DISSERTATION

ENTWURF VON HOCHPERFORMANTEN LOW POWER MIKROPROZESSORSYSTEMEN

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften
unter der Leitung von

em. Univ. Prof. Dipl.-Ing. Dr. techn. Gerfried Zeichen
Institut E376
Institut für Automatisierungs- und Regelungstechnik

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von
Dipl.-Ing. Thomas Berndorfer
Matr. Nr. 9225099
Konrad Grefe Gasse 22
1230 Wien

Wien, im Mai 2004 (Thomas Berndorfer)

Danksagung

Mein erster Dank gilt meinem Betreuer und Mentor, Herrn em. Univ. Prof. Dipl.-Ing. Dr. techn. Gerfried Zeichen für die großzügige Unterstützung während meiner Tätigkeit am Institut und besonders während dieser Arbeit. Neben seiner fachlichen Kompetenz konnte ich vor allem im Bereich der „Soft Skills“, wie z. B. im Umgang mit Projekten und Kunden sehr viel von Prof. Zeichen lernen, wofür ich besonders dankbar bin.

Meinem Diplomarbeitsbetreuer, Herrn Dipl.-Ing. Dr. techn. Herbert Hufnagl bin ich für die freundschaftliche Zusammenarbeit und besonders für die Unterstützung bei der Umsetzung meiner Ideen sehr dankbar. Ohne Ihn wäre die Realisierung der Prototypen nicht möglich gewesen.

Dem Kollegen Dipl.-Ing. Zsolt Tamasi danke ich für seine großartigen Anstrengungen beim Routing des PCB. Herrn Dipl.-Ing. Dr. techn. Peter Rössler danke ich für die Zusammenarbeit bei der Erstellung des FPGA-Designs.

Die langjährige Zusammenarbeit mit den Kollegen Dipl.-Ing. Walter van Dyck, Dipl.-Ing. Dr. techn. Alexander Brenner, Dipl.-Ing. Hartwig Plach und Gernot Lechner war eine besondere Freude für mich. Die zahlreichen Diskussionen und das gemeinsame Vorwärtskommen in unserem Team waren stets unersetzbare Motivationsfaktoren.

Meinen Diplomanden Dipl.-Ing. Robert Kroiß, Dipl.-Ing. Robert Jelinek und Dipl.-Ing. Gerhard Khüny gebührt großer Dank für die hervorragenden Leistungen bei der Erarbeitung ihrer Diplomarbeiten, die in engem Zusammenhang mit dieser Dissertation stehen.

Den Herren Dipl.-Ing. Dr. techn. Thomas Sykora, Dipl.-Ing. Dr. techn. Andreas Kercek und Dipl.-Ing. Dirk Goldbeck danke ich für die interessante und intensive Zusammenarbeit.

Nicht zuletzt schulde ich meiner Familie, besonders meiner Frau Ulrike großen Dank für die Geduld und das Verständnis für meine Dissertationsarbeit. Der familiäre Rückhalt, die Liebe unserer Kinder und die Geborgenheit eines Zuhauses gehören zu den großartigsten und wertvollsten Gütern unserer Zeit.

Kurzfassung

Die Entwicklung von modernen, hochperformanten Mikroprozessorsystemen ist entscheidend geprägt von der Problematik hoher Verlustleistungsdichten von mehr als 10W/cm^2. Handelsübliche Mikroprozessoren für Arbeitsplatz-PCs weisen bereits Verlustleistungen in der Größenordnung von 100W auf. Neben der Rechenleistung und der Baugröße ist daher der Leistungsverbrauch zum entscheidenden Erfolgsfaktor für die Realisierbarkeit und Akzeptanz besonders von mobilen, batteriebetriebenen Endgeräten geworden.

Die vorliegende Arbeit stellt einen neuen Ansatz für die hard- und softwaretechnische Realisierung von hochperformanten Low-Power Mikroprozessorsystemen vor. Grundkonzept dabei ist die umfassende Berücksichtigung verlustleistungsrelevanter Zusammenhänge bei erweiterten Betrachtungsgrenzen und damit die vertikale Vernetzung von Verlustleistungsprozessen von der Logikebene bis zur Softwareebene.

Mithilfe dieses neuen Ansatzes wird - entsprechend dem Trend zu immer höheren Rechenleistungen im Bereich der Embedded Systeme - ein neues Mikroprozessormodul im Format einer Scheckkarte realisiert, das bei einer Taktfrequenz von 733MHz eine Leistungsaufnahme von weniger als 2W aufweist.

Zur weiteren Verbesserung der Verlustleistungsbilanz bei gleichzeitiger Erhöhung der Rechengeschwindigkeit wird eine neue Methode der energieoptimalen Cachezuteilung vorgeschlagen, mit der - ohne Änderung von Hard- und Software - die Verlustleistung um weitere 30% reduziert werden kann.

Für die softwaretechnische Implementierung wird das offene, frei verfügbare Betriebssystem „Linux“ auf die realisierte Hardware portiert.

Zur Unterstützung bei der Inbetriebnahme und zur Untersuchung der Verlustleistungseffizienz wird ein neuer Simulator entwickelt, der eine eventgesteuerte Hardwaresimulation des Mikroprozessormoduls auf Maschinensprachebene ermöglicht.

Abstract

The development of modern high performance microprocessor systems suffer from high power dissipation densities of more than 10W per cm^2. Well known microprocessors for state-of-the-art personal computers already consume a total power of 100W and above. Besides of performance and size, power dissipation has become one of the most important design constraints especially for battery-powered mobile devices.

This work presents a new approach for the hard- and software development of highly performant low-power microprocessor systems. The basic idea behind this approach is to have a wider view on all the relevant relationships concerning power dissipation and to vertically integrate power dissipation phenomena from transistor level up to software level.

By means of this approach a new microprocessor module with the size of a credit card is developed and implemented. According to the trend of a steadyly growing need for performance in the area of embedded systems the microprocessor module offers a clock frequency of up to 733MHz while consuming less than 2W of power.

To further improve both, power dissipation and performance a new method of energy-optimal cache assignment ist presented which further reduces total power consumption by typically 30% without any change in soft- and hardware.

The open source operating system „Linux“ is chosen and ported to the new microprocessor module.

To get a deep insight into cache efficiency and to support initial implementation of the hardware a new event-driven simulator software is developed which allows the execution of operating system and application software with instruction-level accuracy.

Inhaltsverzeichnis

Abkürzungen

ARM	Advanced RISC Machine
ARMv5TE	ARM Version 5 Thumb Enhanced
ASIC	Application Specific Integrated Circuit
BGA	Ball Grid Array
CBIC	Cell Based Integrated Circuit
CISC	Complex Instruction Set Computer
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
D-Cache	Datencache
DC	Direct Current
DCT	Discrete Cosine Transform
DMA	Direct Memory Access
DSP	Digital Signal Processor
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GPL	GNU General Public License
GNU	GNU's Not Unix
gpMIPS	General Purpose MIPS
IDE	Integrated Drive Electronics
I/O	Input/Output
I-Cache	Instruction- bzw. Befehlscache
JPEG	Joint Photographic Expert Group
JTAG	Joint Test Access Group
LCD	Liquid Crystal Display
LGPL	GNU Lesser General Public License
MGA	Mask Grid Array
MHV	Miss/Hit-Verhältnis
MIPS	Million Instructions Per Second
MMU	Memory Management Unit
MOS	Metal Oxide Semiconductor
MPEG	Motion Pictures Expert Group
NOP	No Operation
PC	Personal Computer

PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PIO	Parallel Input/Output
PLL	Phase Locked Loop
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RT	Real Time
RTAI	Real Time Application Interface
SBC	Single Board Computer
SDRAM	Synchronous Dynamic Random Access Memory
SMS	Short Message Service
SOC	System On Chip
SOI	Silicon On Insulator
SPEC	Standard Performance Evaluation Corporation
SPS	Speicherprogrammierbare Steuerung
TCP/IP	Transport Control Protocol/Internet Protocol
UART	Universal Asynchronous Receiver Transmitter
UMTS	Universal Mobile Telephony System
USB	Universal Serial Bus
VGA	Video Graphics Adapter
VHDL	Very High Speed Hardware Description Language
VLSI	Very Large Scale Integration
VoIP	Voice over Internet Protocol
WLAN	Wireless Local Area Network
XIP	Execute In Place

Kapitel 1

Das Problem der Verlustleistung bei hoher Performance

Die Bereitstellung hoher Rechenleistungen für komplexe und umfangreiche Aufgaben der Automatisierungs- und Computertechnik wird heute als Selbstverständlichkeit erachtet. Dank der hohen Verarbeitungsgeschwindigkeiten moderner Mikroprozessoren ist es in den letzten Jahren gelungen, berechnungsintensive und komplexe Probleme der Embedded Systeme[1] und der Integrierten Sensorik[2] zu lösen. Ein typisches Beispiel hierfür ist die digitale Bildverarbeitung [2, 3, 4, 5], wo Datenströme in der Größenordnung von 100Mbyte/s zu bewältigen sind.

Der Trend zu höheren Rechenleistungen kann naheliegend und unmittelbar aus der Entwicklung handelsüblicher Arbeitsplatzrechner und Personal Computer abgelesen werden (Abb. 1.1). Hier zeichnet sich auch bereits sehr deutlich und anschaulich eines der größten Probleme ab, welches Halbleiterhersteller bei der Entwicklung neuer Mikroprozessoren haben: die hohe Verlustleistung.

Hauptursache für die stark steigenden Verlustleistungen ist der Bedarf an immer höheren Taktfrequenzen. Maßnahmen zur Kühlung bzw. Wärmeabfuhr sind daher unvermeidbar. Für große Serversysteme werden bereits Wasserkühlungen angeboten [7]. Tatsächlich ist auch die Verlustleistung zum entscheidenden Kostenfaktor (Kosten für Wärmeabfuhr sowie Energiekosten) bei großen Internet Service Providern geworden [9].

Bei Betrachtung der je Flächeneinheit auftretenden Verlustleistungen (Abb. 1.2) lassen sich Mikroprozessoren der kommenden Generationen - Extrapolation der

[1] Definition „Embedded System“ nach [1]: *„A computer system that is part of a larger system and performs some of the requirements of that system; for example, a computer system used in an aircraft or rapid transport system.“*

[2] Unter einem Integrierten Sensorsystem versteht man die Gesamtheit aus Sensorelement, zugehöriger Auswerteelektronik sowie der dafür notwendigen Softwarekomponenten. Integrierte Sensorsysteme sind damit Embedded Systeme, die Sensorsignale verarbeiten, umwandeln und zu entsprechenden Aktorsignalen verknüpfen.

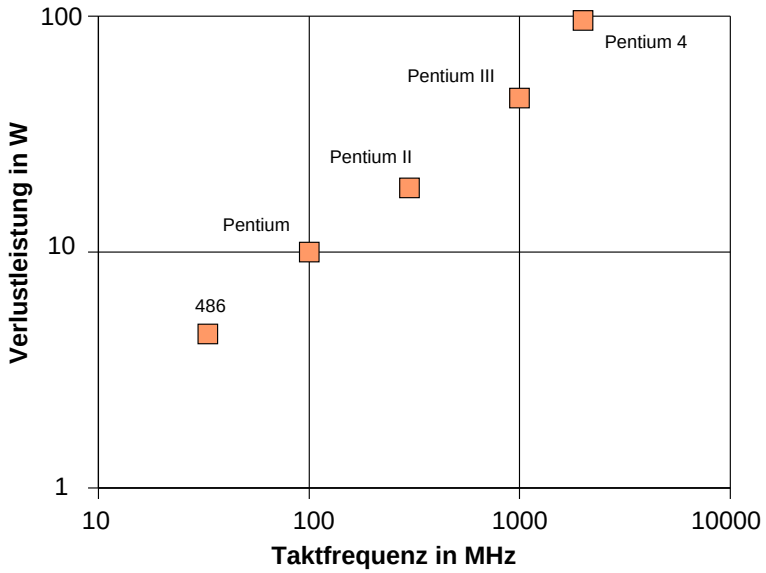


Abbildung 1.1: *Rechengeschwindigkeit und Leistungsverbrauch für Mikroprozessoren der Fa. Intel.* [80] Steigende Taktfrequenzen und Komplexität verursachen zusehends Probleme mit hohen Verlustleistungen.

