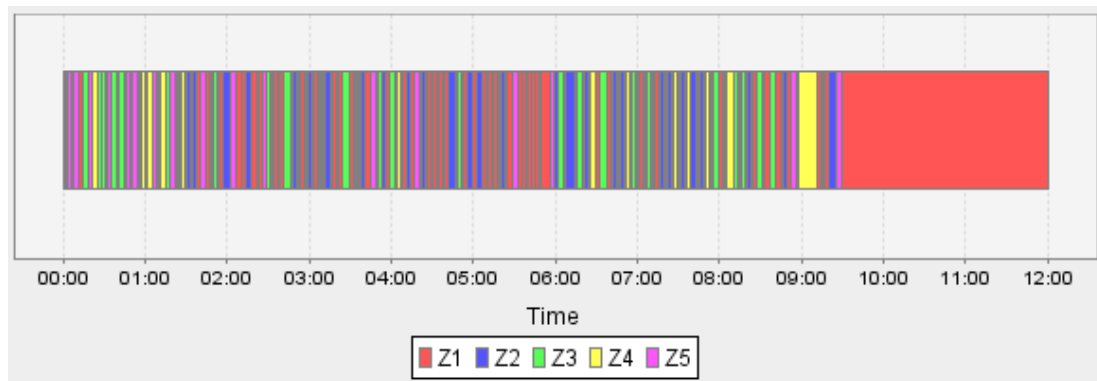
Bei der nächsten Untersuchung wurden bei der Alterungssimulation reale Transistor-Signalwahrscheinlichkeiten genutzt, weiterhin bleibt die maximale Temperatur als auch die maximale Versorgungsspannung unverändert. Das Ergebnis ist in der vierten Spalte dargestellt. Es zeigt sich hier, dass sich die kritischen Pfade teilweise geändert haben. Bei drei Schaltungen, wie z.B. C17, gibt es anscheinend keinen anderen kritischen Pfad, der durch die Alterung der Gatter aktiviert werden kann.

Bei der letzten Untersuchung wurde ein Stressszenario gewählt, in dem die Temperatur, die Versorgungsspannung und die Signalwahrscheinlichkeit der Gatter variabel sind. Hierbei zeigte sich, dass die kritischen Pfade bei den Schaltungen c499 und c880, die sich bei der vorherigen Untersuchung verändert haben, nun aber konstant bleiben. D.h. hier zeigt sich, dass nicht nur die Aktivität eine Rolle spielt, sondern auch die Temperatur und die Versorgungsspannung.

Als Ergebnis lässt sich festhalten, dass sich die kritischen Pfade ändern können, dies muss allerdings nicht zwangsläufig passieren. Für eine Alterungssimulation reicht es nicht aus, nur den kritischen Pfad eines ungealterten Designs zu betrachten und nur diesen altern zu lassen. Außerdem haben sich durch eine Worst-Case-Alterung die kritischen Pfade nicht verändert. Erst durch die Berücksichtigung der Signalwahrscheinlichkeit passiert eine Änderung der Pfade. Aber auch durch eine Alterungssimulation mit Stressszenarien, bei denen sowohl die Temperatur, die Versorgungsspannung und die Signalwahrscheinlichkeit sich über die Zeit ändern, muss es nicht zu einer Änderung des Pfades kommen.

### 6.3 Stressszenario und Optimierung

In diesem Teil soll die Alterung und die Optimierung eines 32bit-Addierers betrachtet werden. Um ein realitätsnahes Missionsszenario zu erzeugen wurde mit einem PC-Diagnoseprogramm<sup>10</sup> Auslastungsdaten auf einem PC aufgezeichnet. Diese Daten, wie die CPU-Auslastung und die Taktfrequenz über die Zeit, wurden genutzt, um ein Missionsszenario zu erstellen. In Abbildung 6.8 ist das erstellte Missionsszenario, bestehend aus fünf unterschiedlichen Auslastungszuständen, für eine Zeitdauer von 12 Stunden zu sehen.

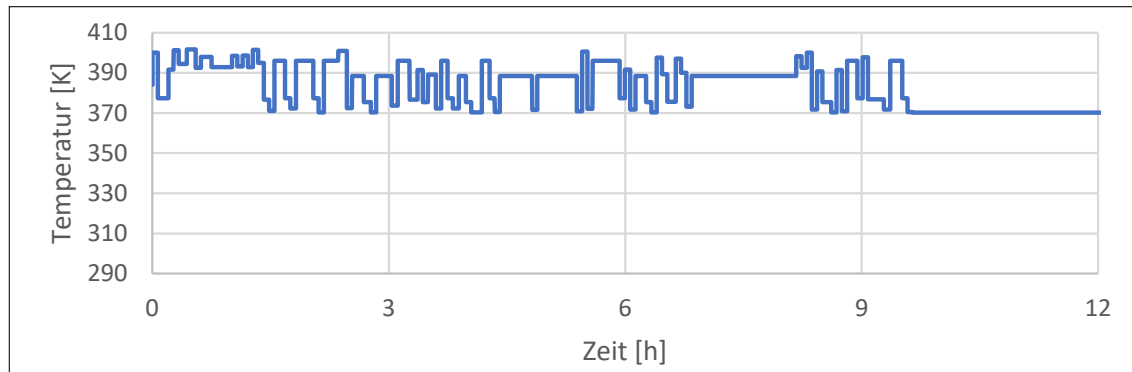


**Abbildung 6.8:** Missionsszenario, aus dem die Stressszenarien erstellt werden. Das Szenario besteht aus fünf unterschiedlichen Zuständen, die den Addierer unterschiedlich stark auslasten.

Ausgehend von diesem Missionsszenario wurde mit Hilfe der in dieser Arbeit entwickelten Methoden Stressszenarien erstellt, mit denen die Alterung des Addierers simuliert wird. In Abbildung 6.9 ist der Temperaturverlauf für den einfachen Fall, d.h. ohne die Nutzung von „Dynamic Voltage and Frequency Scaling“ oder Power Gating, dargestellt. Da in diesem Szenario keine Anpassung der Versorgungsspannung oder eine Abschaltung des Addierers durchgeführt wird, bewegt sich die Temperatur im Bereich von 370 bis 400 K. Dieses Stressszenario wird für einen Zeitraum von zehn Jahren für die Alterung des Addierers genutzt.

In Tabelle 6.18 sind die Ergebnisse für den Worst-Case-Alterungsfall und das Stressszenario des Addierers für 16 nm und 45 nm dargestellt. Hierbei zeigt der kritischste Pfad des Addierers

<sup>10</sup>Core Temp: <http://www.alcpu.com/CoreTemp/>. Mit Hilfe des Programmes wurden folgende Daten eines Prozessors erfasst: Zeit [s], CPU-Auslastung [%] und CPU-Taktung [MHz]



**Abbildung 6.9: Temperatur Stressszenario, das aus dem Missionsszenario mit Hilfe der Temperatursimulation erstellt wurde.**

eine Alterung des Delays bei 16 nm von 95% und bei 45 nm von 68%. Diese Worst-Case-Abschätzung würde zu einer Fehleinschätzung der Alterung führen und somit müsste die Auslegung der Taktfrequenz für ein solches System halbiert werden. Im Gegensatz zu dieser Abschätzung führt eine Simulation des Stressszenarios zu einer Alterung des kritischen Pfades von 14% für die 16 nm und zu 10% für 45 nm Gatter. Es ist zum einen zu sehen, dass der Worst Case deutlich von dem eigentlichen Ergebnis des Szenarios abweicht und zum anderen, dass bei 16 nm sowohl bei dem Worst-Case-Fall als auch bei dem Stressszenario die Änderung des Delays größer ist als bei der 45 nm Transistoren.

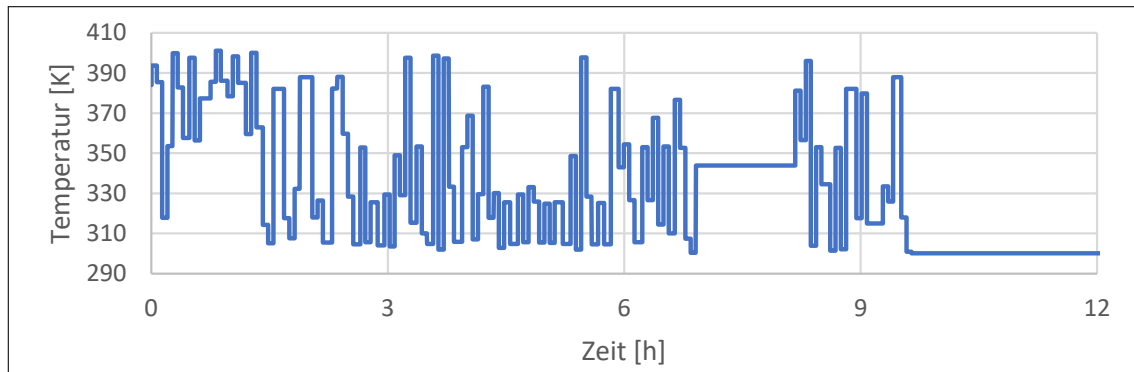
Herstellungstechnologie	Worst Case	Stressszenario
16 nm	94.8%	14.0%
45 nm	67.7%	10.0%

**Tabelle 6.18: Prozentuale Erhöhung des Timings des kritischen Pfades eines 32bit-Addierers unter Worst-Case-Annahmen und realistischem Stressszenario.**

### Dynamic Voltage and Frequency Scaling

Bei der nächsten Betrachtung haben die fünf Zustände des Szenarios unterschiedliche Versorgungsspannungen, um damit die Wirkung von „Dynamic Voltage and Frequency Scaling“ zu untersuchen. In Abbildung 6.10 ist die Temperatur des Stressszenarios dargestellt. Im Vergleich zu dem ersten Stressszenario verändert sich die Temperatur des Addierers deutlich durch die Nutzung von DVFS. Die Temperaturen bewegen sich bei einer starken Auslastung des Systems von 400 K, wohingegen bei einer geringen Auslastung der Addierer eine Temperatur von fast 300 K hat. Dieser Sachverhalt, dass DVFS die Temperatur beeinflusst, aber auch gleichzeitig die Versorgungsspannung verändert, hat einen großen Einfluss auf die Alterungssimulation.

Die Ergebnisse der Alterung durch das Stressszenario sind in Tabelle 6.19 dargestellt. Die Nutzung von DVFS reduziert die Alterung des Addierers bei 16 nm um 38% und bei 45 nm um 42%. DVFS ist also eine gute Möglichkeit, die Alterung eines Systems zu senken, die Auswirkung von DVFS ist abhängig von dem eigentlichen Szenario, d.h. wie häufig ist welcher Zustand aktiv und wie ist die Temperatur des Systems.



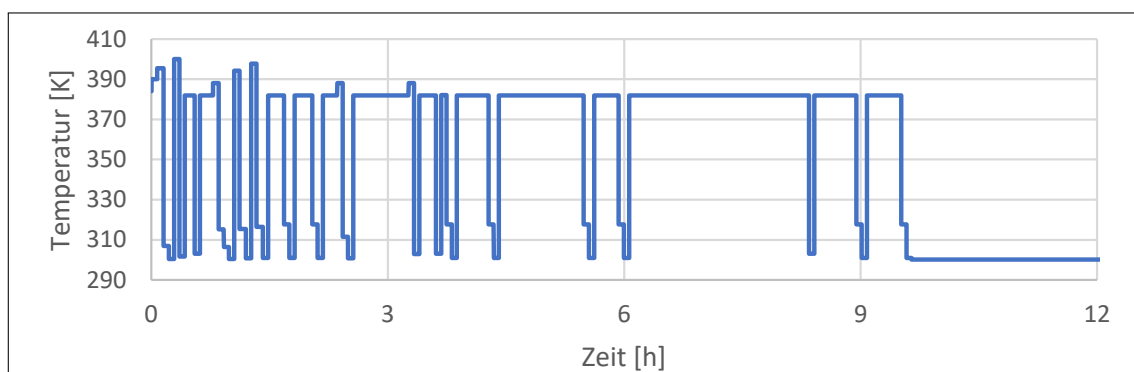
**Abbildung 6.10:** Temperaturverlauf des DVFS-Stressszenarios. Die Temperaturen verändern sich deutlich stärker im Vergleich zum ersten Stressszenario durch die Änderung der Versorgungsspannung.

Herstellungstechnologie	DVFS-Stressszenario	Reduzierung der Alterung im Vergleich zu den Werten aus Tabelle 6.18
16 nm	8.7%	37.9%
45 nm	5.8%	42.4%

**Tabelle 6.19:** Verringerung der Alterung des Addierers durch Anwendung von DVFS auf das Szenario von Tabelle 6.18 .

### Power Gating

Eine weitere Optimierungsmöglichkeit ist Power Gating, dabei werden Komponenten eines Systems, die nicht genutzt werden, ausgeschaltet. Bei diesem Szenario wird bei Zuständen mit einer geringen Auslastung der Addierer ausgeschaltet. In Abbildung 6.11 ist der simulierte Temperaturverlauf dargestellt. Da im Vergleich zum DVFS-Szenario die Versorgungsspannung nicht in verschiedenen Stufen abgesenkt wird, sondern nur zwischen Komponente an oder aus wechselt, hat der Temperaturverlauf Sprünge zwischen 300 K und 380 K.



**Abbildung 6.11:** Temperaturverlauf des Power Gating Stressszenarios. Das Abschalten des Addierers durch Power Gating führt zu deutlichen Temperatursprüngen, da im ausgeschaltetem Zustand das System sich schnell abkühlt.

Dieses Szenario führt zu dem Ergebnis, das in Tabelle 6.20 abgebildet ist. Power Gating führt zu einer Reduktion der Alterung von etwa 35% für den 16 nm und 45 nm Addierer. Es zeigt, ebenso wie DVFS, eine gute Möglichkeit die Alterung zu verringern. Im Vergleich zu



DVFS werden häufiger hohe Temperaturen erreicht, da die Spannung in den unterschiedlichen Zuständen nicht verringert wird. Das Ergebnis ist aber auch hier abhängig von dem Szenario. Die in dieser Arbeit vorgestellten Modelle ermöglichen genau diese Abwägung, wie sinnvoll bestimmte Optimierungsvarianten genutzt werden können und unter welchen Szenarien welche Auswirkungen auf die Alterung eines Systems bestehen.