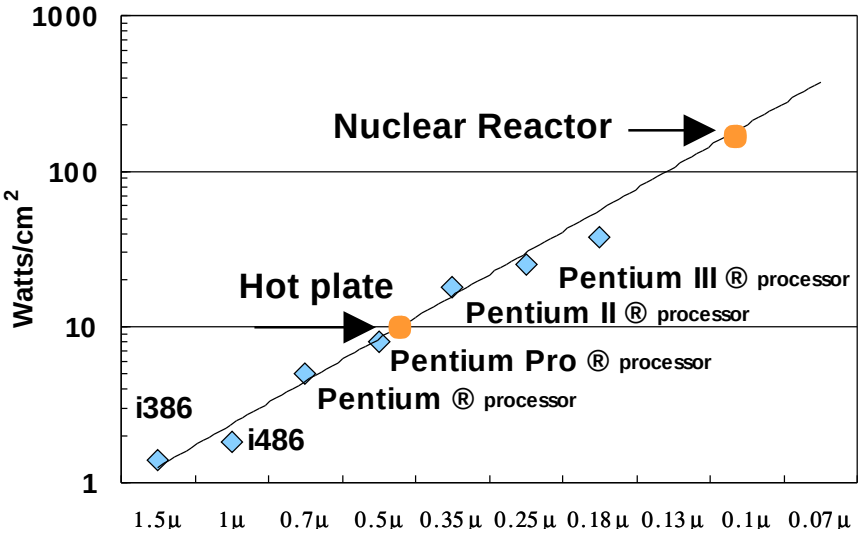


Abbildung 1.2: *Verlustleistung je Flächeneinheit in Abhängigkeit von der Strukturgröße für Prozessoren der Fa. Intel aus [8].* Die Mikroprozessoren der kommenden Generationen werden je Flächeneinheit ähnlich viel Wärme erzeugen, wie in den Brennstäben von Kernkraftwerken bezogen auf deren Oberfläche entsteht. Der eingezeichnete Wert für den nuklearen Reaktor ergibt sich aus einer thermischen nuklearen Leistungsdichte von 440W/cm^3 im Brennmaterial bezogen auf die Mantelfläche des Brennstabes mit 12mm Durchmesser [10, 11].

derzeitigen Entwicklung vorausgesetzt - durchaus mit der Oberfläche von Brennstäben in Kernreaktoren vergleichen [8]. Mikroskopisch betrachtet ist die in einem integrierten Schaltkreis auftretende Verlustleistung nicht gleichmäßig über die Chipfläche verteilt, sondern konzentriert sich an Stellen hoher Stromdichte, die mit speziellen Messverfahren ausfindig gemacht werden können [12].

Das Problem der Wärmeabfuhr wird derzeit bei Arbeitsplatzrechnern durch den Einsatz von Prozessorlüftern gelöst. Mit Ausnahme des höheren Lärmpegels ergeben sich dadurch keine wesentliche Beeinträchtigungen. Für Computersysteme in der Automatisierungstechnik sind jedoch eine Reihe von Einflussfaktoren bei der Bewältigung des Wärmeabfuhrproblems zu berücksichtigen:

- teilweise hohe Umgebungstemperaturen
- unzulässige Verkürzung der Lebensdauer durch Temperaturzyklen hoher Amplitude
- mechanische Lüfter erhöhen die Fehleranfälligkeit und beeinträchtigen die Zuverlässigkeit
- störende Einflüsse durch Erwärmung (Schaltschrank, Temperaturdrift, Verschlechterung von Halbleitereigenschaften)
- stärkere Verschmutzung durch erzwungene Lüfterkonvektion
- der Platzbedarf von Kühlern oder Lüftern liegt um Größenordnungen über dem der gekühlten Komponenten
- Zufuhr hoher Ströme ist bei miniaturisierten Komponenten problematisch (Steckkontakte, Leiterquerschnitte auf Leiterplatten)

Anforderungen der Integrierten Sensorik

In der Integrierten Sensorik spielt die Verlustleistung und die damit verbundenen Probleme der Wärmeabfuhr eine mitentscheidende Rolle beim Entwurf solcher Systeme. Eine möglichst weitreichende Reduktion des Leistungsverbrauchs vermeidet derartige Probleme von vorne herein und ist daher stets notwendig. Die Einschränkung von Verlustleistung stößt allerdings auf Probleme, falls hohe Verarbeitungsgeschwindigkeiten im Sensorsystem erforderlich sind.

Hohe Rechenleistung sind in der Integrierten Sensorik vor allem beim Einsatz von Sensoren mit hohen Datenübertragungsraten (z. B. in der Industriellen Bildverarbeitung) erforderlich, wo Datenströme in der Größenordnung von 100Mbyte/s zu bewältigen sind. Darüber hinaus gibt es eine Vielzahl von Problemstellungen, die nur durch sehr rechenintensive und komplexe Verarbeitungsalgorithmen gelöst werden können (z. B. Codierung und Kompression digitaler Daten, Erkennen von kontinuierlicher Sprache, Erkennen von Objekten und Mustern).

Neben hoher Rechenleistungen ist ein modernes Sensorsystem auch mit einer Reihe von zusätzlichen Anforderungen konfrontiert:

- kleine Bauform
- geringe Stromaufnahme
- hoher Grad an „Intelligenz“
- weitgehend autonomer Betrieb
- Benutzerfreundlichkeit
- Diagnosefunktionen, Unterstützung bei Konfiguration und Wartung
- hervorragendes Preis/Leistungsverhältnis

Zielkonflikt bei mobilen Sensorsystemen

Zunehmend mehr Sensorsysteme werden in mobilen Geräten eingesetzt bzw. an Batterien/Akkumulatoren betrieben (Digitalkameras, Mobile Roboter, sprachgesteuerte Systeme, Überwachungs- und Sicherheitssysteme, Verkehrstechnik, Stationen mit Solarbetrieb, etc.). Solche Sensorsysteme erfordern minimale Verlustleistungen um die Batterien zu schonen bzw. das Gerät nicht unzulässig hoch zu erwärmen, erfordern allerdings auch wegen ihrer Komplexität hohe Rechenleistungen. Weiters sind Ladezyklen und Betriebsdauer mitentscheidende Kriterien für die Akzeptanz solcher mobiler Sensorsysteme.

Die Mobiltelefon-Industrie ist seit Jahren mit diesem Zielkonflikt konfrontiert. Die fortlaufende Miniaturisierung in diesem Bereich und die gleichzeitig steigende Funktionalität waren ein Motor für das Vorantreiben innovativer Schaltungs- und Gehäusetechnik in der Mikroelektronik. Die zuletzt stattgefundene Integration von bildgebenden Sensoren und die Erweiterung der Übertragungsbandbreite durch UMTS machen Mobiltelefone zu komplexen Embedded Systemen.

Die Forderung nach sehr hohen Rechenleistungen

Ungelöst ist bisher das Problem, sehr hohe Rechenleistungen in mobilen Integrierten Sensorsystemen zu implementieren. Kompakte Bauform, geringe Stromaufnahme und hohe Rechenleistung sind gegenläufige Anforderungen, die bisher nur unbefriedigende Kompromisslösungen erlaubt haben. Abb. 1.3 veranschaulicht diese Problematik.

Obwohl zusehends damit konfrontiert konnte man diesem Problem im Bereich der Mobiltelefonie bisher aus dem Weg gehen, weil einerseits keine berechnungsintensiven Aufgaben zu bewältigen waren und andererseits die Rechenleistung auf

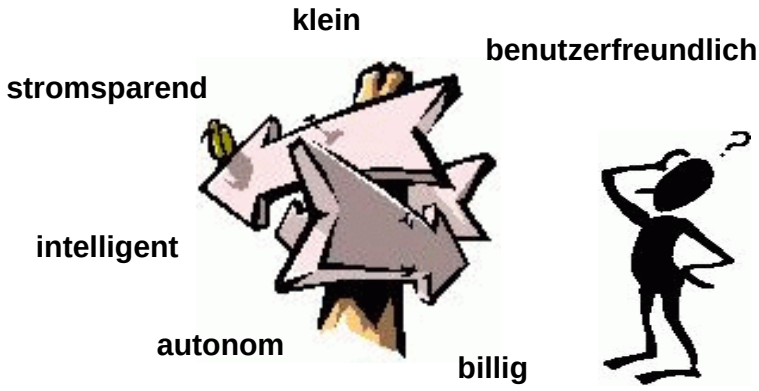


Abbildung 1.3: *Ziele bei mobilen Sensorsystemen.* Die Anforderungen an ein mobiles Integriertes Sensorsystem sind gegenläufig und provozieren unbefriedigende Kompromisse.

mehrere kleinere Einheiten verteilt werden konnte (z.B. Digitaler Signalprozessor für die Sprachcodierung).

Durch den anhaltenden Trend zunehmender Komplexität und die ständig steigenden Anforderungen an Integrierte Sensorsysteme werden sehr hohe Rechengeschwindigkeiten in künftigen Mikroprozessorsystemen bei anhaltender Miniaturisierung und Reduktion der Stromaufnahme unbedingt notwendige Voraussetzungen sein.

Die Lösung des Zielkonflikts zwischen Rechenleistung, Stromaufnahme und Miniaturisierung ist Gegenstand der vorliegenden Dissertation.

Kapitel 2

Aufgabenstellung

Bei Analyse der vorherrschenden Trends zu immer höheren Rechengeschwindigkeiten zeigt sich ein auffallender Unterschied im Trendverlauf zwischen Mikroprozessoren für Arbeitsplatzrechner („Desktops“) und Mikroprozessoren für Embedded Systeme (Abb. 2.1).

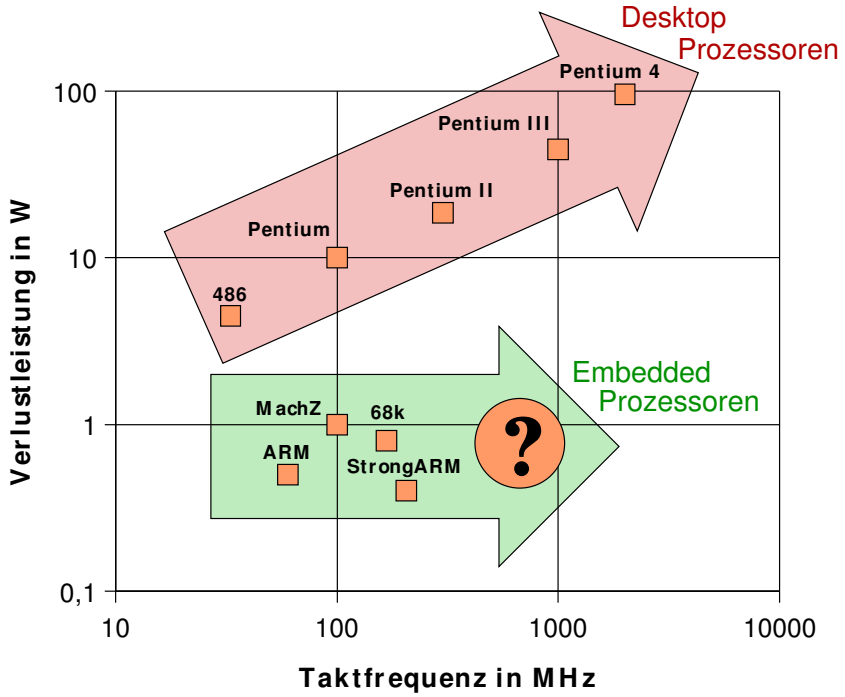


Abbildung 2.1: *Trends bei Mikroprozessoren.* Bei Desktop-Prozessoren werden für höhere Rechengeschwindigkeiten auch hohe Verlustleistungen in Kauf genommen. Embedded-Prozessoren befinden sich hingegen im Low-Power Bereich von etwa 1W.

Während für Desktop-Prozessoren bei höheren Taktfrequenzen auch bedeutend höhere Verlustleistungen in Kauf genommen werden, verläuft die Trendlinie der Embedded-Prozessoren waagrecht im Bereich geringer Leistungsaufnahmen von etwa 1W. Ein weiterer wesentlicher Unterschied ist in den erreichbaren Taktfrequenzen abzulesen, wo Desktop-Prozessoren eine Größenordnung schneller sind.

Im Rahmen der vorliegenden Dissertation soll der Trend zu höheren Rechengeschwindigkeiten bei gleichzeitig geringen Verlustleistungen weiterentwickelt werden. Dies ist in Abb. 2.1 durch den mit einem Fragezeichen versehenen Bereich angedeutet.

Die Aufgabenstellung besteht im Entwurf und der Realisierung eines hochperformanten Mikroprozessormoduls mit möglichst geringer Stromaufnahme und minimalem Platzbedarf für berechnungsintensive Aufgaben der Integrierten Sensorik. Die Verarbeitungsgeschwindigkeit des Prozessors soll mindestens $M = 500$MIPS (Million Instructions per Second) betragen, die gesamte maximale Leistungsaufnahme des Mikroprozessormoduls soll $P = 2W$ nicht übersteigen. Für eine einfache Integration des Mikroprozessormoduls in Integrierte Sensorsysteme soll das geometrische Format einer Scheckkarte angestrebt werden.

Weiters sollte das Mikroprozessormodul folgende Komponenten beinhalten:

- einen RISC-Mikroprozessor oder Digitalen Signalprozessor
- 64Mbyte flüchtigen Speicher (RAM bzw. SDRAM) als Arbeitsspeicher
- 64Mbyte nichtflüchtigen Speicher (ROM bzw. Flash-Speicher) für das Ablegen von Programmen, Konfigurationsdaten und Betriebssystem
- einen programmierbaren Logikbaustein zur Anbindung des Mikroprozessormoduls an Sensoreinheiten und zur Verschaltung der Komponenten untereinander
- Schnittstellen für die Übertragung von Berechnungsergebnissen und Diagnoseinformationen

Die Tab. 2.1 fasst die Randbedingungen für die Entwicklung des neuen Mikroprozessormoduls zusammen. Diese Kennwerte stellen teils gegenläufige Anforderungen dar und zwingen zu Kompromisslösungen. Es sind daher Methoden vorzuschlagen und zu entwickeln, die den Zielkonflikt zwischen den geforderten Kenngrößen Rechenleistung, Stromaufnahme und Baugröße optimal lösen.

Für die Realisierung des Prozessormoduls erhebt sich die Frage, mit welchen Komponenten sich ein derartiges System aufbauen läßt. Es sind daher Bausteine auszuwählen, die im Hinblick auf die geforderten Randbedingungen optimal sind.

Da die Stromaufnahme eines Mikroprozessorsystems wesentlich von den Eigenschaften der ablaufenden Programme und des Betriebssystems mitbestimmt wird,

Prozessortyp	RISC/DSP
Rechenleistung	$\geq$500MIPS
Leistungsaufnahme	$\leq$2W
Flash-Speicher	$\geq$64Mbyte
RAM-Speicher	$\geq$64Mbyte
Baugröße	$\leq$50cm^2 (Scheckkarte)

Tabelle 2.1: *Randbedingungen für die Entwicklung eines neuen Mikroprozessormoduls.*

ist zu untersuchen, welche softwaretechnischen Maßnahmen zur Reduktion des Leistungsverbrauches ergriffen werden können.

Für den Betrieb des Mikroprozessormoduls ist ein Betriebssystem auszuwählen, das für den Einsatz in einem Low Power Mikroprozessorsystem geeignet ist.

Für die Entwicklung von Software für das neue Mikroprozessormodul ist eine Entwicklungsumgebung auszuwählen bzw. bereitzustellen.

Kapitel 3

Stand der Technik

Für die Entwicklung eines neuen Mikroprozessormoduls ist der Stand der Technik in zweierlei Hinsicht interessant.

Einerseits sind die in der Aufgabenstellung geforderten Merkmale des Mikroprozessormoduls den derzeit am Markt befindlichen Geräten bzw. Komponenten gegenüberzustellen. Dazu ist es notwendig, jene Messgrößen zu definieren, anhand derer eine Gegenüberstellung erfolgen kann.

Andererseits ist die Entwicklung des neuen Mikroprozessormoduls geprägt von Methoden des Low Power Designs. Aus diesem Grund ist die Erhebung derartiger Methoden und Verfahren sowie des einschlägigen Literaturstatus in diesem Bereich von besonderem Interesse.

3.1 Messgrößen

Um die Aufgabenstellung aus Kapitel 2 dem Stand der Technik gegenüberzustellen bzw. einen Vergleich von hochperformanten Low Power Systemen untereinander durchzuführen, ist es notwendig, geeignete Messgrößen, Kennzahlen und Beurteilungskriterien heranzuziehen. Dabei sind in erster Linie Rechenleistung und Leistungsverbrauch Gegenstand dieses Vergleiches.

Folgende Messgrößen sind Stand der Technik und werden im Rahmen dieser Arbeit verwendet:

Speicherplatz Die Höhe des verfügbaren Speicherplatzes ist eine wichtige Kennzahl für die Einschätzung der Anwendungsmöglichkeiten für ein Computersystem, da sowohl Programme und Betriebssystem als auch sämtliche Daten, die während des Betriebes anfallen, im Speicher Platz finden müssen. Besonders speicherintensiv sind Anwendungen mit komplexer Algorithmik und hohem Datenaufkommen, wie z. B. Bildverarbeitungssysteme.

Geometrische Abmessungen Um ein Mikroprozessormodul in bestehende Sensor-/Aktorsysteme integrieren zu können, ist es notwendig, das Modul entsprechend klein aufzubauen. Das beschränkte Platzangebot beeinflusst den gesamten Designprozess nachhaltig und provoziert die Auswahl von kleinsten Hardwarebauteilen, die mitunter sehr kostspielig sind. Die geometrischen Abmessungen sind damit ein entscheidender Faktor.

Taktfrequenz Die einfachste Beurteilungsmöglichkeit für die Rechenleistung eines Mikroprozessors bietet seine Taktfrequenz. Da moderne Mikroprozessoren überwiegend RISC (Reduced Instruction Set Computer) Architekturen besitzen, kann man bei derartigen Prozessoren damit rechnen, dass pro Taktzyklus ein Maschinenbefehl abgearbeitet wird, wodurch ein Performancevergleich über die Taktfrequenz überhaupt erst ermöglicht wird. Erfüllen Prozessoren diese Voraussetzung nicht (z. B. bei einer CISC (Complex Instruction Set Computer) Architektur oder mehreren parallelen Verarbeitungseinheiten), ist die Bedeutung der Taktfrequenz für eine Performancebeurteilung zu relativieren.

MIPS (Million Instructions Per Second) Diese Messgröße M wird für die Beurteilung und den Vergleich der Rechenleistung verschiedener Prozessoren sehr häufig verwendet. Sie gibt die Anzahl von Maschineninstruktionen an, die ein Mikroprozessor je Sekunde abarbeiten kann [14, 15]. Da M sehr stark von der zugrundeliegenden Prozessorarchitektur (Pipeline, Superscalar, parallele Architekturen, etc.) abhängt, können sich die Werte für M von zwei unterschiedlichen Prozessoren trotz identischer Taktfrequenz stark unterscheiden.

Es ist daher M stets im Zusammenhang mit den architektonischen Eigenschaften eines Mikroprozessors zu bewerten.

Verlustleistung Für den Vergleich zwischen verschiedenen Mikroprozessoren ist der absolute Leistungsverbrauch nicht aussagekräftig genug. Vielmehr ist der Bezug zur Verarbeitungsgeschwindigkeit maßgeblich. Es wird daher als Kenngröße der auf die Performancegröße M bezogene Leistungsverbrauch

$$p = \frac{P}{M} \tag{3.1}$$

in mW/MIPS herangezogen.

Energy-Delay Produkt Für die Bemessung der Energieeffizienz einer hochperformanten mikroelektronischen Schaltungsstruktur ist das sog. Energy-Delay Produkt gebräuchlich [16]. Es verknüpft den erforderlichen Energiebedarf und die notwendige Zeitdauer um eine bestimmte Rechenoperation (z. B. Addition in einem Addierwerk) durchzuführen.

Dhrystone Der Dhrystone Benchmark misst die Ausführungsdauer einer genau festgelegten, synthetischen Programmsequenz [17]. Die Sequenz ist so zusammengestellt, dass sie einen typischen Mix aus Operationen darstellt, der aus einem großen Satz von untersuchten Programmen ermittelt wurde. Die Anzahl der ausgeführten Sequenzen je Sekunde wird als sog. „Dhrystones“ bezeichnet. Dividiert man diesen Wert durch 1757 (das ist die Anzahl von Dhrystones auf einem VAX 11/780 Computer, welche als Normierung herangezogen wurde), so erhält man die sog. „Dhrystone MIPS“ [18].

SPEC CPU Ähnlich wie bei Dhrystone wird die Ausführungsdauer von Programmsequenzen gemessen mit dem Unterschied, dass es sich hierbei um einen festgelegten Satz von typischen Anwendungsprogrammen handelt [20]. Anders als bei Dhrystone, wo nur eine kurze synthetische Programmsequenz abgearbeitet wird, kommt bei SPEC CPU eine ganze Reihe von realen Programmen zur Ausführung, was zu einer repräsentativen, allgemeinen Performancegröße führt.

Sonstige Benchmarks Durch die laufende Weiterentwicklung und Diversifizierung moderner Mikroprozessoren entstehen im Laufe der Zeit immer wieder neue Benchmark-Methoden. Im Bereich der Embedded Systeme geht die Fa. Motorola und das Embedded Microprocessor Benchmark Consortium neue Wege in der Beurteilung von Performance, indem für wichtige Marktsegmente (Consumerelectronic, Telekom, Netzwerk, Automobil und Büro) branchenspezifische Benchmarks definiert werden [21, 22].

3.2 Kommerziell verfügbare Mikroprozessormodule und Geräte

Bei der Entwicklung von digitalen elektronischen Geräten und Systemen erhebt sich während des konzeptionellen Entwurfs stets die Frage nach einer geeigneten Recheneinheit, die den gestellten Anforderungen an Verarbeitungsgeschwindigkeit, Speicherplatz, Platzbedarf, Leistungsverbrauch, etc. genügt. Der Markt stellt hierfür eine schier unüberschaubare Fülle an Mikroprozessormodulen unterschiedlichster Leistungsklasse zur Verfügung, die eine Alternative zu Eigenentwicklungen sind.

Im Rahmen der vorliegenden Arbeit wurde eine Marktrecherche durchgeführt. Sie erstreckte sich auf Mikroprozessormodule mit einer Taktfrequenz zwischen 100MHz und 1GHz, einer Geometrie von weniger als 150cm^2 und Verlustleistungen von weniger als 25W, um von vorne herein die Vielfalt der Möglichkeiten zu Reduzieren und für die Aufgabenstellung aus Kapitel 2 ungeeignete Lösungen a priori auszuscheiden.

Zur Darstellung der Marktsituation wurde eine Einteilung in die folgenden Kategorien von Mikroprozessormodulen getroffen:

- Industrie-PCs
- PC-kompatible Single Board Computer (SBC)
- nicht PC-kompatible Mikroprozessormodule
- Mikroprozessormodule für Taschencomputer (sog. Handhelds oder PDAs)

Aus jeder dieser Kategorien wird ein typischer, im Sinne der Aufgabenstellung leistungsfähiger Stellvertreter entnommen und anhand der in Kapitel 3.1 dargestellten Messgrößen gegenübergestellt. In diesen Vergleich werden auch die geforderten Randbedingungen aus der Aufgabenstellung miteinbezogen.

3.2.1 Industrie-PCs

Industrie-PCs haben den enormen Vorteil, dass durch die Kompatibilität zu Arbeitsplatz-PCs ausgereifte und weit verbreitete Entwicklungswerkzeuge zur Verfügung stehen. Weiters können Industrie-PCs unmittelbar von der raschen Weiterentwicklung im PC-Sektor profitieren. Nachteilig wirkt sich allerdings die PC-Kompatibilität auf die Verlustleistung (vgl. Abb. 1.1) aus.

Ein Vertreter aus der Familie der Industrie-PCs ist z. B. „Razor Blade“ von der Fa. PFU Systems [23]. Es handelt sich hier um einen Pentium III Prozessor mit 700MHz Taktfrequenz. „Razor Blade“ besitzt kleine Abmessungen (ca. 14x8x1cm) und die bei PCs üblichen Schnittstellen [24]. Die Abb. 3.1 zeigt Vorder- und Rückseite sowie das Seitenprofil von „Razor Blade“.

3.2.2 PC-kompatible Single Board Computer

Neben den Industrie-PCs sind zur Zeit immer mehr PC-kompatible Alternativlösungen am Markt erhältlich [83, 90, 91, 92]. Durch die PC-Kompatibilität bleiben die Vorteile der umfangreichen und ausgereiften PC-Entwicklungsumgebungen erhalten. Andererseits ermöglicht die Abkehr von der PC-Architektur auch die Eliminierung von entscheidenden Nachteilen, wie z. B. der hohen Verlustleistung.

Die Abb. 3.2 zeigt einen Vertreter dieser Kategorie von Mikroprozessormodulen. Der „HS1600“ ist ein ein sog. Single Board Computer (SBC) mit einem 400MHz Transmeta Crusoe Prozessor inklusive einer Reihe von Ein-/Ausgabebausteinen für Netzwerk, Audio, USB und VGA auf einer einzigen 10x11cm großen Platine [25, 26].

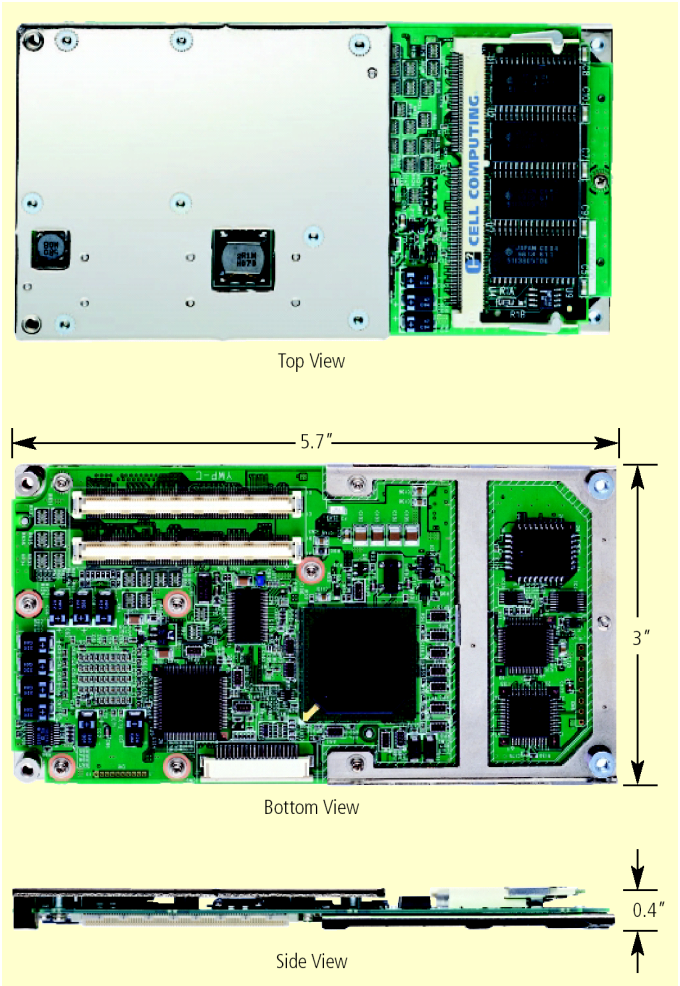


Abbildung 3.1: *„Razor Blade" Industrie-PC der Fa. PFU Systems.* [24] Dieses Produkt bietet hohe Rechenleistung auf einer Fläche von etwa 14x8cm. Über das Metallgehäuse wird die Verlustwärme des 700MHz Pentium III Prozessors abgeführt.