Herstellungstechnologie	Power Gating Stressszenario	Reduzierung der Alterung im Vergleich zu den Werten aus Tabelle 6.18
16 nm	9.1%	35.1%
45 nm	6.6%	34.6%

**Tabelle 6.20: Auswirkungen von Power Gating auf die Alterung. Verglichen wird an dieser Stelle mit dem Beispiel aus Tabelle 6.18.**

### Pin Swapping

Pin Swapping ist eine Optimierungsmöglichkeit, die bei in Reihe geschalteten PMOS-Transistoren angewendet werden kann. Dieser Sachverhalt wurde im Kapitel Gatter-Modelle erläutert. Durch ein Vertauschen der Eingangspins kann die Alterung des Gatters in Abhängigkeit von den Signalwahrscheinlichkeiten verringert werden. Dieses Vorgehen funktioniert nur bei Gattern, bei der die PMOS-Transistoren in Reihe geschaltet sind, wie z.B. einem NOR-Gatter. Da der kritische Pfad des Addierers aus den obigen Beispielen zum größten Teil nur aus NAND-Gattern besteht, hat diese Optimierung keine Verbesserung für den kritischen Pfad gezeigt. Aus diesem Grund wurde der Addierer zusätzlich nur aus NOR- und INV-Gattern aufgebaut. Anschließend wurde das Design mit dem Stressszenario gealtert und die Gatter mit Pin Swapping auf dem kritischen Pfad optimiert. Anschließend wurde mit dem neuen Design eine Alterung durchgeführt. Die Ergebnisse sind in Tabelle 6.21 dargestellt. Die Optimierung brachte eine Verringerung der Alterung um 22% für den 16 nm und 19% für den 45 nm Addierer. Diese Optimierung hatte eine Verringerung der Alterung erwirkt, ohne in das Szenario einzugreifen. Es hat eindeutig den Nachteil, dass dies nicht für alle Designs zu einem Erfolg führen muss, da diese Optimierung nur für Gatter, bei denen die Transistoren in Reihe geschaltet sind, funktioniert. Außerdem kann die Änderung der Eingangspins Auswirkungen auf den kritischen Pfad haben, was durch eine Timing Analyse überprüft werden muss.

Herstellungstechnologie	Stressszenario	Pin Swapping Stressszenario	Reduzierung
16 nm	27.4%	21.4%	21.7%
45 nm	12.3%	10.0%	18.6%

**Tabelle 6.21: Auswirkung von der Pin Swapping Optimierung auf den kritischen Pfad eines Addierers, der aus NOR- und INV-Gattern aufgebaut ist.**

### Temperatur- und Floorplan-Optimierung

Neben der Versorgungsspannung und der Signalwahrscheinlichkeit spielt die Temperatur für die Alterung eine entscheidende Rolle. D.h. eine weitere Möglichkeit die Alterung zu verringern, besteht darin, die Temperatur des Systems durch eine bessere Kühlung oder eine bessere Platzierung an einer kühleren Stelle der kritischen Komponente innerhalb des Systems zu reduzieren. Um diese zu verdeutlichen, wurde bei dem Stressszenario, bei dem die maximale Temperatur 400 K beträgt, um 10, 25 und 50 K gesenkt. Die Ergebnisse der Alterung sind in Tabelle 6.22 dargestellt. An den Ergebnissen ist deutlich zu sehen, dass sich die Alterung des kritischen Pfades durch eine Senkung der Temperatur von 10 K um 14% bzw. 19% für die 16 nm und 45 nm verringert. Bei einer größeren Absenkung der Temperatur um 50 K verringert sich die Alterung bei dem 45 nm sogar um deutliche 68%. Der 16 nm Addierer zeigt für die gleiche Änderung der Temperatur nur eine Verringerung der Alterung von 44%. Diese Unterschiede der verschiedenen Auswirkungen der Temperatur sind auf die andere Strukturgröße zurückzuführen. Dies zeigt, wofür die in dieser Arbeit vorgestellten Modelle und Umsetzungen eingesetzt werden können: Es können die Alterung von Designs, die in unterschiedlichen Herstellungstechnologien hergestellt werden, verglichen und Optimierungen durchgeführt werden.

Herstellungstechnologie	Reduzierung der Alterung im Vergleich zu den Werten aus Tabelle 6.18		
	-10 K	-25 K	-50 K
16 nm	14.4%	29.3%	43.7%
45 nm	18.5%	43.8%	67.6%

**Tabelle 6.22: Reduzierung der Alterung durch Verringern der Temperatur.**

#### 6.3.1 Großes Industriedesign

Um zu zeigen, dass die in dieser Arbeit vorgestellten Methoden nicht nur auf einfache Registertransferkomponenten wie einen Addierer angewendet werden können, sondern dass die Methoden auch die Alterungssimulation eines kompletten größeren Designs, das aus mehreren Komponenten besteht, leisten kann, wurde ein LEON 3 32bit-Prozessor von Aeroflex Gaisler[1] genutzt. Dieser Prozessor wird im wissenschaftlichen als auch kommerziellen Bereich genutzt. Die Synthese des Prozessors ergab mit den in dieser Arbeit genutzten Industrietools eine Designgröße von 1.2 Millionen Logikgattern. Hierbei wurde der interne Speicher des Prozessors nicht mit einberechnet. Mit diesem Syntheseergebnis wurde eine Alterungssimulation der 1.2 Millionen Gatter für zehn Jahre durchgeführt. Die Rechenzeit dieser Simulation dauerte auf der Intel CPU 38.6 Stunden. Die generierte SDF-Datei mit den gealterten Delays wurde anschließend für eine Static Timing Analysis genutzt. Die SDF des gesamten Designs hat eine Größe von ca. 400 MB und ließ sich mit den Industrietools ohne weiteres verarbeiten. Der kritischste Pfad des gealterten Designs hat eine Länge von 179 Gattern und befindet sich in dem 32bit-Multiplizierers des LEON 3 Prozessors. Die Ergebnisse für das simulierte Stressszenario und den Worst Case sind in Tabelle 6.23 dargestellt. Der Worst Case ist in diesem Fall die höchste Temperatur,

Versorgungsspannung und maximale Schaltaktivität, die in dem Stressszenario vorkommt. Bei den Ergebnissen ist zum einen zu sehen, dass die 16 nm eine stärkere Alterung gegenüber der 45 nm Technologie aufweist. Zum anderen ist zu sehen, dass die Worst-Case-Alterung des Designs eine deutliche Überschätzung des eigentlichen Stressszenarios des Prozessors darstellt und nicht für eine hilfreiche Aussage genutzt werden kann. Neben der Nutzung der SDF-Datei zur Static Timing Analysis kann diese auch zu Timing-Simulationen genutzt werden. Mit Hilfe dieser Simulationen kann z.B. untersucht werden, wie sich Berechnungen auf dem gealterten Prozessor auswirken, hierbei kann z.B. auch eine Optimierung der Software durchgeführt werden. Da bei größeren Designs die Timing-Simulation sehr lange dauern kann, gibt es unterschiedliche Ansätze die Timing-Simulation zu beschleunigen, wie z.B. die Timing-Simulation auf FPGAs mit Hilfe der SDF-Delay-Daten[79].

Herstellungstechnologie	Worst Case	Stressszenario
16 nm	141.2%	19.6%
45 nm	75.8%	13.4%

**Tabelle 6.23: Ergebnisse der Alterungssimulation über zehn Jahre des LEON 3 Prozessors: Der kritischste Pfad hat eine Länge von 179 Gatter und befindet sich im Multiplizierer des Prozessors.**

## 6.4 Zusammenfassung

In diesem Kapitel *Evaluation* werden die in dieser Arbeit implementierten Konzepte und Modelle evaluiert. Hierbei wird der relative Fehler, der maximale und minimale Mittelwert und die Standardabweichung als Fehlermaß genutzt. Um die Ergebnisse dieser Arbeit bewerten zu können, werden drei unterschiedliche OpenCL Devices als Evaluationsplattform ausgewählt. Als Technologiedaten werden 45 und 16 nm Transistoren eingesetzt. Hieraus wird eine Gatterbibliothek mit den Gattern NOR, NAND und INV genutzt. Die Modelle werden im einzelnen überprüft, dabei werden beim NBTI-Modell zuerst die Annahmen zur short- und long-term Unterteilung, danach zur Reduktion der Traps, und zum Schluss der Stresszustände erfolgreich überprüft. Anschließend wird das entwickelte NBTI-Modell mit anderen Modellen verglichen. Hierbei stellt sich heraus, dass das Modell einen sehr geringen Fehler und durch das optimierte NBTI-Modell und die OpenCL-Implementierung eine deutlich höhere Performance hat. Im Anschluss wird die OpenCL-Performance des NBTI-Modells auf der Evaluationsplattform untersucht. Hierbei stellt sich heraus, dass es keine deutlichen Vorteile für die GPU gab. Die GPU ist im Maximum dreimal so schnell wie eine CPU.

Das thermische Modell wird mit den Simulationsergebnissen mit HotSpot verglichen. Dabei zeigt sich, dass die OpenCL-Implementierung in einem Beispiel 20000 mal schneller war als die HotSpot-Simulation bei einem vernachlässigbaren Fehler.

Bei der OpenCL-Performance auf der Evaluationsplattform zeigten die GPU im Maximum eine 73-fache deutliche Beschleunigung gegenüber der CPU.

Das IR-Drop-Modell wird mit einer SPICE-Simulation evaluiert und zeigt einen vernachlässigbaren Fehler. Im Maximum war die OpenCL-Performance mit GPU um das 65-fache schneller als die CPU.

Das Delay-Modell, das auch die Slew-Rate-Berechnung beinhaltet, errechnet sich aus einer Interpolation aus Lookup-Tabellen. Die Ergebnisse aus dieser Interpolation werden zum Überprüfen des Modells mit SPICE-Simulationen verglichen. Dabei zeigt sich, dass der gemittelte Fehler bei 16 nm unter 1% und bei 45 nm unter 4% lag.

Anschließend wird untersucht, wie sich die kritischen Pfade von zehn Schaltungen des ISCAS-Benchmarks durch Alterung verändern können. Hierbei ergibt sich, dass es durch eine Worst-Case-Alterung zu keiner Änderung der kritischen Pfade kommt. Erst eine Alterungssimulation, bei der die Signalwahrscheinlichkeiten berücksichtigt werden, führt zu einer Änderung der kritischen Pfade. In der Untersuchung wird auch deutlich, dass diese kritischen Pfade abhängig vom Szenario, also der Temperatur, Spannung und Aktivität, sind. Dies zeigt, dass man bei Alterungssimulationen nicht nur den ungealterten kritischen Pfad berücksichtigen muss, sondern es deutlich mehr Transistoren sind, die durch die Änderung des kritischen Pfades ebenfalls betrachtet werden müssen.

Im Folgenden werden verschiedene Optimierungen, die die Alterung verringern sollen, an einem 32bit-Addierer durchgeführt: Dies sind „Dynamic Voltage and Frequency Scaling“, Power Gating, Pin Swapping und Temperatur-Floorplan-Optimierung. Der Erfolg der umgesetzten Methodik

---

wird durch die Anwendung des Konzepts auf ein großes Industriedesign unterstrichen. Ein Leon 3-Prozessor mit 1.2 Millionen Gattern wurde über zehn Jahre altern gelassen. Diese Berechnung konnte auf einer Standard-Intel-CPU unter 40 Stunden durchgeführt werden.



---

## Fazit und Ausblick

---

Durch den heutigen technologischen Fortschritt werden Systeme immer komplexer, wodurch es aber auch zu Problemen der Zuverlässigkeit kommen kann. Wie bereits in der *Einleitung* erläutert, betrifft dies vor allem die Alterung. Das Ziel dieser Arbeit ist es, unter Berücksichtigung eines Nutzungsszenarios, eine Alterungssimulation von Systemen durchführen zu können. Da die Alterungseffekte von vielen Einflussfaktoren abhängen, gehört es zu der Zielsetzung der Arbeit, auch geeignete Modelle für die Einflussgrößen zu entwickeln. Durch die komplexen Systeme ist es zudem ein Ziel, dass die entwickelten Modelle in einem industriellen Designflow genutzt und auch Alterungssimulationen von großen Designs durchgeführt werden können.

Für die Alterungssimulation wird ein geeignetes NBTI-Modell entwickelt, das es durch eine parallele Berechnung der NBTI-Traps ermöglicht, eine effektive Alterung zu simulieren. Hierbei wird in der Arbeit OpenCL verwendet. Aufgrund des Alterungseffektes werden geeignete Modelle für die Einflussgrößen entwickelt. Da die Alterungszeiträume in Bereichen von einigen Jahren liegen, müssen die Modelle - wie schon das NBTI-Modell - eine genügend hohe Performance haben, um dies leisten zu können. Bei der Betrachtung von mehreren Schaltungen wird in der Evaluation gezeigt, dass eine Alterung mit Worst-Case-Werten eine deutliche Überschätzung der Alterung ist. Auch bei einer Worst-Case-Alterung ändern sich die kritischen Pfade nicht. Erst bei einer Simulation mit variablen Einflussgrößen, wie dies in der Arbeit durchgeführt wird, gibt es auch einen Einfluss auf die kritischen Pfade.

Um die Alterung zu verringern, hat es sich als sinnvoll erwiesen, die Optimierungen zu nutzen, die die Einflussgrößen verringern, ohne dabei in die Funktionsweise des Systems tiefer gehend einzugreifen. Die getesteten Optimierungen haben dabei unterschiedliche Einflüsse auf das System. Aus den Schwächen, die sich aus dem *Stand der Technik* herausarbeiten lassen und den Fragen aus der *Einleitung* ergeben sich fünf Ziele für die Arbeit:

Es wird ein NBTI-Modell entwickelt, das mit einer hohen Performance verschiedene Nutzungsszenarien simulieren kann und dabei zusätzlich eine hohe Genauigkeit besitzt. Die Evaluation hat gezeigt, dass das Modell deutlich präziser arbeitet als andere Ansätze und gleichzeitig eine deutlich höhere Performance aufweist.