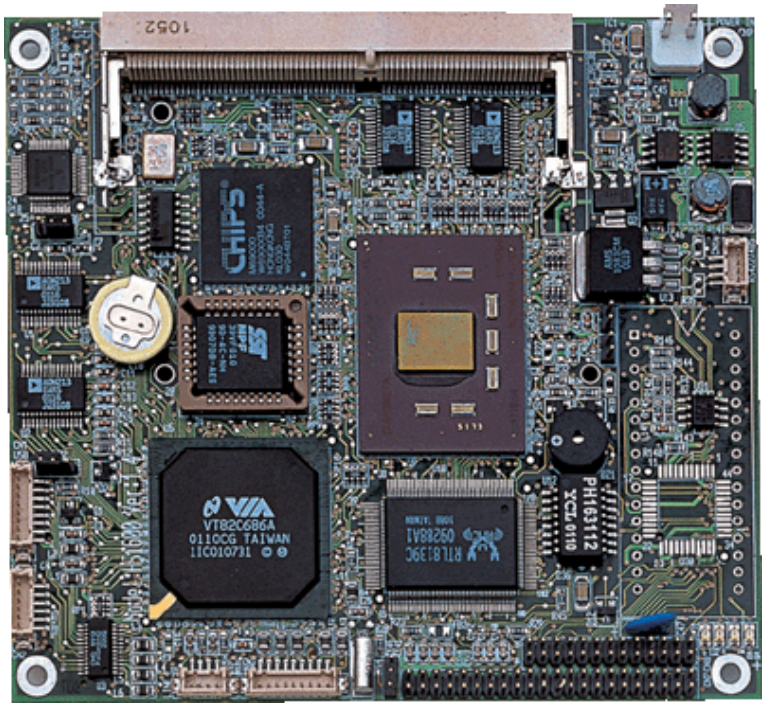


Abbildung 3.2: *„HS1600" Single Board Computer der Fa. Boser Technology [25].* Das 10x11cm große Board verfügt über einen 400MHz Transmeta Crusoe Prozessor, VGA-Grafikchip, USB und einen 100Mbit Netzwerkcontroller.

3.2.3 Nicht PC-kompatible Mikroprozessormodule

Neben den PC-orientierten Produkten existiert eine Vielzahl von Mikroprozessormodulen, die völlig andere Architekturen aufweisen. Weite Verbreitung haben die Familien der ARM-, PowerPC- und 68k-Prozessoren gefunden. Sie werden in erster Linie in Embedded Systemen eingesetzt, die stark applikationsbezogen aufgebaut sind und wo eine allgemeine Verwendbarkeit eine untergeordnete Rolle spielt.

In Abb. 3.3 ist ein Mikroprozessormodul dargestellt, das eine Abmessung von 6.7x3.7cm aufweist und mit einem Prozessor der ARM-Familie (Intel StrongARM) ausgestattet ist [27, 28]. Der Prozessor wird mit 206MHz getaktet und benötigt inklusive der Speicherbausteine im Volllastbetrieb nur 1.2W. Es darf allerdings nicht außer Acht gelassen werden, dass zusätzlich zu diesem Modul gezwungenermaßen noch Ein-/Ausgabebausteine in einem realen Mikroprozessorsystem notwendig sind, die in die Energiebilanz miteinbezogen werden müssen.

3.2.4 Mikroprozessormodule für Taschencomputer

Aufgrund ihrer mobilen und damit gleichzeitig energiesparenden Eigenschaften sind Mikroprozessormodule für tragbare Taschencomputer im Zusammenhang mit

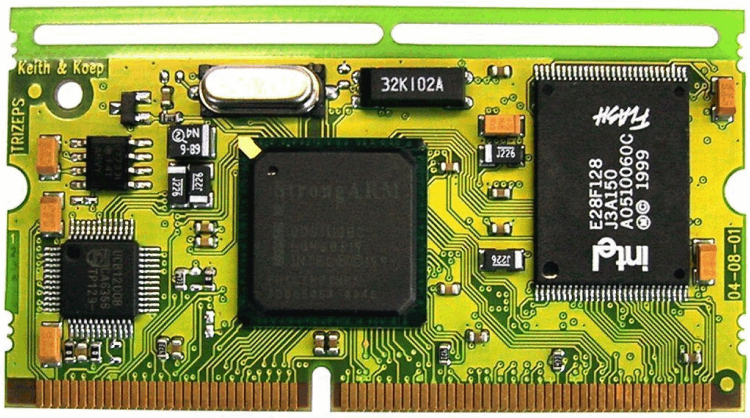


Abbildung 3.3: *„Trizeps" StrongARM Mikroprozessormodul der Fa. Keith & Koep GmbH [27].* Das 6.7x3.7cm große Board verfügt über einen 206MHz Intel StrongARM Prozessor, 4Mbyte Flash-Speicher und 16Mbyte SDRAM. Das Modul zeichnet sich durch einen geringen Leistungsverbrauch von nur etwa 1.2W aus.

dieser Arbeit besonders interessant. Sie werden deshalb zu einer eigenen Kategorie zusammengefasst.

Allen Mikroprozessormodulen dieser Kategorie gemeinsam ist ihre kompakte Bauform und geringe Stromaufnahme. Je nach Herstellerfirma werden unterschiedliche Prozessorfamilien in den Gräten eingesetzt: StrongARM bei HP/Compaq [79], Motorola 68k bei Palm [93] oder SH bei Hitachi [88].

In Abb. 3.4 ist der Taschencomputer iPAQ 3838 der Fa. Compaq dargestellt [29]. Die Hauptplatine dieses Gerätes ist mit einem 206MHz StrongARM Prozessor, 32Mbyte Flash- und 64Mbyte RAM-Speicher ausgestattet.

3.2.5 Gegenüberstellung

Zum Vergleich der in den Abschnitten 3.2.1 bis 3.2.4 vorgestellten Mikroprozessormodule sind in der Tab. 3.1 technische Daten und Messgrößen dargestellt. Anhand dieser Tab. 3.1 verdeutlicht sich wiederum der bereits in Kapitel 1 erwähnte Zielkonflikt zwischen Rechenleistung, Stromaufnahme und Baugröße. So ist beispielsweise „Razor Blade" um den Faktor 7.8 performanter, benötigt allerdings im Gegenzug 15 mal mehr Leistung als „iPAQ". Für die Speicherausstattung gilt ähnliches: „Razor Blade" und „HS1600" bieten zwar 256Mbyte Speicher, besitzen allerdings größere geometrische Abmessungen.

Um den Bezug zur Aufgabenstellung zu erhalten, sind in der Tab. 3.1 auch die im Rahmen dieser Arbeit geforderten Randbedingungen aufgelistet. Man erkennt, dass die am Markt befindlichen Mikroprozessormodule die gestellten Anforderungen zwar in einzelnen Punkten, nicht aber zur Gänze erfüllen können.

Abbildung 3.4: *„iPAQ 3835" Taschencomputer der Fa. Compaq [29].* Der 8x13cm große Taschencomputer verfügt über einen 206MHz Intel StrongARM Prozessor, 32Mbyte Flash-Speicher und 64Mbyte SDRAM.

Produkt	Razor Blade	HS1600	Trizeps	iPAQ	Aufgabe
Prozessortyp	Pentium III	Crusoe	StrongARM	StrongARM	
Taktfrequenz in MHz	700	400	206	206	
Dhrystone MIPS	1843	729	235	235	$\geq$500
Flash-Speicher in Mbyte	0	0	16	32	$\geq$64
RAM-Speicher in Mbyte	256	256	64	64	$\geq$64
Fläche in cm^2	110	115	25	112	$\leq$50
Verlustleistung in W	22.9	6	1.2	1.5	$\leq$2

Tabelle 3.1: *Gegenüberstellung von Mikroprozessormodulen.* Die in den Abschnitten 3.2.1 bis 3.2.4 vorgestellten Mikroprozessormodule werden in dieser Tabelle den aus der Aufgabenstellung vorgegebenen Randbedingungen gegenübergestellt. Dargestellt sind Taktfrequenz, Rechenleistung in Dhrystone MIPS, Speichergrößen, Fläche und Leistungsbedarf. Anhand dieser Messgrößen verdeutlicht sich der Zielkonflikt zwischen Rechenleistung, Verlustleistung und Baugröße.

3.3 Einflussfaktoren der Verlustleistung in Integrierten Schaltungen

In integrierten Mikroprozessorsystemen ohne mechanische Komponenten (wie z. B. Festplatten), Energiespeicher und Wellenabstrahlung wird unter Vernachlässigung der Stromwärmeverluste in der Leiterplatte und den diskreten Bauelementen die gesamte Verlustleistung durch integrierte Schaltkreise hervorgerufen. Diese sind heute in der Mikroprozessortechnik fast ausschließlich in CMOS-Technologie ausgeführt.

Um einen gezielten Entwurf von Low Power Mikroprozessorsystemen zu ermöglichen, ist das Verständnis von verlustleistungsbehafteten Vorgängen in integrierten CMOS-Schaltungen notwendig. In der Literatur (z. B. [30]) ist das folgende Verlustleistungsmodell für CMOS-Schaltkreise gebräuchlich:[1]

$$P = \alpha f C U^2 + \tau \alpha f U I_k + U I_0 \tag{3.2}$$

Die Gl. (3.2) beschreibt die gesamte Verlustleistung P eines CMOS Schaltkreises. Sie besteht aus drei additiven Termen. Der erste Term $\alpha f C U^2$ beschreibt die Verlustleistung, die durch das Laden und Entladen der an den Ausgängen von CMOS-Gattern wirkenden kapazitiven Lasten entsteht. Sie ist proportional zu der an den Ausgängen wirkenden Gesamtkapazität C, der System-Taktfrequenz f, der Gatteraktivität α (nicht alle Gatter schalten in jedem Taktschritt) und quadratisch proportional zur Versorgungspannung U.

Der zweite Term $\tau \alpha f U I_k$ beschreibt jene Verlustleistung, die während des Umschaltens eines CMOS-Gatters durch den kurzzeitig im Gatter fließenden Kurzschlussstrom I_k verursacht wird. I_k wird durch den komplementären Aufbau eines CMOS-Gatters hervorgerufen und fließt während der Zeit τ zwischen der Versorgungsspannung U und dem Masseanschluss. Die Anzahl der Umschaltvorgänge ist wiederum durch das Produkt αf berücksichtigt.

Der dritte Term $U I_0$ beschreibt die statischen Verluste, die durch die ständig (auch im Ruhezustand) fließenden Leckströme I_0 verursacht werden.

In modernen hochintegrierten CMOS Bauelementen dominiert der erste Term in Gl. (3.2) [30]. Daher ist naheliegend, dass sich eine Reduktion der Versorgungsspannung U in einer signifikanten Verbesserung im Leistungsverbrauch niederschlägt. Trotz Vervierfachung der Taktfrequenz f (und damit der Verarbeitungsgeschwindigkeit) könnte bei Halbierung der Versorgungsspannung die Verlustleistung etwa konstant bleiben. Die Verringerung der Versorgungsspannung wirkt sich allerdings auch negativ auf die Schaltgeschwindigkeit der MOS-Transistoren

[1] Da es sich um ein Modell [31] handelt, beschreibt Gl. (3.2) die Verlustleistung nicht exakt. Trotzdem erlaubt diese Modellierung die Berechnung und Erklärung aller wesentlichen Verlustleistungsprozesse in CMOS-Schaltkreisen.

aus, wodurch sich die maximale Schaltfrequenz f_{max} des MOS-Transistors proportional zu

$$f_{max} \sim \frac{(U - U_{th})^2}{U} \tag{3.3}$$

verringert. U_{th} bedeutet dabei die technologisch festgelegte Schwellenspannung des MOS-Transistors [32]. Eine Verringerung der Versorgungsspannung begrenzt daher auch die maximale Taktfrequenz einer CMOS-Logikschaltung.

Aus der Gl. (3.3) erkennt man, dass bei einer Reduktion der Versorgungsspannung U auch eine Reduktion der Schwellenspannung U_{th} nötig ist, um eine ausreichend hohe Schaltfrequenz zu erhalten. U_{th} kann technologisch durch Dotierungsverhältnisse eingestellt werden, ist allerdings durch den Zusammenhang

$$I_0 \sim \exp\left(-\frac{qU_{th}}{kT}\right) \tag{3.4}$$

nach unten beschränkt, um den statischen Leckstrom I_0, der in exponentiellen Zusammenhang mit U_{th} steht, zu begrenzen. q bezeichnet dabei die Elementarladung und k die Boltzmannkonstante. Ruheströme machen bei derzeit handelsüblichen Prozessoren (z. B. Intel Pentium 4) bereits einen beachtlichen Teil der Stromaufnahme aus [33].

Das Modell der Gln. (3.2) bis (3.4) beinhaltet alle wesentlichen Zusammenhäge um die in CMOS Logikschaltungen auftretende Verlustleistung mathematisch zu erfassen. In der Abb. 3.5 werden diese Zusammenhänge grafisch veranschaulicht. Tatsächlich existiert noch eine Reihe weiterer, jedoch untergeordneter Einflussfaktoren vor allem auf technologischer Ebene (z. B. beeinflusst die Versorgungsspannung über die Transistor-Strukturgrößen auch die Kapazitäten in der CMOS Logikschaltung).

3.4 Methoden zur Verlustleistungsreduktion

In der Literatur sind zahlreiche Methoden zu finden, die eine Verringerung der Verlustleistung in CMOS Logikschaltungen anstreben. Eine sinnvolle Kategorisierung dieser Maßnahmen kann anhand der in Abb. 3.5 dargestellten Einflussfaktoren erfolgen:

- Reduktion der Versorgungsspannung
- Reduktion der Gatteraktivität
- Reduktion der Taktfrequenz

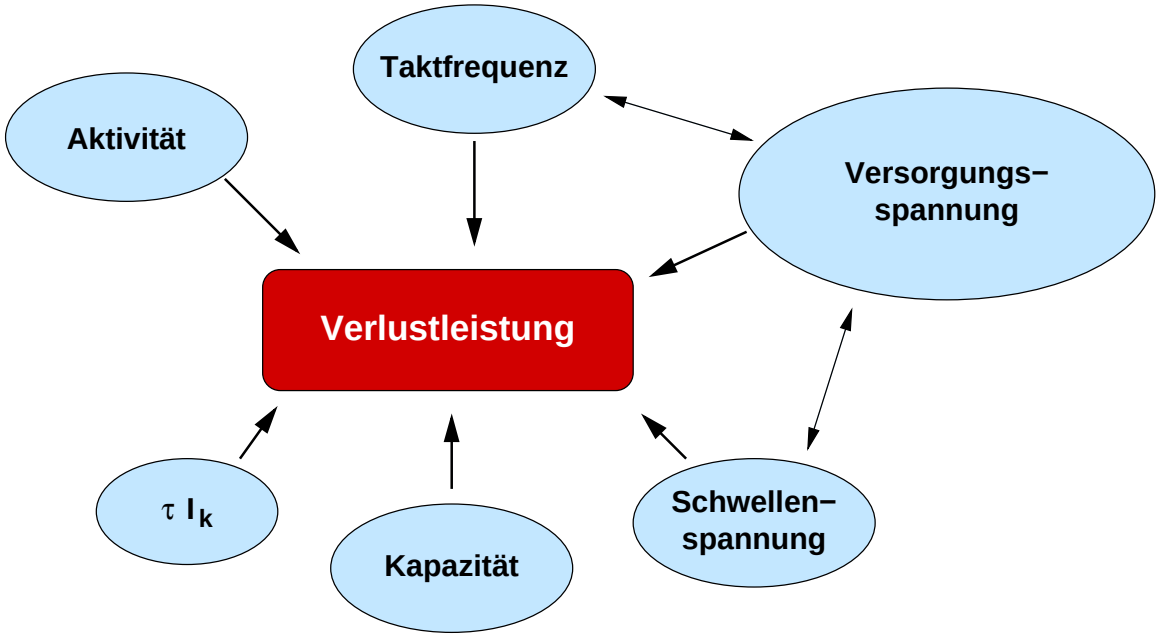


Abbildung 3.5: *Einflussfaktoren für die Verlustleistung in CMOS Logikschaltungen.* Dargestellt sind die wesentlichsten Faktoren gemäß Gln. (3.2) bis (3.4), die Einfluss auf die Verlustleistung in CMOS Logikschaltungen haben. Wichtigste Einflussgröße ist die Versorgungsspannung. Die einzelnen Faktoren stehen auch untereinander in Zusammenhang. Die Größe der Ellipsen stell ein Maß für den Einfluss auf die Verlustleistug dar.

- Technologische Maßnahmen

Der gezielte Eingriff über einzelne Einflussfaktoren ist wegen ihrer gegenseitigen Abhängigkeiten schwierig. In der Praxis finden daher Methoden Anwendung, die gleichzeitig mehrere Einflussfaktoren zur Verringerung der Verlustleistung heranziehen [34, 35, 36, 37, 38].

3.4.1 Reduktion der Versorgungsspannung

Durch die quadratische Abhängigkeit aus Gl. (3.2) ist eine Reduktion der Versorgungsspannung U eine besonders effektive Maßnahme zur Reduktion von Verlustleistung. Es muss jedoch bedacht werden, dass die Versorgungsspannung nicht unabhängig gewählt werden kann, sondern auch starken Einfluss auf die Schaltgeschwindigkeit und die Schwellenspannung der CMOS Transistoren hat.

Zumeist ist die Versorgungsspannung eines integrierten CMOS Schaltkreises fest vorgegeben und kann während des Betriebes nur innerhalb eines kleinen Toleranzbereiches verändert werden. Moderne Mikroprozessoren, die für den Einsatz in mobilen Geräten geeignet sind (z. B. [39, 47]) bieten hingegen die Möglichkeit selbst während des Betriebes die Spannung über einen größeren Bereich zu variieren. Der XScale-Prozessor der Fa. Intel kann z. B. mit einer Spannung zwischen 0.75V und 1.65V betrieben werden.

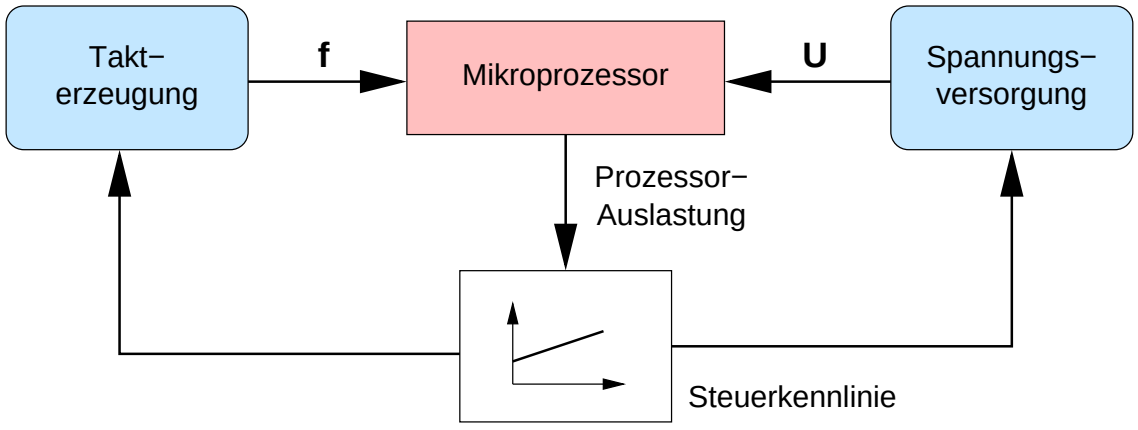


Abbildung 3.6: *Belastungsabhängige Spannungs- und Frequenzsteuerung.* Das Betriebssystem bestimmt aus der aktuellen Prozessorauslastung anhand einer Steuerkennlinie Sollwerte für die Taktfrequenz und Versorgungsspannung.

Die Abb. 3.6 zeigt die schematische Anordnung einer dynamischen Anpassung von Taktfrequenz und Versorgungsspannung während des Betriebes eines Mikropro-

zessors. Das Betriebssystem, welches für die dynamische Anpassung verantwortlich ist, ermittelt aus der aktuellen mittleren Auslastung des Prozessors einen Sollwert für die Taktfrequenz f und die Versorgungsspannung U. Die alleinige Reduktion der Taktfrequenz würde zwar gemäß Gl. (3.2) eine proportionale Verminderung der Verlustleistung bewirken, aber aufgrund der längeren Berechnungsdauer keinen Gewinn in der insgesamt für eine Berechnungsaufgabe benötigten Energiemenge einbringen. Erst die zusätzliche, gleichzeitige Verminderung der Versorgungsspannung macht dies möglich. Die alleinige Absenkung der Versorgungsspannung ist aufgrund der Gl. (3.3) bei gleichbleibender Taktfrequenz nicht zielführend, das dies wegen der längeren Schaltzeiten der Transistoren zu unvorgesehenen Prozessorfehlern führen kann.

Die Frage nach der theoretischen unteren Grenze für die Versorgungsspannung wird in [40, 41] diskutiert und beträgt etwa 83mV bei Raumtemperatur und minimal notwendigen Werten für die Transistorverstärkung und den Signal-Rauschabstand.

3.4.2 Verringerung der Gatteraktivität

Aus Gl. (3.2) und Abb. 3.5 ist zu erkennen, dass die Gatteraktivität α linear in die Verlustleistung einer CMOS Logikschaltung eingeht. Da α völlig unabhängig zu den anderen Einflussfaktoren ist, spielen die Methoden zur Verringerung von α in der Praxis eine sehr große Rolle. Entsprechend sind aus der Literatur auch zahlreiche Verfahren bekannt, die man in folgende Kategorien zusammenfassen kann

1. *Abschalten von nicht benötigten Schaltungsstrukturen.* In Mikroprozessoren werden innerhalb eines Taktschrittes nicht alle Logikfunktionen benötigt. Beispielsweise wird bei einer Addition von zwei Operanden das Multiplizierwerk nicht benötigt und kann somit vorübergehend abgeschaltet werden. Dies verhindert, dass innerhalb des Multiplizierwerkes Transistoren unnötig schalten. Bei der Entwicklung des StrongARM Prozessors hat ursprünglich die Fa. Digital und in der Folge die Fa. Intel stark von dieser Methode Gebrauch gemacht [42, 43, 44, 45, 46, 47] und damit einen Durchbruch im Bereich der Low Power Mikroprozessoren erzielt. In der Literatur ist diese Methode auch als „Gated Clock“ bekannt.

2. *Änderung der Mikroprozessorsystemarchitektur*. Betrachtet man die Speicherhierarchie eines Mikroprozessorsystems bestehend aus Prozessor, Memorycontroller, Cache und Hauptspeicher, so wird deutlich, dass durch den Einsatz von Caches eine erhebliche Reduktion (Richtwert 80%) der Zugriffe in den Hauptspeicher entsteht und insgesamt dadurch die Gatteraktivität reduziert werden kann. Als besonders vorteilhaft haben sich kleine, verlustleistungsarme Filtercaches erwiesen, die dem eigentlichen Cache vorangeschaltet sind und zusätzliche Einsparungen in der gesamten Verlustleistung

bringen [48, 49, 50]. Weitere Einsparungsmöglichkeiten ergeben sich z. B. durch Einsatz eines alternativen Zahlensystems [51]. Dies bedeutet zwar eine grundlegende Abänderung in der Architektur, kann aber für spezielle Berechnungsaufgaben sehr vorteilhaft sein. Interessant ist in diesem Zusammenhang auch die Berücksichtigung statistischer Eigenschaften der zu verarbeitenden digitalen Signale, die aufgrund ihrer statistischen Verteilung von logischen „0“ und „1“ Pegeln die Schaltungsaktivität stark beeinflussen können [65].

3. *Software*. Großen Einfluss auf die Gatteraktivität besitzt die in einem CMOS-basierenden Mikroprozessorsystem ablaufende Software. Je effizienter Programme abgearbeitet werden können, desto wegiger Schaltvorgänge sind in den Transistoren nötig. Große Bedeutung kommt hier dem Betriebssystem zu, dessen Aufgabe es ist, nicht benötigte Hardware abzuschalten und das komplette Mikroprozessorsystem während der Berechnungspausen in einen energiesparenden Zustand überzuführen (meist als „Idle“ oder „Sleep“-Modus bezeichnet). Durch gezielte Auslegung der Software kann darüber hinaus die Effizienz der Speicherhierarchie (Cache-Performance) gesteigert und somit die Gatteraktivität weiter reduziert werden [52, 53, 54, 55].