Es gibt drei Einflussgrößen für den simulierten Alterungseffekt NBTI, die als Nutzungsszenario gesehen werden: Diese sind Temperatur, Versorgungsspannung und die Signalaktivität. Es werden in der Arbeit geeignete Modelle für Temperatur und Versorgungsspannung entwickelt und das Vorgehen zur Berechnung der Signalaktivität, um ein Gesamtkonzept zum Nutzungsszenario zu erstellen.

Das genutzte Alterungsmodell basiert auf der Trap-Simulation, d.h. für die Nutzung von anderen Technologien, z.B. Strukturgrößen, müssen nur die Traplisten ausgetauscht werden. Es hat sich in der Literatur gezeigt, dass HCI und NBTI zusammenhängen und sich auch über eine Trap-Simulation simulieren lassen. Damit lässt sich also das vorliegende Konzept auch auf den Alterungseffekt von HCI übertragen und anwenden. Zusätzlich ist es so, dass kleinere Strukturen zu einer kleineren Anzahl von Traps führen[81] und dies beschleunigt wiederum den Ansatz. Derzeit werden ein Transistor pro Simulationsdurchlauf simuliert. So hat sich die Trapanzahl bei kleineren Strukturen immer weiter verringert, die vorliegenden Daten zeigen, dass die Trapanzahl von 130 nm im fünfstelligen Bereich liegen, in dem Bereich von 16 nm im vierstelligen und bei zukünftigen Technologien gehen sie dann nur noch in den dreistelligen Bereich bzw. noch geringer[61]. Dies führt dazu, dass mit dem in der vorliegenden Arbeit entwickelten Ansatz ohne große Anpassung mehrere Transistoren gleichzeitig in einem Durchlauf simuliert werden können. Dieses Vorgehen beschleunigt den Ansatz erheblich.

Der Aufbau des Konzeptes wird auch so gewählt, dass jede Teillösung autark arbeitet. Dies bedeutet, dass Teilprobleme überschaubar und einfach auszutauschen sind. Dies wird auch mit dem Ausführungsmodell zur Überwachung und Ausführung mit Petri-Netzen verwendet. Über XML-Formate können Technologiedaten und Informationen, z.B. zu Nutzungsszenarien, ausgetauscht werden, die einfach nach Bedarf erweitert werden können.

Neben den entstandenen Modellen sind auch erste Prototypen für die Entwicklungssoftware ausgearbeitet worden, die den Ablauf des Gesamtkonzeptes veranschaulichen und es erst ermöglicht haben, die Ergebnisse dieser Arbeit zu liefern. Sie bestätigen die erfolgreiche Umsetzbarkeit des vorliegenden Gesamtkonzeptes. Diese werden im Anhang umfangreich dargestellt.

Es wird aufgezeigt, wie die Ergebnisse der Simulation durch SDF-Dateien in dem Industrieflow genutzt werden können. Außerdem werden in der Arbeit Exportskripte eingesetzt, die mit den Industrietools genutzt werden, so können zur Evaluation industrielle Designs verwendet werden. Sowohl das Temperatur-, als auch das IR-, sowie das NBTI-Modell werden so entwickelt, dass die Berechnungen parallel ausgeführt werden können, um so eine erhebliche Performance bieten zu können, die es ermöglicht hat, große Industriedesigns problemlos zu simulieren. Zu Beginn des Forschungsvorhabens war die Zielsetzung allgemein gehalten, diese wurde in der *Einleitung* allgemein definiert und im Kapitel *Stand der Technik* konnte dieses Forschungsziel durch eine



---

Lücke bei den Alterungssimulationen weiter präzisiert werden und hieraus wurde schließlich das Ziel konkretisiert: eine Simulation eines variablen Nutzungsszenarios für eine vier- bis fünfstellige Anzahl an Gattern zu ermöglichen. Im Kapitel *Evaluation* wurde das Konzept an einem 1.2 Millionen Gatter großen Design erfolgreich getestet. Dieses Ergebnis überschreitet bei weitem das gesteckte Ziel, das aus dem Stand der Technik abgeleitet wurde.

Aufgrund der Erreichung aller oben genannten Ziele ist es mit den Modellen, Implementierungen und Konzepten dieser Arbeit erfolgreich möglich, Alterungsbewertungen mit Missionsszenarien durchzuführen. Dies erlaubt eine detaillierte Überprüfung, welchen Einfluss Optimierungen und Testschaltungen durch die Alterung haben. Außerdem ist es möglich, mit den Ergebnissen eine Verhaltenssimulation durchzuführen, so kann z.B. auch überprüft werden, welchen Einfluss die Alterung auf ausgeführte Software auf einen gealterten Prozessor haben.

In der Arbeit wurden Modelle für die Gatter- bzw. Registertransferebene aufgestellt, es ist sinnvoll, diese weiter zu abstrahieren, um auch Simulationen auf Systemebene durchzuführen. Hierzu wurde bereits ein Konzept mit ersten Ergebnissen veröffentlicht[40][25]. Es ist ebenso sinnvoll weitere Optimierungen durchzuführen, um weitere Erkenntnisse daraus zu gewinnen, z.B. was eine Floorplan-Optimierung bringt. Hier wäre es auch sinnvoll, den Ansatz zu nutzen, um die Robustheit von Software zu bewerten. Ein weiteres Thema, das den Fokus der Arbeit streift, ist das Reagieren auf Alterungseffekte durch aktives Management des Systems.

Für die Entwicklung zukünftiger Systeme wird es sich ähnlich entwickeln, wie dies in der Vergangenheit bei der Verlustleistung geschehen ist. Hier waren Worst-Case-Abschätzungen typische Parameter eines Systems, für die das System ausgelegt wurde. Es hat sich in der Vergangenheit gezeigt, dass z.B. Intel bei ihren Prozessoren die Worst-Case-Verlustleistung nicht mehr angeben, sondern eine typische Verlustleistung für ein realitätsnahes Szenario. Für diese Werte wird z.B. die Kühlleistung des Systems ausgelegt, sollte das System dennoch während des Betriebes eine höhere Leistung aufnehmen als der typische Wert, für den die Kühlleistung ausgelegt ist, so greift an dieser Stelle ein Thermal Management, das in das System aktiv eingreift, um die Verlustleistung zu senken. Das aktive Eingreifen erfolgt, z.B. durch die Reduktion der Taktfrequenz. Übertragen auf die Alterung bedeutet dies, dass Systeme eine Überwachung der Alterung durchführen, z.B. durch Überwachungsschaltungen und sobald diese signalisieren, dass das System in einem kritischen Alterungszustand ist, muss ein aktives Management ähnlich der Verlustleistung z.B. die Taktfrequenz des Systems reduzieren[86]. D.h. es kann bis auf die Testschaltung auf bereits vorhandene Funktionen zurückgegriffen werden, die bereits in heutigen Systemen genutzt werden. Für die Überprüfung solcher Szenarien, die bei Problemen in zukünftigen Technologien hervorgerufen werden, eignen sich die in dieser Arbeit vorgestellten Methoden.



## NBTI-Trap-Modell

### Herleitung

In diesem Teil wird die Gleichung für die Berechnung der Spannung  $V_{trap}$  aus Abbildung A.1 hergeleitet<sup>1</sup>.

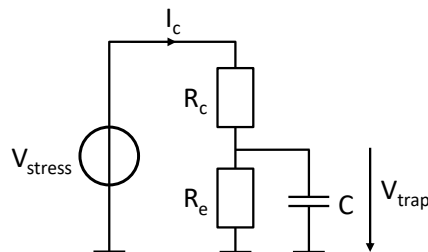


Abbildung A.1: Ersatzschaltung nach [73], die das Verhalten eines NBTI-Traps nachbildet. Die Spannung  $V_{stress}$  lädt und entlädt über die Widerstände den Kondensator. Der Ladezustand des Kondensators ( $V_{trap}$ ) entspricht bei diesem Modell dem Ladezustand des Traps.

Maschengleichung:

$$V_{stress}(t) = I_c(t)R_c + V_{trap}(t) \quad (\text{A.1})$$

Knotengleichung:

$$I_c(t) = \frac{V_{trap}(t)}{R_e} + C \frac{dV_{trap}(t)}{dt} \quad (\text{A.2})$$

Aufstellen der inhomogenen Differentialgleichung durch Einsetzen von A.2 in Gleichung A.1:

$$R_c C \frac{dV_{trap}(t)}{dt} + \left( \frac{R_c}{R_e} + 1 \right) V_{trap}(t) = V_{stress}(t) \quad (\text{A.3})$$

<sup>1</sup>Vorgehen zur Lösung von linearer Differentialgleichungen 1. Ordnung nach Taschenbuch der Elektrotechnik, Kories[41], Kapitel 1.2.5

Anfangsbedingung, der Kondensator ist bereits auf die Spannung  $V_{trap0}$  aufgeladen:

$$V_{trap}(t = 0) = V_{trap0} \quad (\text{A.4})$$

Die Lösung setzt sich aus der homogenen und einer speziellen Lösung zusammen:

$$V_{trap}(t) = V_{trap,homogen}(t) + V_{trap,speziell}(t) \quad (\text{A.5})$$

Lösung des homogenen Teils der DGL (A.3 mit  $V_{stress}(t) = 0$ )

$$\frac{dV_{trap,homogen}(t)}{dt} + \left(\frac{1}{R_e \cdot C} + \frac{1}{R_c \cdot C}\right)V_{trap,homogen}(t) = 0 \quad (\text{A.6})$$

Lösung einer DGL erster Ordnung lautet:

$$V_{trap,homogen}(t) = k \cdot e^{\alpha t} \quad (\text{A.7})$$

Mit dem Einsetzen von A.7 in A.6 ergibt sich:

$$k\alpha \cdot e^{\alpha t} + \left(\frac{1}{R_e \cdot C} + \frac{1}{R_c \cdot C}\right)k \cdot e^{\alpha t} = 0 \quad (\text{A.8})$$

Hierbei ist Folgendes eine mögliche Lösung der Gleichung A.8 :

$$\alpha = -\left(\frac{1}{R_e \cdot C} + \frac{1}{R_c \cdot C}\right) \quad (\text{A.9})$$

Dies ergibt als Lösung der homogenen DGL A.9 in A.7 :

$$V_{trap,homogen}(t) = k \cdot e^{-\left(\frac{1}{R_e \cdot C} + \frac{1}{R_c \cdot C}\right)t} \quad (\text{A.10})$$

Eine spezielle Lösung der DGL ist  $V_{trap}(t \rightarrow \infty)$ :

$$V_{trap,speziell}(t) = \left(\frac{R_c}{R_e} + 1\right) \cdot V_{stress} = \frac{R_e}{R_c + R_e} \cdot V_{stress} \quad (\text{A.11})$$

Die gesamte Lösung der inhomogenen DGL lautet( A.10 + A.11):

$$V_{trap}(t) = V_{trap,homogen}(t) + V_{trap,speziell}(t) = k \cdot e^{-\left(\frac{1}{R_e \cdot C} + \frac{1}{R_c \cdot C}\right)t} + V_{stress} \cdot \frac{R_e}{R_c + R_e} \quad (\text{A.12})$$

Mit der Anfangsbedingung in A.12:

$$V_{trap}(t = 0) = k + V_{stress} \cdot \frac{R_e}{R_c + R_e} = V_{trap0} \quad (\text{A.13})$$

ergibt sich:

$$k = V_{trap0} - V_{stress} \cdot \frac{R_e}{R_c + R_e} \quad (\text{A.14})$$

---

Hieraus ergibt sich die Lösung:

$$V_{trap}(t) = (V_{trap0} - V_{stress} \cdot \frac{R_e}{R_c + R_e}) \cdot e^{-(\frac{1}{R_e \cdot C} + \frac{1}{R_c \cdot C})t} + V_{stress} \cdot \frac{R_e}{R_c + R_e} \quad (\text{A.15})$$

Mit  $\tau_{e/c} = R_{e/c} \cdot C$  kann die Gleichung auch geschrieben werden:

$$V_{trap}(t) = V_{stress} \cdot \frac{\tau_e}{\tau_c + \tau_e} + (V_{trap0} - V_{stress} \cdot \frac{\tau_e}{\tau_c + \tau_e}) \cdot e^{-t \cdot (\frac{1}{\tau_e} + \frac{1}{\tau_c})} \quad (\text{A.16})$$



---

## Tools und Dateiformate

---

In dem folgenden Abschnitt werden Tools sowie genutzte Dateiformate vorgestellt, die bei der Umsetzung des Konzeptes dieser Arbeit entstanden sind. Diese wurden von mir selbst entwickelt, bzw. entstanden unter Mitwirkung von wissenschaftlichen Hilfskräften.