4. *Optimierung von Schaltungsstrukturen*. Durch Optimierung von schaltungstechnischen Strukturen [56] hinsichtlich der Gatteraktivität läßt sich die Verlustleistung verringern. In [57] werden 20 verschiedene Addierschaltungen anhand Verlustleistung und Rechengeschwindigkeit gegenübergestellt. Für den Einsatz in DSPs sind energiesparende Schaltungen für Filter und Codierwerke besonders interessant [58]. In [59] wird eine modifizierte Schaltungsstruktur für CMOS Gatter vorgestellt, die während des Umschaltvorganges eines CMOS Gatters Einsparungen in der Verlustleistung ermöglicht. Auch bei Caches und Speicher können durch gezielte Änderung der Schaltung Einsparungen gemacht werden [60].

5. *Einsatz anwendungsspezifischer Hardware*. Wenn ein Mikroprozessorsystem für einen speziellen Anwendungsfall konzipiert wird, können ansonsten allgemein einsetzbare Logikbausteine durch anwendungsspezifische und verlustleistungsoptimierte Schaltkreise ersetzt werden. Z. B. kann der Ersatz eines Mikroprozessors oder DSPs durch einen ASIC im speziellen Anwendungsfall eine um zwei Größenordnungen bessere Energieeffizienz einbringen [61, 62]. Auch die auf den jeweiligen Anwendungsfall angepasste Codierung von Speicheradressen, kann die Verlustleistung innerhalb eines Mikroprozessorsystems durch eine Reduktion der Gatteraktivität im Bereich von Adressbussen verringern [63, 64].

3.4.3 Reduktion der Taktfrequenz

Da die Taktfrequenz ausschlaggebend für die Performance eines Mikroprozessors ist, kommt eine Reduktion der Taktfrequenz nur dann in Frage, wenn der Mikroprozessor auch bei geringerer Geschwindigkeit seine Aufgaben in ausreichend kurzer Zeit bewältigen kann. Tatsächlich ist dies bei geringen Belastungszuständen des Mikroprozessors auch problemlos möglich (sofern der Mikroprozessor bei unterschiedlichen Frequenzen betrieben werden kann).

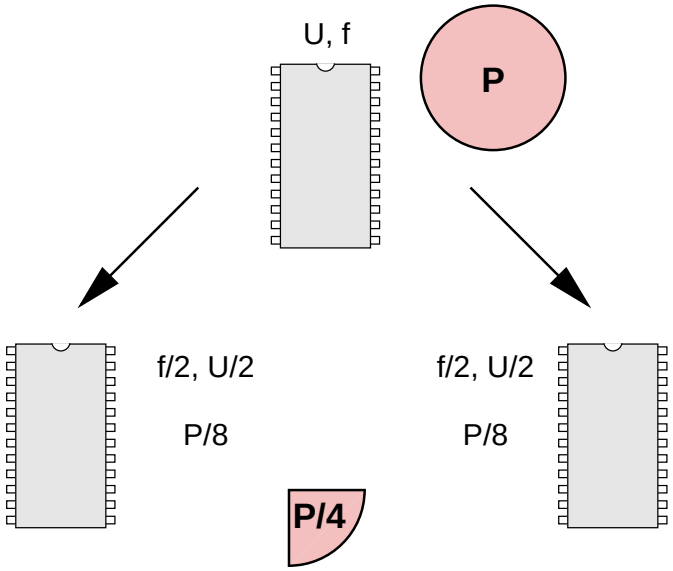


Abbildung 3.7: *Reduktion der Verlustleistung durch Parallelverarbeitung.* Durch Parallelverarbeitung kann bei gleicher Performance eine Reduktion der Verlustleistung auf ein 1/4 erreicht werden, da die Prozessoren im Parallelsystem mit geringerer Taktfrequenz und Versorgungsspannung betrieben werden können.

Eine wichtige Methode und Konsequenz aus den Gln. (3.2) bis (3.3) ist, dass durch Parallelverarbeitung idealerweise eine erhebliche Reduktion auf $\frac{1}{8}$ der Verlustleistung erfolgen kann. Der Grund liegt darin, dass bei Aufteilung einer Berechnungsaufgabe auf zwei separate getrennte Prozessoren bei idealer Parallelisierbarkeit jeder der Prozessoren nur mit halber Taktfrequenz arbeiten muss, um dieselbe Gesamtperformance zu erhalten. Gemäß Gl. (3.3) ist dadurch auch eine Reduktion der Versorgungsspannung auf die Hälfte möglich, wodurch sich je Prozessor nur mehr ein Achtel, insgesamt damit ein Viertel der Verlustleistung ergibt. Die Abb. 3.7 veranschaulicht diesen Sachverhalt.

Parallelverarbeitung kann auch durch „Pipelining“ erzielt werden. Im Gegensatz

zur echten parallelen Anordnung von Schaltkreisen ist die Implementierung einer Pipeline schaltungstechnisch weniger Aufwand und wird daher meist bevorzugt [65].

3.4.4 Technologische Maßnahmen

Die in der Abb. 3.5 dargestellten Einflussgrößen τI_k, Schwellenspannung U_{th} und Gatterkapazität C werden durch die technologischen Prozesse in der Halbleiterfertigung bestimmt [66, 67, 68]:

1. Die Schwellenspannung ist ein wichtiger Parameter in der CMOS Fertigung. Aus Gl. (3.4) erkennt man eine quadratische Abhängigkeit zwischen der maximalen Taktfrequenz und der Schwellenspannung. Da $U_{th} < U$ sein muss, ergibt sich bei Verringerung von U_{th} eine quadratische Steigerung von f_{max}. Wegen des exponentiellen Zusammenhangs aus Gl. (3.4) ist die Reduktion von U_{th} jedoch nach unten begrenzt, um den statischen Leckstrom I_0 klein zu halten. Um ohne Geschwindigkeitsverlust auch geringe statische Verlustleistungen zu erzielen, können durch die Herstellung von unterschiedlichen Transistoren schnell getaktete Schaltungsteile mit kleinerem U_{th} und im Hinblick auf die Schaltgeschwindigkeit unkritische Schaltungsteile mit höherem U_{th} ausgeführt werden [70, 71].

2. Der Kurzschlussstrom I_k, der während eines Schaltvorganges in einem CMOS Schaltungselement fließt, wird von den Schwellenspannungen der beteiligten nMOS und pMOS Transistoren beeinflusst. Bei Erfüllung der Bedingung $U < |U_{th,nMOS}| + |U_{th,pMOS}|$ kann der Kurzschlussstrom sogar völlig eliminiert werden. Die mittlere Zeitdauer τ dieses Kurzschlusses wird auch durch steilere Signalflanken an den Eingängen eines CMOS Gatters verkürzt [69].

3. Der statische Leckstrom I_0 wird sehr stark durch den Herstellungsprozess beeinflusst. Durch Verwendung von SOI (Silicon On Insulator) Technologie kann I_0 verringert werden [65].

Kapitel 4

Analyse der Aufgabenstellung im Lichte des Standes der Technik

Aus Kenntnis des in Kapitel 3 dargestellten Stands der Technik kann eine Bewertung und eine Analyse der Aufgabenstellung aus Kapitel 2 vorgenommen werden. Dazu ist es sinnvoll, die wesentlichen Kenngrößen für das geforderte Mikroprozessormodul (Performance, Verlustleistung und Baugröße) zusammen mit den am Markt befindlichen Produkten (Abschnitte 3.2.1 bis 3.2.4) in ein gemeinsames Portfolio (Abb. 4.1) einzutragen.

Das Portfolio enthält neben den Vertretern der am Markt befindlichen Mikroprozessorsysteme „Trizeps“, „Razor Blade“ und „iPAQ“ auch einen derzeit üblichen Arbeitsplatz-PC sowie die Kenngrößen der Aufgabenstellung. Der Durchmesser der Kreise ist ein Maß für die Rechenleistung. Eingetragen ist auch jener Bereich, der in der Literatur als „Low Power“ angesehen wird (etwa <5W).

Der Aufgabenstellung genügt kein kommerziell verfügbares Produkt

Die Marktrecherche ergibt, dass die in der Aufgabenstellung geforderten Kenngrößen zur Zeit von keinem Produkt auf dem Markt erfüllt werden kann. Zwar erreichen Mikroprozessormodule wie „Trizeps“ niedrige Werte für Verlustleistung und Baugröße, können allerdings nicht die erforderliche Performance aufweisen. In der Abb. 4.1 ist dies deutlich zu erkennen. Umgekehrt fallen ausreichend performante Mikroprozessormodule nicht unter die Kategorie „Low Power“.

Entwicklung muss besonders auf die Verlustleistung bedacht sein

Mikroprozessoren mit einer höheren Rechengeschwindigkeit als gefordert (>500MIPS), sind am Markt problemlos zu erhalten. Zieht man allerdings die Verlustleistung mit ins Kalkül, sieht die Situation völlig anders aus. Es ist damit für

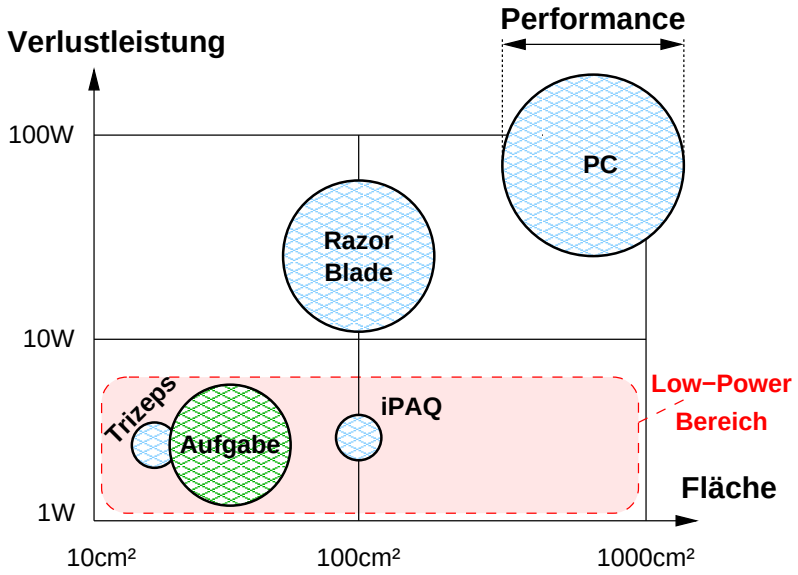


Abbildung 4.1: *Portfolio: Verlustleistung, Baugröße und Performance.* Dargestellt sind die Mikroprozessorsysteme „Trizeps", „iPAQ" und „Razor Blade" im Vergleich zur Aufgabenstellung und zu einem derzeit üblichen Arbeitsplatz-PC. Deutlich erkennbar ist, dass es im eingetragenen Low Power Bereich zur Zeit keine Mikroprozessorsysteme mit entsprechend hoher Rechenleistung gibt.

die Entwicklung des neuen Mikroprozessormoduls das Angebot an Mikroprozessoren genau zu hinterleuchten. Auch das Fassungsvermögen des erforderlichen Speichers (64Mbyte RAM, 64Mbyte ROM) ist unter dem Gesichtspunkt einer gesamten Verlustleistung von weniger als 2W problematisch. Insgesamt hat die Auswahl der für das Mikroprozessormodul erforderlichen Hardwarekomponenten große Bedeutung.

Maßnahmen für die Reduktion der Verlustleistung ergreifen

Um das Ziel einer drastischen Reduktion der Verlustleistung zu erfüllen, sind sämtliche Maßnahmen zu ergreifen, die eine Reduktion der Verlustleistung herbeiführen. Neben den in Kapitel 3 geschilderten, bekannten Methoden sind zusätzliche, neue Maßnahmen umzusetzen, um die Aufgabenstellung zu erfüllen.

Kompakte Bauweise durch BGA unterstützen

Die Beschränkung auf das Format einer Scheckkarte stellt für ein Mikroprozessorsystem mit Speicher, Logikbaustein und Ein-/Ausgabe ein kritisches Designkriterium dar, weil die Gehäuseformen der nötigen Bausteine entsprechend klein

sein müssen. Eine Fokussierung auf kleinste Gehäusegeometrieen im BGA-Format (Ball Grid Array) ist unbedingt notwendig.

Testgerechtes Design

Das Mikroprozessormodul ist so zu entwerfen, dass eine Fehlersuche und Diagnose während der Inbetriebnahmephase ohne größere Schwierigkeiten möglich ist. Aufgrund der kompakten Bauweise und der Verwendung von BGA-Gehäusen ist der Großteil der elektrischen Signalleitungen nicht zugänglich. Daher ist bereits in der Entwurfsphase darauf zu achten, dass wichtige Signalleitungen für den Anschluss von Messgeräten (Logikanalysator, Oszilloskop, etc.) z. B. durch gezieltes Herausführen an Messstellen zugänglich gemacht werden.

Preis entscheidet über Kompromisse

Zunächst stehen die Herstellkosten beim Entwurf eines Prototypen für das neue Mikroprozessormodul nicht im Vordergrund. Für eine spätere Überleitung in eine Serienfertigung spielt dagegen der Preis eine entscheidende Rolle.

Im Prototypen lassen sich einige Zielkonflikte somit besser lösen, da durch Auswahl von teureren Bauteilen die an das Mikroprozessormodul gestellten Anforderungen einfacher erfüllbar sind. Bei Überführung in die Serienreife müssen in der Praxis immer Kompromisse zugunsten der Herstellkosten eingegangen werden, wobei die erwarteten Stückzahlen den wesentlichsten Parameter bei der Preiskalkulation darstellen.