### B.0.1 Tool command language - TCL

Die „Tool command language“, auch kurz TCL genannt, ist eine Skript-Sprache, die ab 1988 von John Ousterhout an der University of California (Berkeley) entwickelt wurde. Sie ist eine häufig genutzte Sprache in der Electronic Design Automation (EDA) Industrie und wird von den meisten Toolherstellern aktiv unterstützt. Mit der Skriptsprache ist es möglich, sämtliche Vorgänge, die der Benutzer in der Software umsetzen will, mit Hilfe von TCL Skripten zu automatisieren. In Auflistung B.1 ist ein TCL-Skript dargestellt, das eine Synthese auf Gatterebene für den Synopsis Design Compiler durchführt. Die meisten in den folgenden Seiten aufgelisteten Ausgabe-Formate sind mit Hilfe der genutzten Industrietools und von TCL Skripten entstanden.

```
read_file -format vhdl {adder.vhd}
analyze -library WORK -format vhdl {adder.vhd}
elaborate adder -architecture vhdl -library DEFAULT
compile -exact_map -ungroup_all
write -format verilog $current_design -output gate_level_adder.v
```

**Auflistung B.1: Beispiel TCL-Skript für den Synopsis Design Compiler, um eine Schaltung auf Registertransferebene einzulesen und daraus die Synthese auf Gatterebene auszuführen. Das Synthese-Ergebnis wird anschließend als Verilog Datei abgespeichert.**

## B.0.2 NBTI-Trap-Datei

```
## NMP four state trap
#i trapNr=0 T=398.0 xPos=6.418e-08 yPos=3.994e-10 xt=3.994e-10 weightFactor
  ↪ =1.096 Et=-1.78 Etp=-0.5061 R12p=0.02229 S12p=1.047 R1p2=0.4136 S1p2
  ↪ =1.065 EpsT2=0.04858 Eps1p1=0.1988 Eps2p2=0.5508 Vd=-0.1 W=1.5e-06 L
  ↪ =4.5e-08 tox1=1.15e-09 epsr1=3.9 tox2=9.5e-10 epsr2=25.0 etar=1.0
#n Vg tauc taue Eox Eleff EatIfChannel charge
#u V s s V/m eV eV 1
1.0 5.8454937e+21 5.4726882e-06 -234915031.0 -2.11428575 0.368232352 0.0
0.9 2.78494246e+21 1.46736182e-05 -178530035.0 -2.05189843 0.41794512 0.0
...
-0.9 1.59988827e+12 0.0607167772 379942886.0 -1.05040194 1.14007731 0.0
-1.0 1.0271339e+12 0.0972989193 446453393.0 -1.01634947 1.1520083 0.0
-1.1 6.85440547e+11 0.15765878 515522076.0 -0.983500256 1.1623235 0.0
-1.2 4.71639725e+11 0.257987527 586214461.0 -0.95160904 1.17132927 0.0
...
```

Auflistung B.2: Auszug aus einer CRV Datei eines NBTI-Traps, in der die  $\tau$  Werte für unterschiedliche Stressspannungen gespeichert werden. Quelle: [61]

## B.0.3 Versorgungsnetz

```
<Design Name="example">
  <PowerNets>
    <Net Name="VDD">
      <Wire X1="15.065" Y1="100.1" X2="100.13" Y2="100.1"
        ↪ Shape="corewire" Width="0.17" Layer="M1"/>
      <Wire X1="15.065" Y1="102.9" X2="100.13" Y2="102.9"
        ↪ Shape="corewire" Width="0.17" Layer="M1"/>
      ...
    </Net>
    <Net Name="VSS">
      <Wire X1="55.065" Y1="109.9" X2="100.13" Y2="109.9"
        ↪ Shape="corewire" Width="0.17" Layer="M1"/>
      <Wire X1="55.065" Y1="107.1" X2="100.13" Y2="107.1"
        ↪ Shape="corewire" Width="0.17" Layer="M1"/>
      ...
    </Net>
  </PowerNets>
```



```
</Design>
```

Auflistung B.3: XML-Datei, in der der Aufbau eines Versorgungsnetzes gespeichert ist.

#### B.0.4 Standard delay format - SDF

```
(DELAYFILE
(SDFVERSION "OVI 2.1")
(DESIGN "adder")
(DATE "01/07/2017")
(PROGRAM "aged SDF")
(DIVIDER /)
(TIMESCALE 1ns)

(CELL
  (CELLTYPE "AND2_X1")
  (INSTANCE U1)
  (DELAY
    (ABSOLUTE
      (IOPATH A1 ZN (0.0099016:0.0099016:0.0099016)
        ↪ (0.0147959:0.0147959:0.0147959))
      (IOPATH A2 ZN (0.0110669:0.0110669:0.0110669) (0.0158:0.0158:0.0158))
    )
  )
)
...
)
```

Auflistung B.4: Beispiel einer SDF-Datei, in der die Delay Informationen eines AND-Gatters eingetragen sind.

#### B.0.5 XML-Datei für die Delay-Berechnung

```
<Design Name="Design_05">
<Cell Name="U7" Type="NAND3_X1" Voltage="1.1" Temp="125" DelVth="0.064 0.064
  ↪ 0.064 0 0 0" Inputs="A1 A2 A3" Outputs="ZN" SignalProp="0 0.500 0.655"
  ↪ TransProp="0 0.1000 0.0156" LoadCap="2.229003">
<Cell Name="U8" Type="INV_X1" Voltage="1.1" Temp="125" DelVth="0.064 0" Inputs
  ↪ ="A" Outputs="ZN" SignalProp="0.837" TransProp="0.0430" LoadCap="
  ↪ 1.579341">
```

```
<Cell Name="U9" Type="OAI21_X1" Voltage="1.1" Temp="125" DelVth="0.064 0.064
  ↪ 0.064 0 0 0" Inputs="B1 B2 A" Outputs="ZN" SignalProp="0.829 0 0.829"
  ↪ TransProp="0.0068 0 0.0684" LoadCap="2.193258">
<Cell Name="U10" Type="AOI21_X1" Voltage="1.1" Temp="125" DelVth="0.064 0.064
  ↪ 0.064 0 0 0" Inputs="B1 B2 A" Outputs="ZN" SignalProp="0.655 0.500 0"
  ↪ TransProp="0.0156 0.1000 0" LoadCap="1.579341">
...
</Design>
```

**Auflistung B.5:** XML-Datei mit Informationen zur Delay-Berechnung der Gatter. Die Datei wird mit eigenen Skripten in dem Synthese-Tool erzeugt (Gatter Informationen) und anschließend werden zusätzliche Informationen durch die Alterungssimulation hinzugefügt.

## B.0.6 Package-Editor und XML-Formate

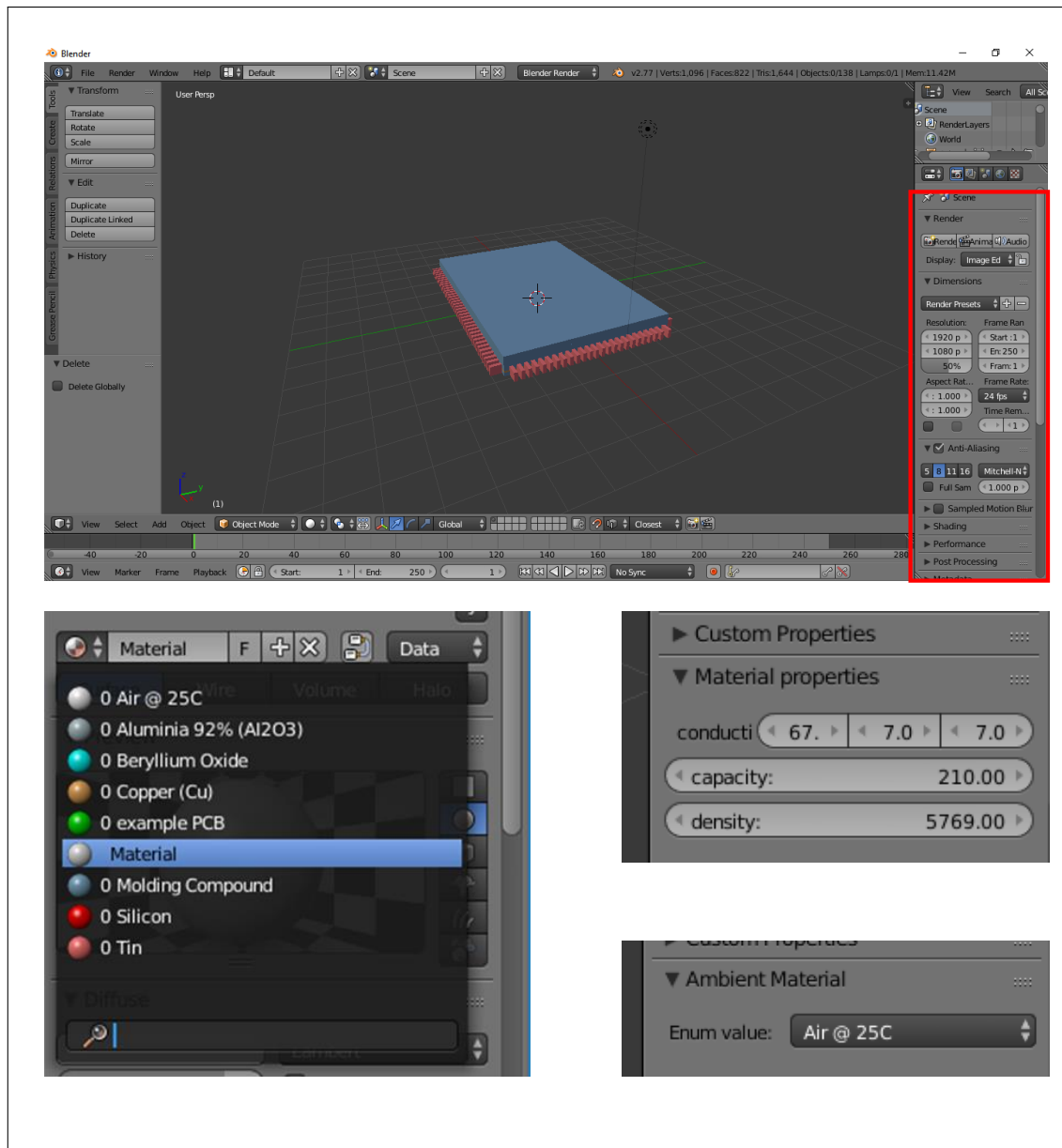


Abbildung B.1: Erweiterung für das Open Source CAD Programm Blender. Mit dieser Erweiterung kann das CAD Programm als Material-Editor benutzt werden. Es wurden in der GUI des Programms neue Elemente hinzugefügt, um bei den erstellten 3D Objekt Material Eigenschaften zuzuweisen, die für die thermische Simulation benötigt werden.

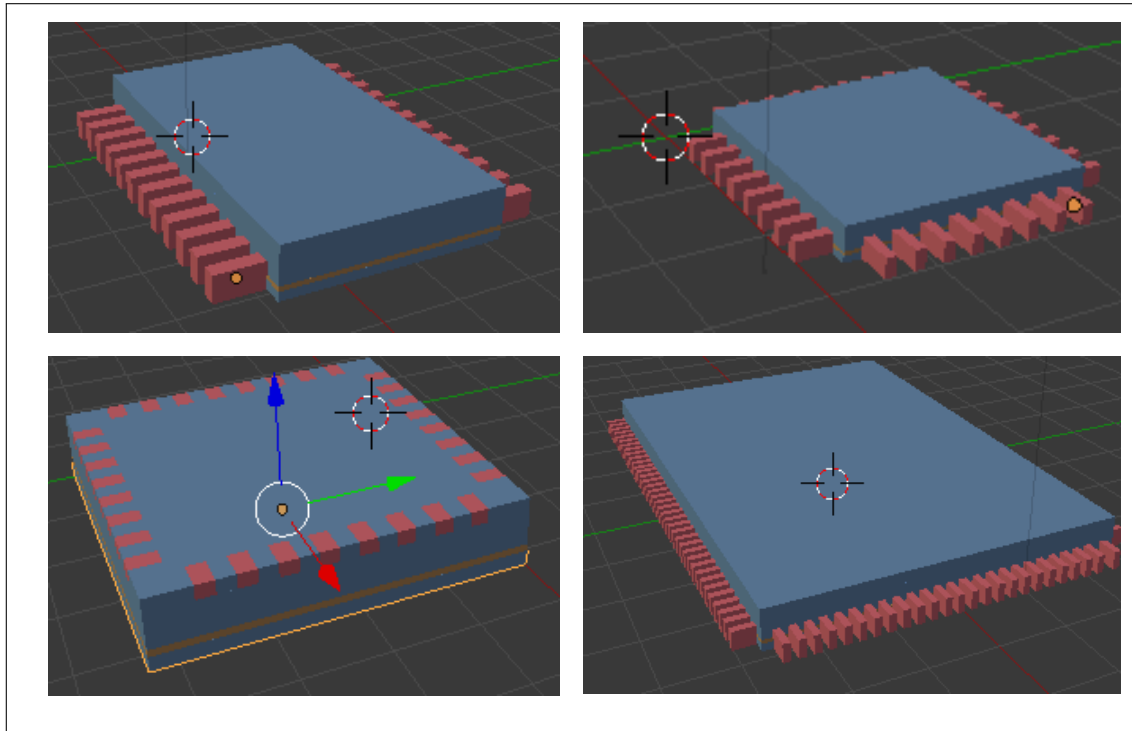


Abbildung B.2: Beispiele unterschiedlicher Packages, die mit Hilfe von Blender und der Material Erweiterung erzeugt wurden.

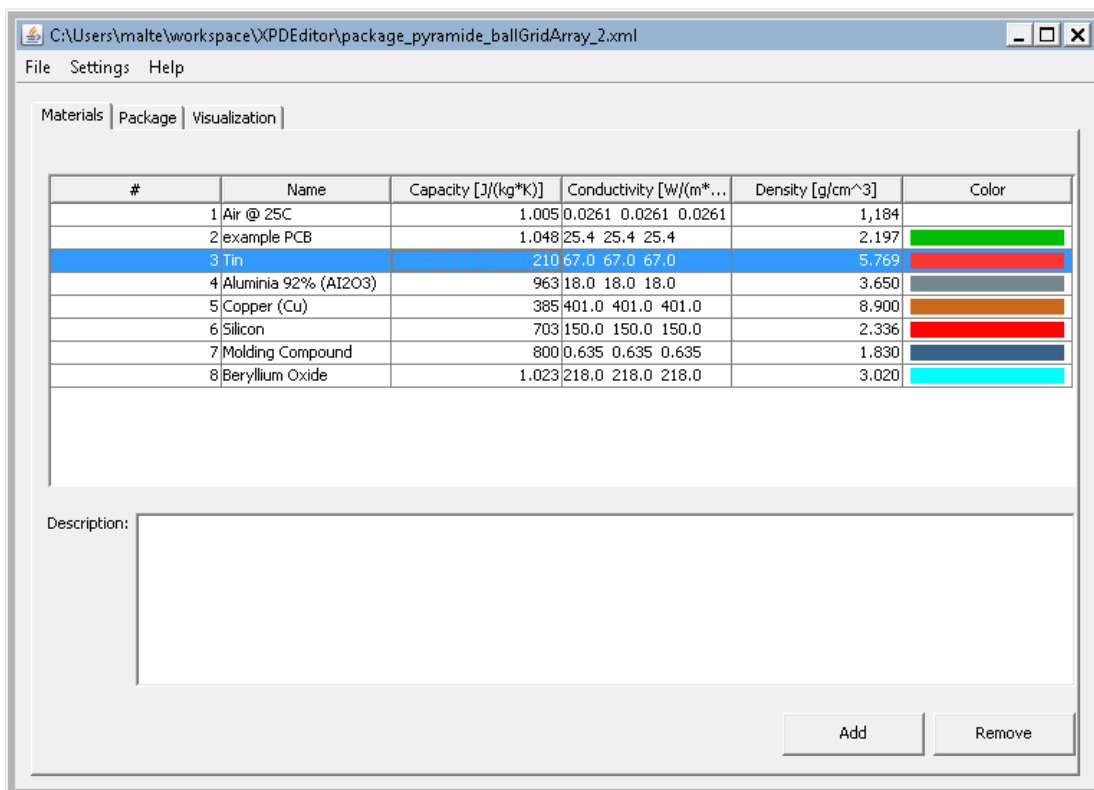


Abbildung B.3: Einfacher Material-Editor zur Bearbeitung von Materialien, die für ein Package genutzt werden können.

```

<system>
  <materials>
    <material id="material2">
      <description>PCB is made of glass-epoxy and copper</description>
      <name>example PCB</name>
      <density>2197</density>
      <capacity>1048</capacity>
      <conductivity>25.4 25.4 25.4</conductivity>
      <color>0 0.75 0</color>
    </material>
    ...
  </materials>
  <packages>
    <package>
      <size>360.0 180.0 20.0</size>
      ...
      <substrate material="material4" layerType="activeLayer">
        <position>0 0 0</position>
        <size>320.0 160.0 10.0</size>
      </substrate>
      ...
    </package>
  </packages>
</system>

```

Auflistung B.6: Beispiel einer XML-Ausgabedatei des Package-Editors

## B.0.7 Package-Slicer und XML-Datei

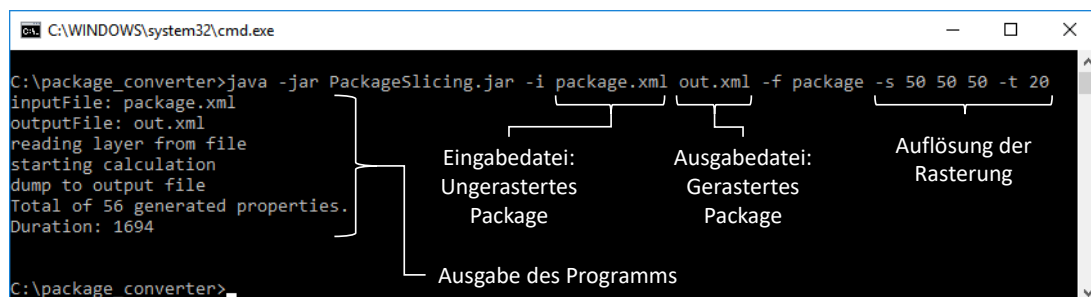


Abbildung B.4: Der Package-Slicer führt die Rasterung des Packages durch. Das Ergebnis ist ein gerastertes Package.

```

<Package XLENGTH="36.0" YLENGTH="18.0" ZLENGTH="20.0">
  <Materials>
    <Material NAME="Aluminia 92% (AI203)" KAPPX="18.0" KAPPY="18.0" KAPPZ="
      ↪ 18.0" CAPA="963.0" RHO="3650.0"/>
    <Material NAME="example PCB" KAPPX="25.4" KAPPY="25.4" KAPPZ="25.4" CAPA="
      ↪ 1048.0" RHO="2197.0"/>
    <Material NAME="Air @ 25C" KAPPX="0.0261" KAPPY="0.0261" KAPPZ="0.0261"
      ↪ CAPA="1005.0" RHO="1.184"/>
    <Material MIX="Aluminia 92% (AI203): 0.56 Air @ 25C: 0.44" KAPPX="0.06"
      ↪ KAPPY="10.01" KAPPZ="10.01" CAPA="981.67" RHO="2028.30"/>
    ...
  </Materials>
  <Composition>
    <Layer>
      <Row>0 0 0 3</Row>
      <Row>0 0 0 3</Row>
      <Row>0 0 0 3</Row>
      <Row>4 4 4 5</Row>
    </Layer>
    ...
  </Composition>
</Package>

```

**Auflistung B.7: Beispiel XML-Datei des Packages nach dem Rastern. Ausgabedatei des Package-Slicers.**

## B.0.8 Analyse-Tool für die Reduktion von kritischen Pfaden

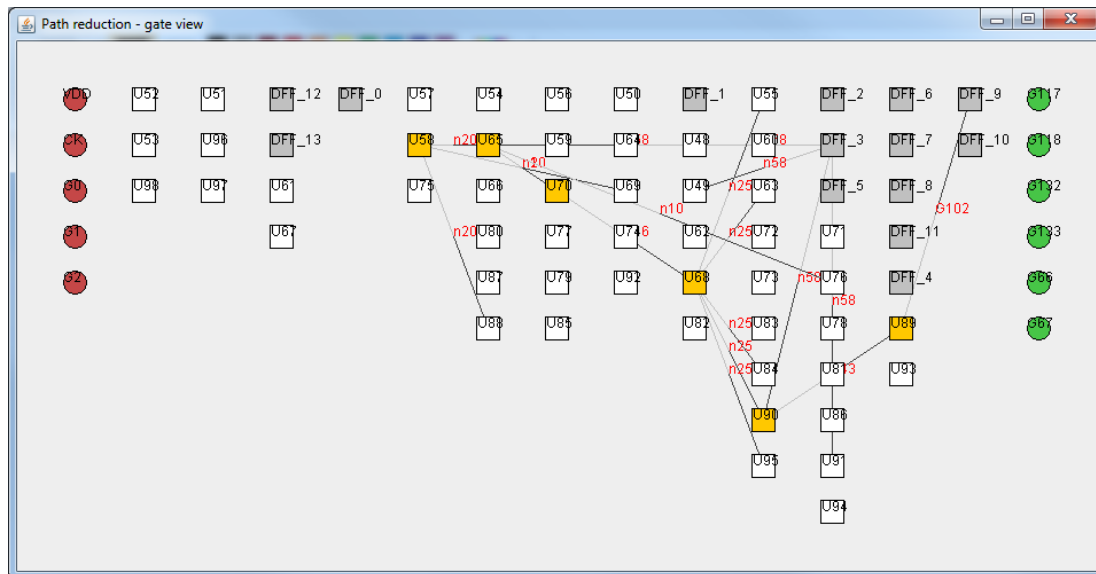


Abbildung B.5: Die Abbildung zeigt die Ein- und Ausgänge der Schaltung (rot und grün), als Blöcke sind die Gatter dargestellt, wobei Flipflops grau zu sehen sind, in gelb wurde der kritische Pfad der Schaltung hervorgehoben.

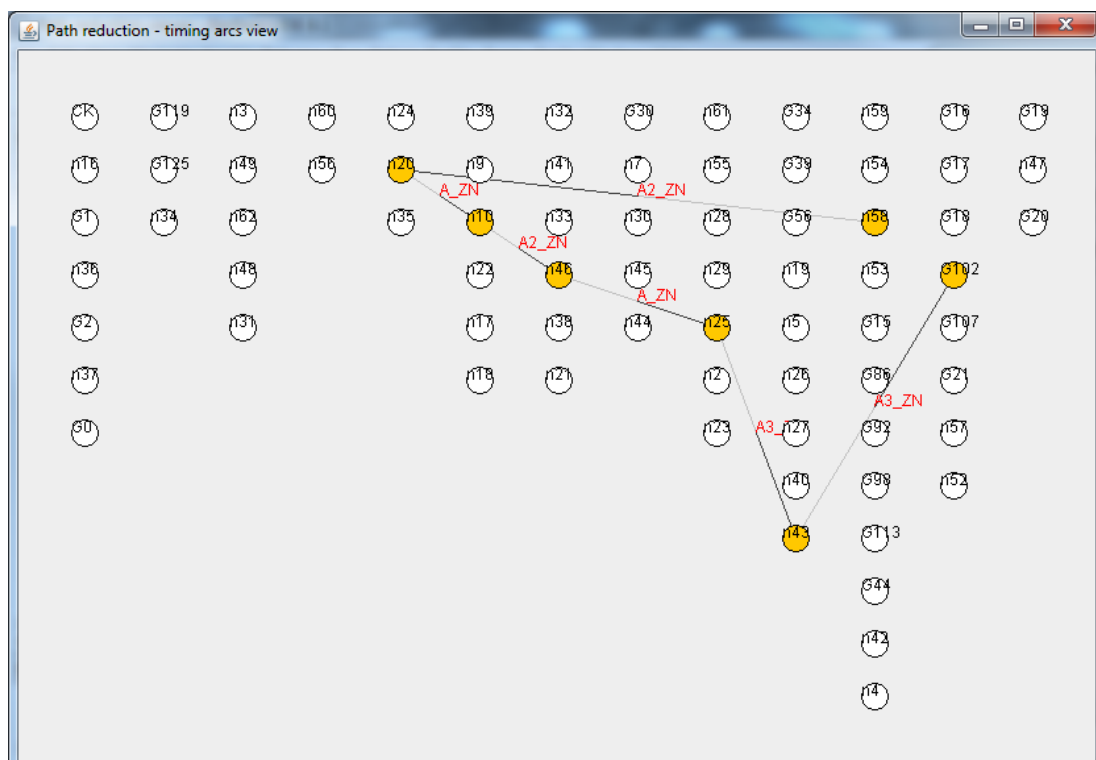


Abbildung B.6: Darstellung derselben Schaltung aus Abbildung B.5 als Timing Graph, der zur Reduktion der kritischen Pfade genutzt wird. Hervorgehoben ist hier der kritische Pfad.

## B.0.9 Missionsszenario-Editor

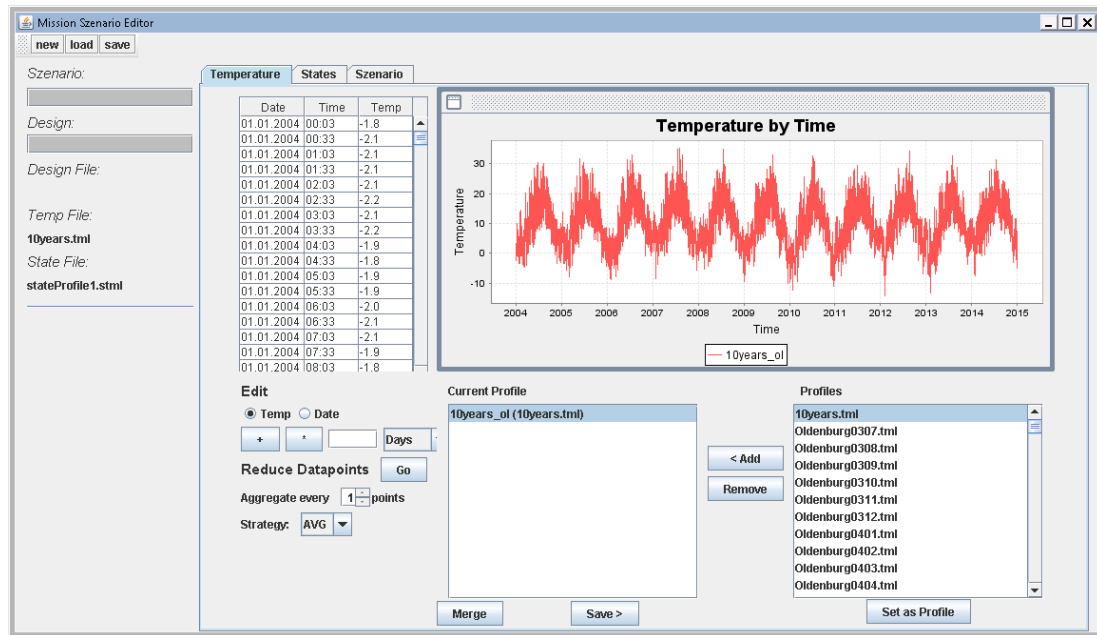


Abbildung B.7: Missionsszenario-Editor: Dargestellt ist die Konfiguration des Temperaturprofiles für das zu simulierende System.

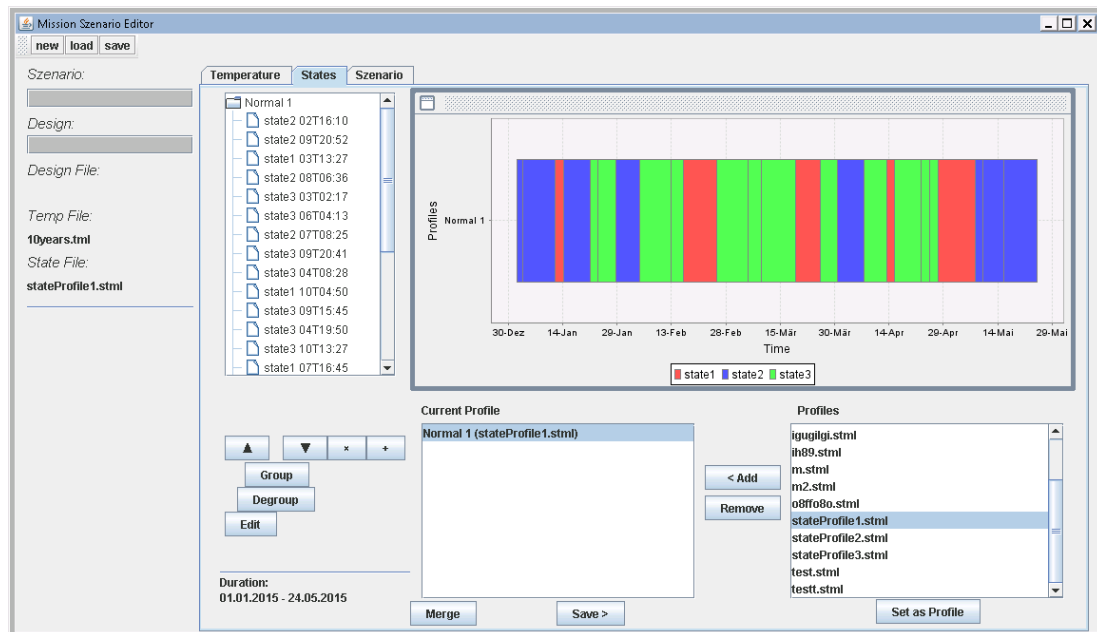


Abbildung B.8: Missionsszenario-Editor: Erstellen der Zustände des Systems die zur Simulation benutzt werden.





Abbildung B.9: Missionsszenario-Editor: Erstellung des Szenarios. Hierbei wird der Temperaturverlauf aus Abbildung B.7 mit den Zuständen des Systems zusammengeführt. Das Ergebnis ist das Missionsszenario.