Kapitel 5

Ansatz zum Entwurf von hochperformanten Low Power Mikroprozessorysteмen durch Erweiterung der Systemgrenzen

Für den Entwurf von leistungsfähigen Low Power Mikroprozessorsystemen ist die Berücksichtigung einzelner Methoden zur Reduktion von Verlustleistung alleine nicht ausreichend. Vielmehr ist eine allgemeine *Betrachtungsweise der Gesamtheit aller verlustleistungsrelevanten Parameter* notwendig, um komplette Mikroprozessorsysteme, die aus einer Summe von Teilkomponenten bestehen, hinsichtlich Performance und Verlustleistung zu optimieren.

Das gemeinsame Miteinbeziehen und Abstimmen sämtlicher Faktoren, die maßgeblich an der Entstehung von Verlustleistung beteiligt sind, ist ein völlig neues Herangehen an das vernetzte, komplexe System „Verlustleistung". An die Stelle von isolierten Einzelmethoden tritt damit eine neue Methodik der totalen Systembetrachtung durch Erweiterung der Systemgrenzen.

Beim Entwurf von Mikroprozessorsystemen ist daher eine Optimierung des Leistungsverbrauches und der Rechengeschwindigkeit z. B. nicht auf hardwaretechnische Aspekte allein beschränkt, sondern erfordert insbesondere auch die Miteinbeziehung der am Mikroprozessorsystem abzuarbeitenden Softwarekomponenten, wie Betriebssystem, Gerätetreiber und Anwendungssoftware.

Die derzeit und traditionell betrieblich getrennte Entwicklung von Hard- und Softwarekomponenten macht eine gemeinsame Optimierung von Hardware und Software nach der Methodik erweiterter Systemgrenzen in der Praxis nahezu unmöglich. Da aber diese gemeinsame Betrachtungsweise für den Entwurf von hochperformanten Low Power Mikroprozessorsystemen von ausschlaggebender Bedeutung ist, sind künftig alternative Modelle einer erweiterten Zusammenarbeit in diesem Bereich nach holistischen Vorbild [72] notwendig.

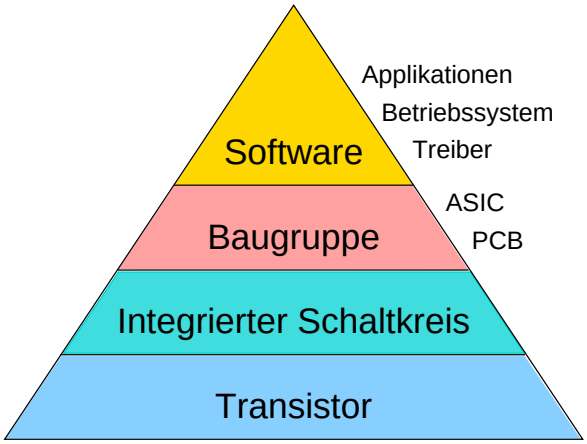


Abbildung 5.1: *Ebenenmodell für Mikroprozessorsysteme: Vertikale Vernetzung als Basis für totale Systembetrachtung.* Die Entwicklung von Low Power Mikroprozessorsystemen erfordert eine ebenenübergreifende Gesamtbetrachtung von verlustleistungsrelevanten Parametern von der technologischen Transistorebene bis hinauf zu Betriebssystem und Anwendungssoftware.

5.1 Ebenenmodell für Mikroprozessorsysteme

Um den Überblick über die zahlreichen, mitunter sehr komplexen Zusammenhänge zwischen Performance, Baugröße und Verlustleistung zu bewahren, ist die Darstellung eines Mikroprozessorsystems in Form eines Ebenenmodells (Abb. 5.1) hilfreich. Darin sind vier Ebenen enthalten, die jeweils für sich genommen zahlreiche Optimierungsmöglichkeiten hinsichtlich Verlustleistung und Performance enthalten:

- **Transistor-Ebene**: Als kleinstes Schaltelement in einem Mikroprozessorsystem stellen die Eigenschaften eines millionenfach vorhandenen Transistors maßgebende Einflussfaktoren auf die Rechengeschwindigkeit und den Leistungsverbrauch dar. Wichtigste Eigenschaften sind dabei die Strukturgröße, die Versorgungsspannung und die Schwellenspannung, wie in Abschnitt 3.4 dargestellt.

- **Integrierter Schaltkreis-Ebene**: Die Verschaltung von Millionen Transistoren und Gattern zu integrierten Schaltkreisen hoher Dichte hat in erster Linie über die Gatteraktivität α aus Gl. (3.2) großen Einfluss auf die Verlustleistung eines integrierten Bauelementes. Durch sorgfältigen Entwurf der logischen Verschaltung läßt sich die Anzahl der Schaltvorgänge deutlich reduzieren und somit Verlustleistung einsparen ohne Geschwindigkeitseinbußen hinnehmen zu müssen. In der Regel existieren für die Realisierung von einzelnen Funktionen (z. B. Integer-Addition) zahlreiche Schaltkreisvarian-

ten, von denen die hinsichtlich Verlustleistung effizienteste Variante gewählt werden kann.

- **Baugruppen-Ebene**: Darunter versteht man die hardwaretechnische Zusammenschaltung sämtlicher Halbleiterbauelemente zu einem gesamten Mikroprozessorsystem. Zentrale Bedeutung kommt in dieser Ebene dem Logikbaustein zu, der als Bindeglied und Vermittler zwischen den einzelnen Bauelementen fungiert. Wird dieser Logikbaustein z. B. in Form eines FPGA ausgeführt, kann flexibel auf individuelle stromsparende Modi der einzelnen Bauelemente eingegangen werden und somit die gesamte Verlustleistung abhängig vom Betriebszustand des Mikroprozessorsystems minimiert werden (z. B. Abschalten von Bauelementen, die gerade nicht benötigt werden). Im Unterschied zu den aus dem PC-Bereich bekannten „Power Management“ Funktionalitäten (z. B. Abschalten einer Festplatte), sind die Energiesparfunktionen auf der Baugruppen-Ebene nicht Software-gesteuert, sondern sind in der Hardware (z. B. im Logikbaustein) direkt verankert. Das bietet den enormen Vorteil, rasch und automatisch auf Änderungen des Belastungszustandes zu reagieren, ohne Einsatz zusätzlicher Software.

- **Software-Ebene**: Die Gesamtheit aller Programme (Betriebssystem, Gerätetreiber, Anwendungsprogramme), die auf einem Mikroprozessorsystem ablaufen, ist wesentlich für den gesamten Leistungsbedarf mitverantwortlich. Nachdem jede Rechenoperation bzw. jeder Programmschritt Energie benötigt, kann durch effiziente Gestaltung der Software ein erhebliches Einsparungspotential genutzt werden. In der Praxis lassen sich z. B. etwa 10-20% aller Mikroprozessorbefehle einer nicht optimierten Software allein durch den Einsatz eines optimierenden Compilers einsparen. Weitere erhebliche Einsparmöglichkeiten ergeben sich durch die Verbesserung der Algorithmen oder in Maschinensprache abgefasste, handcodierte Programmsequenzen. Auch das Betriebssystem besitzt entscheidenden Einfluss auf den Leistungsverbrauch. Da sämtliche Hardwarekomponenten (Ressourcen) eines Mikroprozessorsystems direkt unter der Kontrolle des Betriebssystems stehen, sind ein sparsamer Umgang mit diesen Komponenten und die Ausnutzung von Energiesparmodi bei geringer Rechenlast wichtige Eigenschaften eines Low Power Betriebssystems.

5.2 Verlustleistungsrelevante Zusammenhänge

Es gibt eine Vielzahl von Zusammenhängen, die beim Entwurf eines Low Power Mikroprozessorsystems von Bedeutung sind. Unter Zuhilfenahme des Ebenenmodells aus Abb. 5.1 lassen sich diese Zusammenhänge übersichtlich darstellen und damit ebeneninterne und ebenenübergreifende Abhängigkeiten erkennen. Einige dieser Abhängigkeiten sind exemplarisch in der Abb. 5.2 dargestellt.

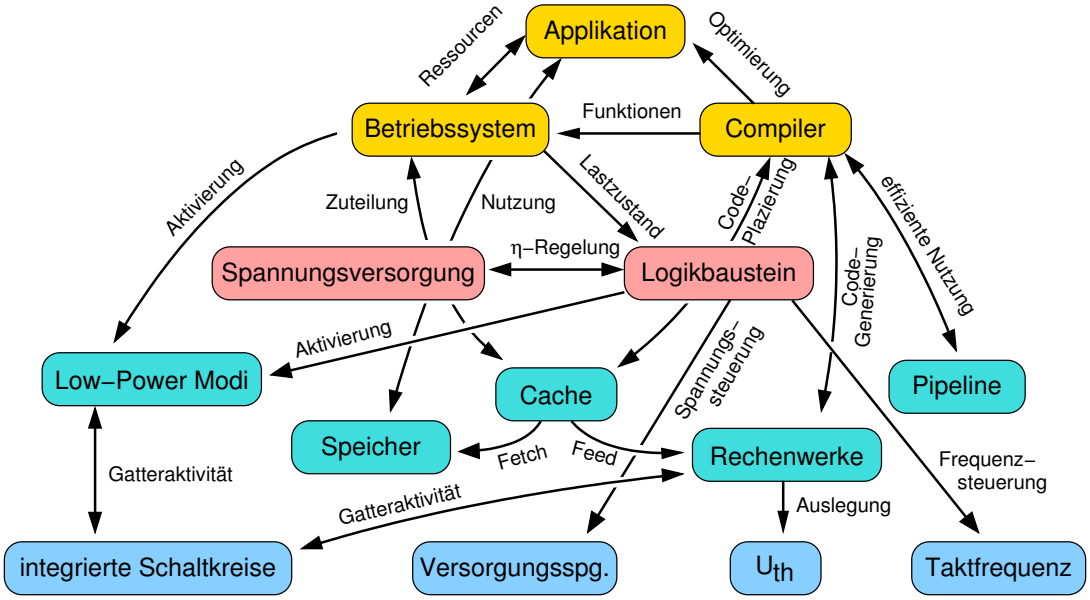


Abbildung 5.2: *Verlustleistungsrelevante Zusammenhänge anhand des Ebenenmodells aus Abb. 5.1.* Die Vielzahl von Zusammenhängen erfordert beim Entwurf von Low Power Mikroprozessorsystemen nicht nur ebeneninterne sondern auch ebenenübergreifende Optimierung.

In der industriellen Praxis werden die Entwicklungstätigkeiten auf den in Abbn. 5.1 und 5.2 dargestellten Ebenen durch unterschiedliche Personen in unterschiedlichen Firmen mit jeweils unterschiedlichen Interessen durchgeführt. In der Regel sind die Entwicklungsaktivitäten auf jeweils einzelne Ebenen beschränkt (Chip-, Board-, Software-Entwicklung), wodurch den ebenenübergreifenden Abhängigkeiten nur untergeordnete Bedeutung zukommt. Es ist naheliegend, dass eine intensive, interaktive und ebenenübergreifende Zusammenarbeit zwischen den beteiligten Entwicklern durch entsprechend feste Verknüpfung ihrer Aktivitäten sehr erstrebenswert ist. Ein erfolgreicher Low Power Entwurf setzt eine solche Vernetzung sämtlicher Tätigkeiten über die Ebenengrenzen hinweg voraus.

5.3 Ein ebenenübergreifender Optimierungsansatz

Die Idee der ebenenübergreifenden Optimierung ist die gleichzeitige und umfassende ebenenübergreifende Betrachtung von verlustleistungsrelevanten Parametern von der Transistorebene bis zur Softwareebene. Nur die gemeinsame Beherrschung sämtlicher verlustleistungsrelevanter Zusammenhänge und Methoden ermöglicht einen effizienten Low Power Entwurf.

Dies verlangt ein enges, reibungsfreies und vor allem ebenenübergreifendes Zusammenspiel zwischen Methoden und Ergebnissen aus den einzelnen Ebenen. So ist beispielsweise die Abstimmung zwischen dem Betriebssystem und dem Logikbaustein für die effiziente Nutzung von Energiesparmodi oder die Berücksichtigung von architekturspezifischen Strukturen des Mikroprozessors durch den Compiler besonders wichtig.

Allgemeine Formulierung

Eine ebenenübergreifende Optimierung verlangt die multidisziplinäre Betrachtung vieler unterschiedlicher verlustleistungsrelevanter Zusammenhänge. Eine geschlossene mathematische Formulierung einer Optimierungsaufgabe gestaltet sich daher äußerst schwierig und erfordert den Einsatz von vereinfachten mathematischen Modellen.