## B.0.10 Toolflow-Editor

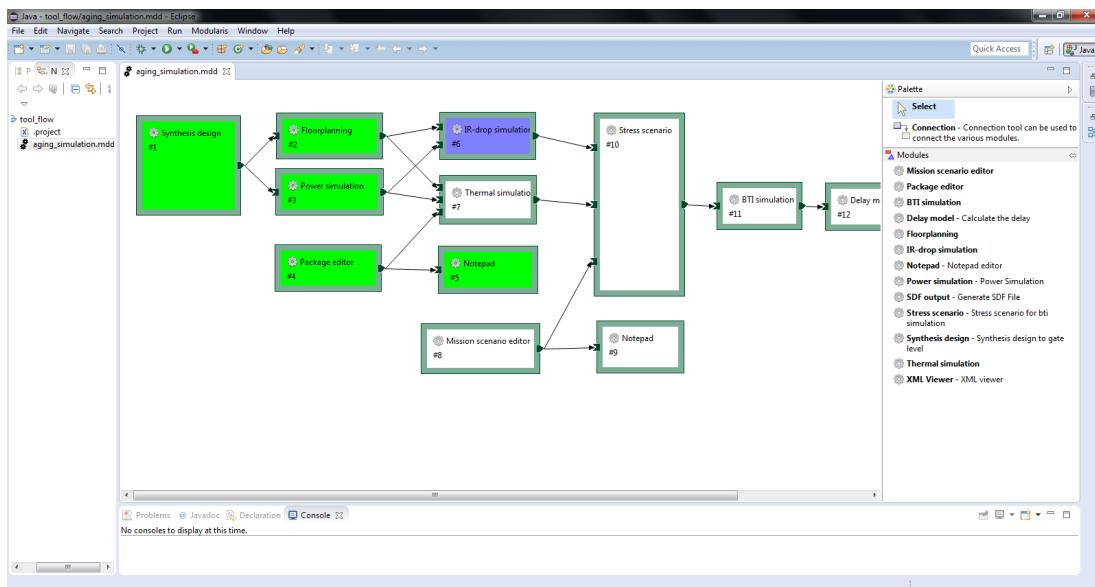


Abbildung B.10: Toolflow-Editor: Dargestellt ist der Toolflow-Editor, der die Ausführung unterschiedlicher Programme ermöglicht, zu planen und ihre Ausführung zu überwachen. In grün ist dargestellt, wenn Programme erfolgreich abgeschlossen sind, in blauer Farbe sind Programme dargestellt, die aktuell laufen. In weiß Programme, die noch ausgeführt werden müssen.

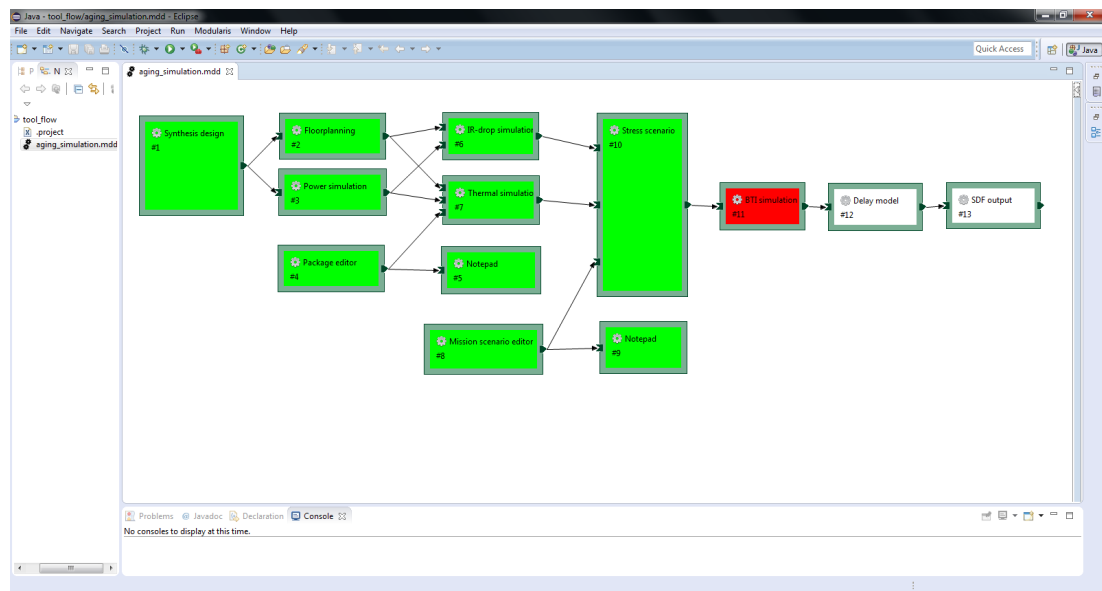


Abbildung B.11: Toolflow-Editor: Sollte eines der Programme nicht erfolgreich abschließen, ist dies in rot dargestellt. Der Toolflow-Editor kann hierauf unterschiedlich reagieren und kann z.B. das Ausführen aller anderen Programme abbrechen.

### Vergleich von GPU und CPU



Abbildung C.1: NVIDIA Kepler GK110 Grafik Prozessor (GPU): Zu sehen sind die 15 Streaming Multiprozessoren(SMX, Compute Units), die jeweils aus 192 Single Precision Units (grün) und 64 Double Precision Units (gelb) bestehen. Insgesamt sind es 2880 Single Precision Units und 960 Double Precision Units die auf eine Rechenleistung von 5.1 TFlops Single Precision und 1.7 TFlops Double Precision kommen. Quelle: [Whitepaper - NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110]

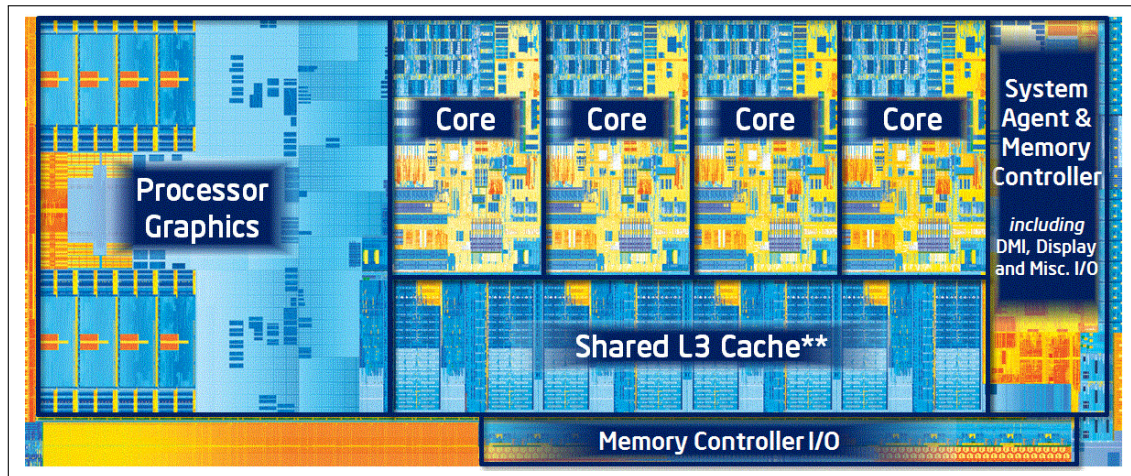


Abbildung C.2: Intel Ivy-Bridge Quad Core CPU mit zusätzlich integrierter GPU. Im Vergleich zur Grafikkarte besteht der Prozessor aus 4 Compute Units. Die integrierte Grafikkarte kann ebenfalls als OpenCL Device genutzt werden, hat aber eine deutlich geringere Leistung als eine externe Grafikkarte. Quelle: [Intel]

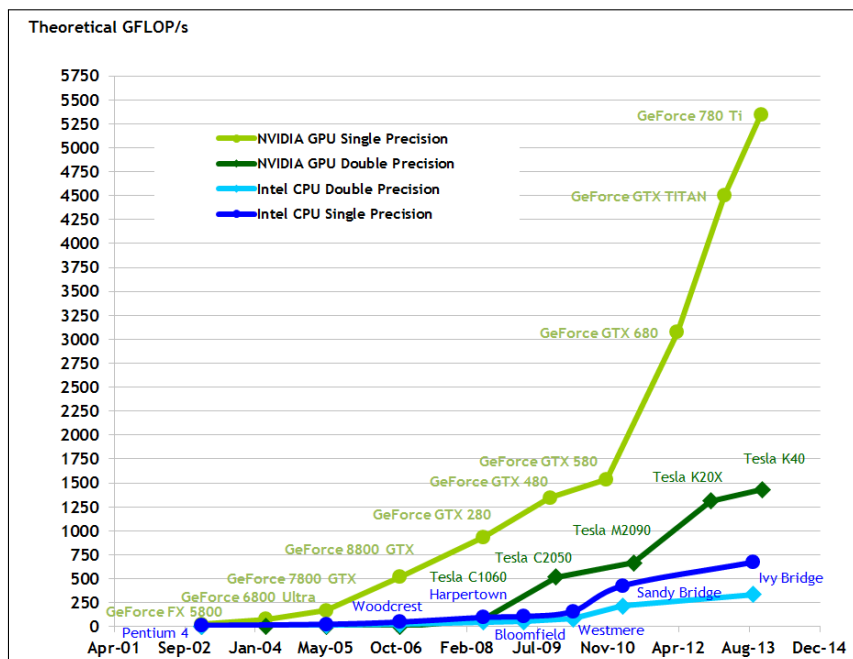


Abbildung C.3: Vergleich der theoretischen Performance von unterschiedlichen Prozessoren und Grafikkarten. Die eigentliche erreichte Performance der Devices ist von der Implementierung und von der Möglichkeit der Parallelisierung der umgesetzten Berechnung abhängig. Hier spielen auch viele weitere Faktoren eine Rolle, wie z.B. die Implementierung den Speicher nutzt. Quelle: [NVIDIA]

---

## BSIM Erweiterung

---

### **Erweiterung der SPICE-Gatter zur Temperatur-, Spannungs- und Alterungssimulation**

Um die Auswirkungen der NBTI Alterungssimulation auf das Delay in SPICE simulieren zu können, müssen bei den gealterten Gattern die Transistorparameter geändert werden. Die Anforderungen durch meine Arbeit sind eine Temperatur-, Versorgungsspannungs- und Schwellspannungsänderung innerhalb eines Gattermodells in SPICE durchzuführen.

Um die genannten Anforderungen umzusetzen, wurden die SPICE-Gatter-Modelle so angepasst, dass sie zusätzliche Parameter haben, um für jeden Transistor des Gatters eine eigene Temperatur und einen Schwellspannungsschaden zu setzen. Die Änderung der Versorgungsspannung musste nicht extra angepasst werden, da sie direkt durch die vorhandenen Versorgungseingänge der Gatter einzeln variiert werden kann. In Abbildung D.1 ist der Aufbau eines NOR-Gatters aus vier Transistoren und in Auflistung D.1 ist das erweiterte Modell der SPICE-Gatter anhand eines NOR-Gatters dargestellt. Hinter dem eigentlichen Gattertyp, in diesem Beispiel ein NOR2 mit der Treiberstärke eins, wird die Versorgungsspannung angegeben, anschließend die Ein- und Ausgänge angeschlossen. Darauf folgen die neu eingebauten Parameter, dies sind: für jeden einzelnen Transistor zuerst die Temperaturen mit dem Parameter T\_M und anschließend die Schwellspannungsschäden DELVTO\_M. Diese Erweiterung wurde für die gesamte NanGatter Bibliothek umgesetzt und die Parameterliste der einzelnen Gatter in einer Liste, die in Abbildung D.2 zu sehen ist, gespeichert. Mit dieser Liste können nach der Alterungssimulation die passenden Parameter identifiziert und zugewiesen werden.

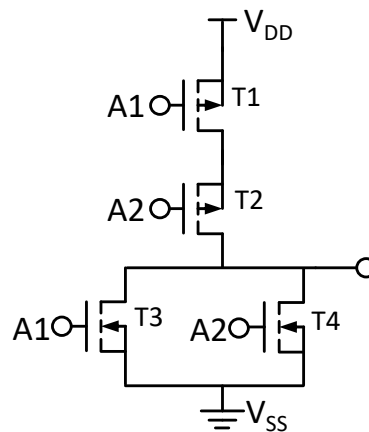


Abbildung D.1: Aufbau eines NOR-Gatters

```
.SUBCKT NOR2_X1_TV VDD VSS A2 ZN A1 TEMP_M0 DELVTO_M0 TEMP_M1 DELVTO_M1
TEMP_M2 DELVTO_M2 TEMP_M3 DELVTO_M3
```

**Versorgungsspannung** -> Voltage Scaling, Power Gating und IR-Drop

**Ein-/Ausgänge** -> Aktivität

**Temperatur** -> Temperatursimulation

**Vth** -> Schwellspannungsschaden durch Alterung

**Auflistung D.1:** Anpassung der SPICE-Gatterdateien. Abgebildet ist der Header eines NOR-Gatters, bei dem neue Parameter für die Temperatur und den Schwellspannungsschaden für jeden einzelnen Transistor hinzugefügt wurden.

```
* Auto generated nangate spice models with temperature (unit: kelvin) and
  ↳ delta vth (unit: volt) parameters
* example: .SUBCKT AND2_X1 VDD VSS A1 A2 ZN TEMP_M0=273.15 DELVTO_M0=0V ...
  ↳ TEMP_M5=300.15 DELVTO_M5=0.1V
* number of gates: 268
* filename / transistor count / spice subcircuit with new parameters
AND2_X1_lpe_tv.spi 6 .SUBCKT AND2_X1 VDD VSS A1 A2 ZN TEMP_M0 DELVTO_M0
  ↳ TEMP_M1 DELVTO_M1 TEMP_M2 DELVTO_M2 TEMP_M3 DELVTO_M3 TEMP_M4 DELVTO_M4
  ↳ TEMP_M5 DELVTO_M5
...
NOR2_X1_lpe_tv.spi 4 .SUBCKT NOR2_X1 VDD VSS A2 ZN A1 TEMP_M0 DELVTO_M0
  ↳ TEMP_M1 DELVTO_M1 TEMP_M2 DELVTO_M2 TEMP_M3 DELVTO_M3
...
```

**Auflistung D.2:** Liste mit allen Gattern einer Bibliothek, für die die neuen Parameter der Gatter angegeben sind.

---

## Auflistungsverzeichnis

---

5.1	Pseudo Code der elektrothermischen Kopplung . . . . .	105
B.1	Beispiel TCL-Skript . . . . .	149
B.2	NBTI-Trap-Datei . . . . .	150
B.3	Beispiel eines Versorgungsnetzes . . . . .	150
B.4	Beispiel einer SDF-Datei . . . . .	151
B.5	Datei zur Berechnung des Delays . . . . .	151
B.6	Package-Editor XML-Datei . . . . .	155
B.7	Gerastertes Package XML-Datei . . . . .	156
D.1	SPICE-Parameterliste eines Gatters . . . . .	164
D.2	Parameterliste aller Gatter einer Gatterbibliothek . . . . .	164





---

## Tabellenverzeichnis

---

2.1	Alterungseffekte von CMOS-Schaltungen . . . . .	14
2.2	Leakagewerte eines NOR-Gatters . . . . .	18
3.1	Vergleiche zu Alterungsmodellen . . . . .	28
4.1	Berechnung der Signalwahrscheinlichkeiten . . . . .	64
4.2	Signalwahrscheinlichkeiten eines AND-Gatters . . . . .	65
4.3	Delay-Simulation NAND und NOR . . . . .	69
6.1	OpenCL-Evaluationsplattform . . . . .	114
6.2	Evaluationsergebnisse von unterschiedlichen NBTI-Trap-Simulationen . . . . .	117
6.3	Evaluation der Reduktion der NBTI-Traps A . . . . .	118
6.4	Evaluation der Reduktion der NBTI-Traps B . . . . .	118
6.5	Reduktion der Zustände von Stressszenarien A . . . . .	119
6.6	Reduktion der Zustände von Stressszenarien B . . . . .	120
6.7	Geschwindigkeitsvergleich unterschiedlicher NBTI-Modellen A . . . . .	121
6.8	Geschwindigkeitsvergleich unterschiedlicher NBTI-Modellen B . . . . .	122
6.9	Vergleich unterschiedlicher NBTI-Modellen . . . . .	122
6.10	Ergebnisse der Temperatursimulation . . . . .	125
6.11	IR-Drop-Performance: NGSPICE vs. GPU . . . . .	127
6.12	IR-Drop-Reduktion . . . . .	128
6.13	Delay-Parameter . . . . .	129
6.14	Ergebnisse des Delay-Modells . . . . .	129
6.15	Ergebnisse des Slew-Rate-Modells . . . . .	130
6.16	Performance SPICE vs. Delay-Modell . . . . .	130
6.17	Kritische Pfade ISCAS-Benchmark . . . . .	131
6.18	Ergebnisse eines 32bit-Addierers . . . . .	133

6.19 Ergebnisse des DVFS Szenarios . . . . .	134
6.20 Ergebnisse des Power Gating Szenarios . . . . .	135
6.21 Ergebnisse des Pin Swapping Szenarios . . . . .	135
6.22 Ergebnisse des Temperaturszenarios . . . . .	136
6.23 Ergebnisse des LEON 3 Prozessors . . . . .	137

---

## Abbildungsverzeichnis

---

1.1	Fertigungsprozesse der Computer- und Automobilindustrie . . . . .	6
1.2	Grenzen der Fertigungstechnik . . . . .	8
2.1	Y-Diagramm . . . . .	12
2.2	Dreidimensionale Darstellung von MOSFETs . . . . .	13
2.3	Zuverlässigkeit von Systemen . . . . .	14
2.4	NBTI-Multi-State-Modell . . . . .	16
2.5	NBTI-RC-Trap-Modell . . . . .	16
2.6	CET-Karte . . . . .	17
2.7	Unterschiedliche Leakage-Varianten . . . . .	18
2.8	IR-Drop-Schaltung . . . . .	19
2.9	Elektrothermische Kopplung . . . . .	20
2.10	Synchron getaktete Digitalschaltung . . . . .	20
2.11	Definition des Delays . . . . .	21
2.12	Definition der Slew Rate . . . . .	21
2.13	Reduktion der kritischen Pfade . . . . .	22
2.14	Beispiel eines QFN32-Packages . . . . .	24
2.15	RC-Netzwerk . . . . .	24
2.16	ASIC-Designflow . . . . .	25
3.1	Thermische Simulation mit Green-Funktion . . . . .	31
4.1	Übersicht der entwickelten Modelle . . . . .	38
4.2	Anforderungen an das NBTI-Modell . . . . .	39
4.3	NBTI-Trap-Ersatzschaltung . . . . .	39
4.4	NBTI-Modell: Spannungs- und Temperaturabhängigkeit . . . . .	41
4.5	Logiksignale bei der Trap-Simulation . . . . .	43

---

4.6	Ergebnisse unterschiedlicher Trap-Simulationen . . . . .	44
4.7	NBTI-Eingangsaktivität . . . . .	45
4.8	Beispiel einer NBTI-Eingangsaktivität . . . . .	46
4.9	NBTI-Modell: short-long-term Unterteilung . . . . .	47
4.10	NBTI-Trap-Reduktion . . . . .	48
4.11	Ablauf der NBTI-Alterungssimulation . . . . .	49
4.12	Temperaturverlauf Oldenburg . . . . .	50
4.13	Entstehung von Missionsszenarien . . . . .	51
4.14	Rasterung eines Packages . . . . .	53
4.15	Beispiel einer Rasterung eines Packages . . . . .	54
4.16	Thermisches RC-Netzwerk . . . . .	55
4.17	Thermisches LZI-System . . . . .	55
4.18	Thermische Impulsantwort . . . . .	56
4.19	Versorgungsnetz . . . . .	58
4.20	Superpositionsansatz des IR-Drops . . . . .	61
4.21	Elektrothermische Kopplung . . . . .	62
4.22	Anforderungen an das Delay-Modell . . . . .	63
4.23	Darstellung eines AND-Gatters . . . . .	64
4.24	Alterung eines NOR-Gatters . . . . .	66
4.25	Alterungsrelevante Transistoren eines AND-Gatters . . . . .	67
4.26	Aufbau eines NAND- und NOR-Gatters . . . . .	68
4.27	Kritischer Pfad aus NAND-Gattern . . . . .	68
4.28	Alterung und Delay-Berechnung . . . . .	70
4.29	Lastkapazität der Gatter . . . . .	71
4.30	Kritischer Pfad einer Schaltung . . . . .	72
4.31	Syntheseflow . . . . .	73
4.32	SDF-Flow . . . . .	74
4.33	Gesamtflow . . . . .	75
4.34	Konzept zur Toolausführung . . . . .	77
4.35	Systembewertung einer gealterten Komponente . . . . .	78
5.1	OpenCL Compute Device . . . . .	85
5.2	OpenCL-Speicherhierarchie . . . . .	86
5.3	OpenCL Host Interface . . . . .	86
5.4	Beispiel zu Workgroup und Workitems . . . . .	88
5.5	OpenCL-NBTI-Flow . . . . .	91
5.6	Missionsszenario Temperaturverlauf . . . . .	92
5.7	Missionsszenario Zustandsverlauf . . . . .	92
5.8	Verbindung des Temperatur- und Zustandsverlaufs . . . . .	93
5.9	Diskretisierung der Temperaturen . . . . .	93

---

5.10	Rasterung eines Packages . . . . .	94
5.11	Berechnung der Materialeigenschaften . . . . .	95
5.12	Ablauf einer thermischen Charakterisierung . . . . .	97
5.13	Beispiel einer thermischen Charakterisierung . . . . .	97
5.14	Beispiel einer thermischen Simulation . . . . .	98
5.15	OpenCL - thermischer Flow . . . . .	99
5.16	Dynamische Verlustleistung . . . . .	101
5.17	Statische Verlustleistung . . . . .	101
5.18	Versorgungsnetz des IR-Drops . . . . .	102
5.19	OpenCL-IR-Drop-Flow . . . . .	103
5.20	Delay-Berechnung . . . . .	106
5.21	Kritische-Pfad-Reduktion . . . . .	107
5.22	Delay-Berechnung . . . . .	108
5.23	Überarbeiteter Designflow . . . . .	109
5.24	Konzept zur Toolausführung . . . . .	110
6.1	Performance der Trap-Simulation A . . . . .	123
6.2	Performance der Trap-Simulation B . . . . .	124
6.3	Floorplan und Temperatur des Alpha-EV6 . . . . .	125
6.4	Fehler des Temperatur-Modells . . . . .	126
6.5	Performance des thermischen Modells A . . . . .	126
6.6	Performance des thermischen Modells B . . . . .	127
6.7	Performance des IR-Drops . . . . .	128
6.8	Missionsszenario . . . . .	132
6.9	Stressszenario A . . . . .	133
6.10	Stressszenario DVFS . . . . .	134
6.11	Stressszenario Power Gating . . . . .	134
A.1	NBTI-Trap-Ersatzschaltung . . . . .	145
B.1	Material-Editor - Blender Erweiterung . . . . .	153
B.2	Beispiel Packages . . . . .	154
B.3	Einfacher Material-Editor . . . . .	154
B.4	Package-Slicer . . . . .	155
B.5	Analyse-Tool kritischer Pfad A . . . . .	157
B.6	Analyse-Tool kritischer Pfad B . . . . .	157
B.7	Missionsszenario-Editor A . . . . .	158
B.8	Missionsszenario-Editor B . . . . .	158
B.9	Missionsszenario-Editor C . . . . .	159
B.10	Toolflow-Editor A . . . . .	159

B.11 Toolflow-Editor B . . . . .	160
C.1 GPU NVIDIA Kepler GK110 . . . . .	161
C.2 CPU Intel Ivy-Bridge . . . . .	162
C.3 Performance GPU vs. CPU . . . . .	162
D.1 Aufbau eines NOR-Gatters . . . . .	164

---

## Literatur

---

- [1] Cobham Gaisler AB. *LEON 3, 32-bit Prozessor*. <http://gaisler.com/leon3/>. Online; Stand 01. Juni 2017.
- [2] Martin Barke. „Aging Aware Robustness Validation of Digital Integrated Circuits“. In: *Dissertation* (2014).
- [3] Friedrich-Karl Beier und Andreas Heinemann. *Patent- und Musterrecht*. Beck-Texte im dtv, 2006.
- [4] Jayaram Bhasker und Rakesh Chadha. *Static Timing Analysis for Nanometer Designs*. Springer Verlag, 2009.
- [5] Pradip Bose, Zhigang Hu, Jude A. Rivers, Jeonghee Shin und Victor Zyuban. „Method of predicting microprocessor lifetime reliability using architecture-level structure-aware techniques“. In: *IBM, Patent US2008256383* (2008).
- [6] Ilja N. Bronstein. *Taschenbuch der Mathematik*. Harri Deutsch Verlag, 2001.
- [7] Cadence. *CAD Software*. <https://www.cadence.com/>. Online; Stand 01. Juni 2017.
- [8] Yu Cao, Jyothi Velamala, Ketul Sutaria, Mike Shuo-Wei Chen, Jonathan Ahlbin, Ivan Sanchez Esqueda, Michael Bajura und Michael Fritze. „Cross-Layer Modeling and Simulation of Circuit Reliability“. In: *TCAD* (2014).
- [9] Francky Catthoor, Ben Kaczer, Dimitrios Rodopoulos, de Almeida Camargo Vinicius Valduga und Mahato Swaraj Bandhu. „Time and workload dependent circuit simulation“. In: *IMEC, Patent EP2509011* (2012).
- [10] Rakesh Chadha und Jayaram Bhasker. *An ASIC Low Power Primer*. Springer Verlag, 2013.
- [11] Yi-Kan Cheng, Ching-Han Tsai, Chin-Chi Teng und Sung-Mo (Steve) Kang. *Electrothermal Analysis of VLSI Systems*. Kluwer Academic Publishers, 2002.

- 
- [12] Ian Cutress. *19.2 billion transistors in a single package*. <https://www.anandtech.com/show/11183/amd-prepares-32-core-naples-cpus-for-1p-and-2p-servers-coming-in-q2>. Online; Stand 01. Juni 2017.
- [13] Michael DeBole, K. Ramakrishnan, Varsha Balakrishnan, Wenping Wang, Hong Luo, Yu Wang, Yuan Xie, Yu Cao und N. Vijaykrishnan. „A Framework for Estimating NBTI Degradation of Microarchitectural Components“. In: *ASP-DAC* (2009).
- [14] Reef Eilers. „Abstraction of Aging Models for High Level Degradation Prediction“. In: *Dissertation* (2017).
- [15] Reef Eilers, Malte Metzdorf, Domenik Helms und Wolfgang Nebel. „Efficient NBTI Modeling Technique Considering Recovery Effects“. In: *ISLPED* (2014).
- [16] Martin Fischer. *Bericht: Netgear plant Rückruf von NAS- und WLAN-Controllern mit fehlerhaften Intel-Prozessoren*. <https://www.heise.de/newsticker/meldung/Bericht-Netgear-plant-Rueckruf-von-NAS-und-WLAN-Controllern-mit-fehlerhaften-Intel-Prozessoren-3645470.html>. Online; Stand 01. Juni 2017.
- [17] Daniel D. Gajski und Robert H. Kuhn. *Guest Editor's Introduction: New VLSI Tools*. IEEE Computer, 1983.
- [18] Franz Graser. *Halbleitermarkt überspringt erstmals 400-Milliarden-Dollar-Marke*. <https://www.elektronikpraxis.vogel.de/halbleitermarkt-ueberspringt-erstmal-400-milliarden-dollar-marke-a-623824/>. Online; Stand 01. Juni 2017.
- [19] Tibor Grasser. „Stochastic charge trapping in oxides: From random telegraph noise to bias temperature instabilities“. In: *Microelectronics Reliability* (2011).
- [20] Tibor Grasser. *Bias Temperature Instability for Devices and Circuits*. Springer Verlag, 2014.
- [21] Tibor Grasser, Ben Kaczer, Wolfgang Goes, Thomas Aichinger, Philipp Hehenberger und Michael Nelhiebel. „A Two-Stage Model for Negative Bias Temperature Instability“. In: *IRPS* (2009).
- [22] Tibor Grasser, Ben Kaczer, Wolfgang Goes, Hans Reisinger, Thomas Aichinger, Philipp Hehenberger, Paul-Jürgen Wagner, Franz Schanovsky, Jacopo Franco, Maria Toledano Luque und Michael Nelhiebel. „The Paradigm Shift in Understanding the Bias Temperature Instability: From Reaction-Diffusion to Switching Oxide Traps“. In: *TED* (2011).
- [23] Mark C. Hansen, Hakan Yalcin und John P. Hayes. „Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering“. In: *IEEE Design & Test of Computers* (1999).
- [24] Domenik Helms. „Leakage Models for High Level Power Estimation“. In: *Dissertation* (2009).
- [25] Domenik Helms, Kim Grüttner, Reef Eilers, Malte Metzdorf, Kai Hylla, Frank Poppen und Wolfgang Nebel. „Considering Variation and Aging in a Full Chip Design Methodology at System Level“. In: *ESLsyn* (2014).



- 
- [26] Virginia Martin Heriz, Je-Hyoung Park, Travis Kemper, Sung-Mo Kang und Ali Shakouri. „Method of images for the fast calculation of temperature distributions in packaged VLSI chips“. In: *THERMINIC* (2007).
- [27] V. Huard, E. Pion, F. Cacho, D. Croain, V. Robert, R. Delater, P. Mergault, S. Engels, P. Flatresse, N. Ruiz Amador und L. Anghel. „A predictive bottom-up hierarchical approach to digital system reliability“. In: *IRPS* (2012).
- [28] Scott Huck. *Measuring Processor Power*. White Paper Intel, 2011.
- [29] Howard Huff. *Into The Nano Era: Moore's Law Beyond Planar Silicon CMOS*. Springer Verlag, 2009.
- [30] Kai Hylla, Malte Metzdorf, Armin Grünwald, Kai Hahn, Andy Heinig, Uwe Knöchel, Susann Wolf, Felix Miller, Thomas Wild, Artur Quiring, Markus Olbrich, Sebastian Sattler und Dieter Treytnar. „NEEDS - Nanoelektronik-Entwurf für 3D-Systeme“. In: *ZuE* (2012).
- [31] „IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process“. In: *IEEE Std. 1497-2001* (2001).
- [32] Yury Illarionov, Anderson Smith, Sam Vaziri, Mikael Ostling, Thomas Mueller, Max Lemme und Tibor Grasser. „Hot-Carrier Degradation and Bias-Temperature Instability in Single-Layer Graphene Field-Effect Transistors: Similarities and Differences“. In: *TED* (2015).
- [33] Sasan Iman und Massoud Pedram. *Logic Synthesis for Low Power VLSI Designs*. Springer Verlag, 1998.
- [34] *International Technology Roadmap for Semiconductors (ITRS) - executive report*. <http://www.itrs2.net/>. Online; Stand 01. Juni 2017.
- [35] Keith A. Jenkins und Barry P. Linder. „BTI DEGRADATION TEST CIRCUIT“. In: *IBM, Patent US2017276728* (2017).
- [36] B. Kaczer, S. Mahato, V. Valduga de Almeida Camargo, M. Toledano-Luque, Ph. J. Roussel, T. Grasser, F. Catthoor, P. Dobrovolny, P. Zuber, G. Wirth und G. Groeseneken. „Atomistic approach to variability of bias-temperature instability in circuit simulations“. In: *IRPS* (2011).
- [37] David Kaeli, Perhaad Mistry, Dana Schaa und Dong Ping Zhang. *Heterogeneous Computing with OpenCL 2.0*. Morgan Kaufmann, 2015.
- [38] Travis Kemper, Yan Zhang, Zhixi Bian und Ali Shakouri. „Ultrafast Temperature Profile Calculation“. In: *THERMINIC* (2006).
- [39] Halil Kükner, Moustafa Khatib, Sebastien Morrison, Pieter Weckx, Praveen Raghavan, Ben Kaczer, Francky Catthoor, Liesbet Van der Perre, Rudy Lauwereins und Guido Groeseneken. „Degradation Analysis of Datapath Logic Subblocks under NBTI Aging in FinFET Technology“. In: *ISQED* (2014).

- [40] Nils Koppaetzky, Malte Metzdorf, Reef Eilers, Domenik Helms und Wolfgang Nebel. „RT Level Timing Model for Aging Prediction“. In: *DATE* (2016).
- [41] Ralf Kories. *Taschenbuch der Elektrotechnik*. Harri Deutsch Verlag, 2003.
- [42] Karl Küpfmüller. *Einführung in die theoretische Elektrotechnik*. Springer Verlag, 1988.
- [43] Frank Kreith, Raj M. Manglik und Mark S. Bohn. *Principles of Heat Transfer*. Cengage Learning, 2010.
- [44] Woojoo Lee, Yanzhi Wang, Tiansong Cui, Shahin Nazarian und Massoud Pedram. „Dynamic Thermal Management for FinFET-Based Circuits Exploiting the Temperature Effect Inversion Phenomenon“. In: *ISLPED* (2014).
- [45] Sung Kyu Lim. *Design for High Performance, Low Power, and Reliable 3D Integrated Circuits*. Springer Verlag, 2013.
- [46] Dominik Lorenz. „Aging Analysis of Digital Integrated Circuits“. In: *Dissertation* (2012).
- [47] Dominik Lorenz, Martin Barke und Ulf Schlichtmann. „Aging analysis at gate and macro cell level“. In: *ICCAD* (2010).
- [48] Dominik Lorenz, Georg Georgakos und Ulf Schlichtmann. „Aging analysis of circuit timing considering NBTI and HCI“. In: *IOLTS* (2009).
- [49] David Manners. *Intel Out-Spends Everyone On R&D*. <https://www.electronicweekly.com/blogs/mannerisms/markets/intel-spends-everyone-rd-2017-02/>. Online; Stand 01. Juni 2017.
- [50] Elie Maricau und Georges Gielen. *Analog IC Reliability in Nanometer CMOS*. Springer Verlag, 2013.
- [51] Malte Metzdorf und Wolfgang Nebel. „Funktional-thermische Simulation von 3D-Systemen“. In: *ZuE* (2013).
- [52] Malte Metzdorf, Domenik Helms, Reef Eilers und Wolfgang Nebel. „Abstracting TCAD aging models above the circuit level“. In: *DATE* (2015).
- [53] Malte Metzdorf, Reef Eilers, Domenik Helms, Kay-Uwe Giering Wolfgang Nebel, Roland Jancke, Gerhard Rzepa, Tibor Grasser, Markus Karner, Praveen Raghavan, Ben Kaczer, Dan Alexandrescu, Adrian Evans, Gunnar Rott, Peter Rotter, Hans Reisinger und Wolfgang Gustin. „MoRV: Modelling Reliability under Variability“. In: *SELSE* (2016).
- [54] Evelyn Mintarno, Vikas Chandra, David Pietromonaco, Robert Aitken und Robert W. Dutton. „Workload Dependent NBTI and PBTI Analysis for a sub-45nm Commercial Microprocessor“. In: *IRPS* (2013).
- [55] Aaftab Munshi, Benedict R. Gaster, Timothy G. Mattson, James Fung und Dan Ginsburg. *OpenCL Programming Guide*. Addison-Wesley, 2012.
- [56] NanGate. *NanGate Open Cell Library*. <http://www.nangate.com/>. Online; Stand 01. Juni 2017.

- [57] NGSPICE. *NGSPICE, Schaltungssimulator*. <http://ngspice.sourceforge.net/>. Online; Stand 01. Juni 2017.
- [58] Carl von Ossietzky Universität Oldenburg. *Klimawerte in Oldenburg*. <https://www.uni-oldenburg.de/wetter/>. Online; Stand 01. Juni 2017.
- [59] Je-Hyoung Park, Ali Shakouri und Sung-Mo Kang. „Fast Evaluation Method for Transient Hot Spots in VLSI ICs in Packages“. In: *ISQED* (2008).
- [60] Europäisches Patentamt. *Suchmaschine für Patente*. <https://worldwide.espacenet.com/>. Online; Stand 01. Juni 2017.
- [61] MoRV project. *Modeling of Reliability under Variability project & Representative trap data for PTM*. <https://morv-project.eu/>. Online; Stand 01. Juni 2017.
- [62] Wolfgang Reisig. *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien*. Vieweg+Teubner Verlag, 2010.
- [63] Bernd Rosenstengel und Udo Winand. *Petri-Netze: Eine anwendungsorientierte Einführung*. Vieweg Verlag, 1991.
- [64] Sven Rosinger, Malte Metzdorf, Domenik Helms und Wolfgang Nebel. „Behavioral-Level Thermal- and Aging-Estimation Flow“. In: *LATW* (2011).
- [65] Matthew Scarpino. *OpenCL in Action: How to Accelerate Graphics and Computations*. Manning Publications, 2012.
- [66] Kevin Skadron, Mircea R. Stan, Wei Huang, Sivakumar Velusamy, Karthik Sankaranarayanan und David Tarjan. „Temperature-Aware Microarchitecture“. In: *ISCA* (2003).
- [67] Global TCAD Solutions. *TCAD Simulationen*. <http://www.globaltcad.com>. Online; Stand 01. Juni 2017.
- [68] Z. Stanojevic, O. Baumgartner, F. Mitterbauer, H. Demel, C. Kernstock, M. Karner, V. Eyert, A. France-Lanord, P. Saxe, C. Freeman und E. Wimmer. „Physical modeling - A new paradigm in device simulation“. In: *IEDM* (2015).
- [69] PwC's Strategy&. *Top 20 R&D Spenders 2005-2016*. [https://www.strategyand.pwc.com/global/home/what-we-think/innovation1000/top-20-rd-spenders-2016-interactive\\_only](https://www.strategyand.pwc.com/global/home/what-we-think/innovation1000/top-20-rd-spenders-2016-interactive_only). Online; Stand 01. Juni 2017.
- [70] Haihua Su, Frank Liu, Anirudh Devgan, Emrah Acar und Sani Nassif. „Full chip leakage-estimation considering power supply and temperature variations“. In: *ISLPED* (2003).
- [71] Synopsys technology. *CAD Software*. <https://www.synopsys.com/>. Online; Stand 01. Juni 2017.
- [72] Arizona State University. *Predictive Technology Model (PTM), SPICE Transistor Modelle*. <http://ptm.asu.edu/>. Online; Stand 01. Juni 2017.
- [73] Ahmet Unutulmaz, Domenik Helms, Reef Eilers, Malte Metzdorf, Ben Kaczer und Wolfgang Nebel. „Analysis of NBTI Effects on High Frequency Digital Circuits“. In: *DATE* (2016).

- [74] Alessandro Vincenzi, Arvind Sridhar, Martino Ruggiero und David Atienza. „Fast thermal simulation of 2D/3D integrated circuits exploiting neural networks and GPUs“. In: *ISLPED* (2011).
- [75] University of Virginia. *HotSpot, thermische Simulations*. <http://lava.cs.virginia.edu/HotSpot/>. Online; Stand 01. Juni 2017.
- [76] Ting-Yuan Wang und Charlie Chung-Ping Chen. „3-D Thermal-ADI: A Linear-Time Chip Level Transient Thermal Simulator“. In: *TCAD* (2002).
- [77] Wenping Wang, Shengqi Yang, Sarvesh Bhardwaj, Rakesh Vattikonda, Sarma Vrudhula, Frank Liu und Yu Cao. „The Impact of NBTI on the Performance of Combinational and Sequential Circuits“. In: *DAC* (2007).
- [78] Xi Wang, Ali Shakouri, Sina Farsiu und Peyman Milanfar. „Extraction of Power Dissipation Profile in an IC Chip from Temperature Map“. In: *SEMI-THERM* (2007).
- [79] M. Weißbrich, G. Payá-Vayá, L. Gerlach, H. Blume, A. Najafi und A. García-Ortiz. „FLINT+: A Runtime-Configurable Emulation-Based Stochastic Timing Analysis Framework“. In: *PATMOS* (2017).
- [80] Neil H. E. Weste und David M. Harris. *CMOS VLSI Design - A Circuits and Systems Perspective*. Addison-Wesley, 2011.
- [81] Yannick Wimmer, Al-Moatasem El-Sayed, Wolfgang Gös, Tibor Grasser und Alexander L. Shluger. „Role of hydrogen in volatile behaviour of defects in SiO<sub>2</sub>-based electronic devices“. In: *RSPA* (2016).
- [82] Christof Windeck. *Intel-Chef verabschiedet sich vom bisherigen Moore's Law*. <https://www.heise.de/newsticker/meldung/Intel-Chef-verabschiedet-sich-vom-bisherigen-Moore-s-Law-2751848.html>. Online; Stand 01. Juni 2017.
- [83] Christof Windeck. *Intel Hyper-Scale führt das Moore'sche Gesetz weiter*. <https://www.heise.de/newsticker/meldung/Intel-Hyper-Scale-fuehrt-das-Moore-sche-Gesetz-weiter-3669179.html>. Online; Stand 01. Juni 2017.
- [84] Christof Windeck und Benjamin Benz. *Auf Sand gebaut - Fehler in Intels aktueller Chipsatz-Serie 6*. <https://www.heise.de/ct/artikel/Auf-Sand-gebaut-1186032.html>. Online; Stand 01. Juni 2017.
- [85] Kai-Chiang Wu und Diana Marculescu. „Joint Logic Restructuring and Pin Reordering against NBTI-Induced Performance Degradation“. In: *DATE* (2009).
- [86] Cheng Zhuo, Dennis Sylvester und David Blaauw. „Process variation and temperature-aware reliability management“. In: *DATE* (2010).
- [87] Amirkoushyar Ziabari und Ali Shakouri. „Fast thermal simulations of vertically integrated circuits (3D ICs) including thermal vias“. In: *ITHERM* (2012).
- [88] Amirkoushyar Ziabari, Ehsan K. Ardestani, Jose Renau und Ali Shakouri. „Fast thermal simulators for architecture level integrated circuit design“. In: *SEMI-THERM* (2011).

- 
- [89] Amirkoushyar Ziabari, Je-Hyoung Park, Ehsan K. Ardestani, Jose Renau, Sung-Mo Kang und Ali Shakouri. „Power Blurring: Fast Static and Transient Thermal Analysis Method for Packaged Integrated Circuits and Power Devices“. In: *TVLSI* (2014).