Grundsätzlich entspricht eine Optimierungsaufgabe der Suche nach einem globalen Maximum/Minimum einer Güte- bzw. Kostenfunktion G bei Variation eines n-dimensionalen Parametersatzes $\mathbf{x}$:

$$G(\mathbf{x}) \;\rightarrow\; max, \;\; \mathbf{x} = (x_1, x_2, ..., x_n)^T \tag{5.1}$$

unter den m Neben- bzw. Randbedingungen (Gleichungen und Ungleichungen)

$$\mathbf{R}(\mathbf{x}) = (r_1(\mathbf{x}), r_2(\mathbf{x}), ..., r_m(\mathbf{x}))^T \leq \mathbf{0}. \tag{5.2}$$

Im Falle dass die Nebenbedingungen ausschließlich als Gleichungen vorliegen, kann das Optimierungsproblem durch Einführen des Lagrange-Multiplikators gelöst werden [74]. Bei Vorhandensein von Ungleichungen ist

$$\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)^T \tag{5.3}$$

eine Lösung dieses Optimierungsproblems, wenn es m Zahlen

$$\mathbf{u} = (u_1, u_2, ..., u_m)^T \tag{5.4}$$

gibt, so dass folgende (Kuhn-Tucker) Bedingungen erfüllt werden:

$$\begin{aligned} \left.\frac{\partial G}{\partial x_i}\right|_{\mathbf{x}^*} - \sum_{j=1}^{m} u_j \left.\frac{\partial r_j}{\partial x_i}\right|_{\mathbf{x}^*} &\leq 0, \\ x_i^* \left.\left(\frac{\partial G}{\partial x_i} - \sum_{j=1}^{m} u_j \frac{\partial r_j}{\partial x_i}\right)\right|_{\mathbf{x}^*} &= 0, \end{aligned}$$

$$\begin{aligned} r_j(\mathbf{x}^*) &\leq 0, \\ r_j(\mathbf{x}^*)u_j &= 0, \\ x_i &\geq 0, \\ u_j &\geq 0, \\ i &= 1, 2, ..., n. \\ j &= 1, 2, ..., m. \end{aligned} \tag{5.5}$$

Voraussetzung für diese notwendigen Bedingungen ist, dass die Funktionen $G(\mathbf{x})$ und $\mathbf{R}(\mathbf{x})$ stetig differenzierbar sind. Im Falle einer konkaven Funktion $G(\mathbf{x})$ und konvexen Randbedingungen $\mathbf{R}(\mathbf{x})$ ist (5.5) nicht nur notwendig, sondern auch hinreichend [75, 76].

Praktische Umsetzung

Die praktische Anwendung der Optimierungsvorschrift Gl. (5.5) zur Minimierung der Verlustleistung ist eine Herausforderung:

- Die Gleichungen sind nichtlinear
- Die Vielzahl von Zusammenhänge (siehe Abb. 5.2) führt zu hohen Dimensionen n.
- Wesentliche Parameter, wie die Gatteraktivität α, sind abhängig von dem am Mikroprozessorsystem ausgeführten Anwendungsprogramm - eine Optimierung hinsichtlich dieses Parameters ist daher applikationsspezifisch und nicht allgemein gültig.
- Zahlreiche Sachverhalte in einem Mikroprozessorsystem (z. B. die Auslastung von Cache-Speichern, oder die Ausführungshäufigkeit bestimmter Maschineninstruktionen, zeitliches Verhalten eines Anwendungsprogrammes, etc.) lassen sich praktisch ausschließlich durch Simulation erfassen. Eine mathematische Formulierung ist nur unter stark vereinfachten Annahmen möglich.
- Die digitale Natur von Logikschaltungen verursacht Unstetigkeiten in den mathematischen Zusammenhängen und führt dadurch zur Diskretisierung des Problems (z. B. Schaltungsteil ein- oder ausgeschaltet).

Die Minimierung der Verlustleistung führt damit zu einem nichtlinearen Optimierungsproblem mit diskreten Variablen, was den Einsatz von Methoden der nichtlinearen, gemischt-ganzzahligen Optimierung erfordert [77].

Problemseparierung

Um die Komplexität des Optimierungsproblems zu verringern, und damit eine praktische Umsetzung zu erleichtern, bietet sich eine Separierung in zwei unterschiedliche Teilprobleme an. Bei näherer Betrachtung der in Abb.5.2 dargestellten verlustleistungsrelevanten Zusammenhänge zeigt sich nämlich, dass sich diese in zwei Klassen einteilen lassen:

1. effiziente *Erbringung* einer digitalen Logikfunktion
2. effiziente *Nutzung* einer digitalen Logikfunktion

Dies entspricht einer Separierung in eine möglichst energieeffiziente Bereitstellung von Logikfunktionalität (Energie je Schaltvorgang) und einer möglichst effizienten Nutzung der bereitgestellten Logikfunktionen, indem für eine bestimmte, zu erfüllende Aufgabe eine möglichst geringe Menge an Schaltvorgängen verwendet wird.

Die Bereitstellung von energetisch effizienter Logik ist stark technologiegetrieben und findet ausschließlich auf der Transistorebene des Modells aus Abb. 5.1 statt. Die effiziente Nutzung von Logikfunktion ist hingegen auf allen übrigen Ebenen gleichermaßen relevant.

Eine Separierung ermöglicht es, die mathematisch leichter formulierbaren Teilprobleme auf Transistorebene getrennt von den stark software- und ablauforientierten Teilproblemen auf den darüberliegenden Ebenen zu betrachten. Im ersten Fall bieten sich analytische Lösungsmöglichkeiten an, während im zweiten Fall auf Methoden der Simulation zurückgegriffen werden kann.

5.3.1 Effiziente Erbringung einer digitalen Logikfunktion

Die energieeffiziente Erbringung von digitalen, logischen CMOS Funktionen auf der Transistorebene hängt in überwiegendem Ausmaß mit der erreichbaren Miniaturisierung der Transistorstrukturen und der damit korrespondierenden Betriebsspannungen zusammen. Je kleiner die geometrischen Abmessungen, desto geringer ist der Energieaufwand bei logischen Zustandsänderungen. Kleine Transistorstrukturen bedeuten allerdings auch verminderte Treiberleistung und wirken sich somit auch negativ auf die erreichbaren Schaltgeschwindigkeiten und der daraus erzielten Performance aus.

Es ist daher notwendig, nicht nur die benötigte Energie je Zustandswechsel zu minimieren, sondern auch die Schaltgeschwindigkeit in die Betrachtung miteinzubeziehen. Da Rechengeschwindigkeit und Verlustleistung gegenläufige Anforderung darstellen, kann eine Optimierung so erfolgen, dass das Verhältnis zwischen Verlustleistung und Rechengeschwindigkeit gemäß Gl. (3.1)

$$G(\mathbf{x}) \equiv p = \frac{P_V}{M} \rightarrow \min \tag{5.6}$$

als Kostenfunktion im Sinne von Gl. (5.1) minimiert wird. Die Verlustleistung P_V in einem CMOS Schaltkreis kann gemäß Kapitel 3 durch

$$P_V = \alpha f C U^2 + \tau \alpha f U I_k + U I_0 \tag{5.7}$$

mit der Gatteraktivität α, der Taktfrequenz f, der mittleren Gatterkapazität C, der Versorgungsspannung U, der Kurzschlusszeit τ, dem Kurzschlussstrom I_k und dem statischen Leckstrom I_0 berechnet werden, wobei der zweite Term in modernen CMOS Logikschaltungen eine untergeordnete Rolle spielt und vernachlässigt werden kann [30]. Mit dem statischen Leckstrom

$$I_0 = k_I \exp\left(-\frac{qU_{th}}{kT}\right), \tag{5.8}$$

der für RISC-Architekturen geltenden Rechengeschwindigkeit bzw. Performance

$$M = k_M f \tag{5.9}$$

und der Substitution der Schwellenspannung U_{th} aus dem Zusammenhang

$$f = k_f \frac{(U - U_{th})^2}{U} \tag{5.10}$$

ergibt sich für die Gl. (5.6)

$$p = \frac{C}{k_M}\alpha U^2 + \frac{k_I}{k_M}\frac{U}{f} \exp\left[-\frac{q}{kT}\left(U - \sqrt{\frac{Uf}{k_f}}\right)\right]. \tag{5.11}$$

p ergibt sich damit aus zwei Termen, wobei der erste Term eine quadratische Abhängigkeit von U aufweist und der zweite Term exponentiell für kleine U und große f wächst. Der für die Optimierung maßgebliche Parametersatz $\mathbf{x}$ ergibt sich daher zu

$$\mathbf{x} = (\alpha, U, f)^T. \tag{5.12}$$

Eine grafische Veranschaulichung der Verlustleistung enthält die Abb. 5.3, in der das jeweilige Leistungsminimum im Übergangsbereich zwischen quadratischem und exponentiellem Anstieg erkennbar ist. Für die Gewinnung dieses Diagramms wurde von einer konstanten Gatteraktivität $\alpha = 0.24$ ausgegangen, um eine grafische Darstellung in zwei Veränderlichen zu ermöglichen. Weiters liegen in Übereinstimmung mit [78] folgende Zahlenwerte zugrunde:

$$U_{th} = 1\mathbf{V}, C = 1\mathbf{nF}, k_M = 1, k_I = 2.55 \cdot 10^{15}\mathbf{A}, k_f = 666 \cdot 10^6 (Vs)^{-1}, T = 300\mathbf{K} \quad (5.13)$$

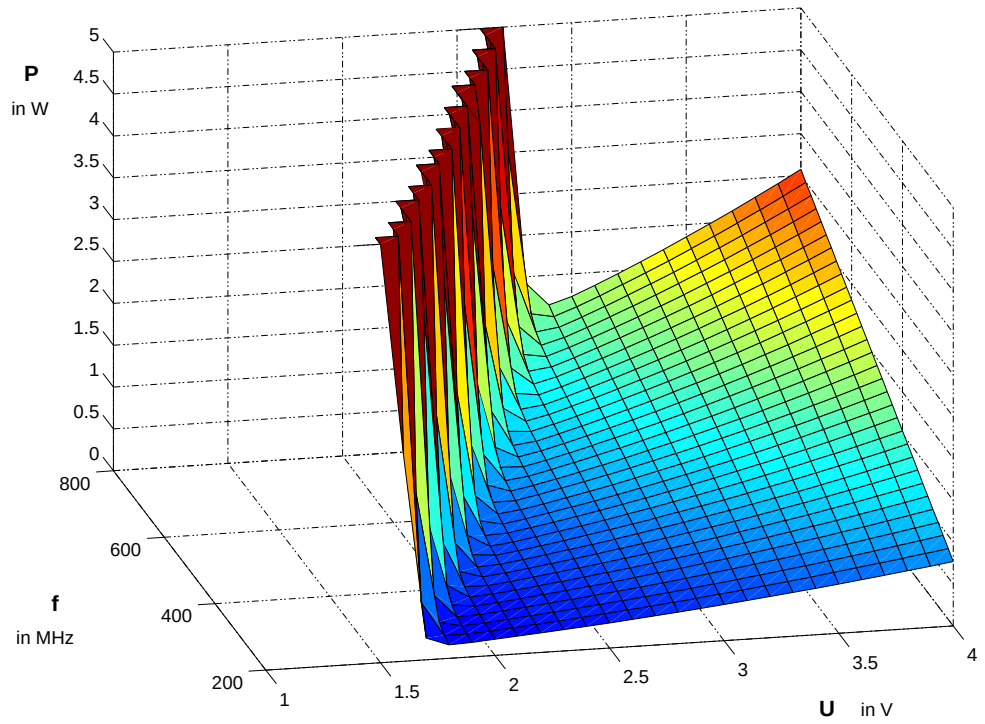


Abbildung 5.3: *Verlustleistung P in Abhängigkeit von Versorgungsspannung U und Betriebsfrequenz f.* Die Verlustleistung steigt quadratisch mit U bzw. linear mit f. Deutlich erkennbar ist der rasche exponentielle Anstieg der Verlustleistung bei kleinen U und großen f, hervorgerufen durch den statischen Leckstrom. Je nach erforderlicher Rechenleistung kann durch Anpassung von U die Verlustleistung auf ein Minimum reduziert werden, sodass der Betriebspunkt in der Talsohle dieses charakteristischen Verlaufes zu liegen kommt. Die Zahlenwerte sind aus [98] entnommen.

Als Randbedingung ist für ein Mikroprozessorsystem in der Praxis zumeist eine bestimmte Rechengeschwindigkeit gefordert. Wegen Gl. (5.9) ist daher

$$\mathbf{R}(\mathbf{x}) \equiv f_0 - f \leq 0 \quad (5.14)$$

mit einer Mindestbetriebsfrequenz f_0. Unter Zuhilfenahme der Optimierungsvorschrift aus Gl. (5.5) kann mit $f = f_0$ das Minimum der bezogenen Verlustleistung p aus

$$2f_0C\alpha U + k_I \exp\left[-\frac{q}{kT}\left(U - \sqrt{\frac{Uf_0}{k_f}}\right)\right]\left[1 - \frac{q}{kT}\left(U - \sqrt{\frac{Uf_0}{k_f}}\right)\right] = 0 \qquad (5.15)$$

errechnet werden. Eine Veranschaulichung der numerischen Lösung dieses Zusammenhangs mit den Zahlenwerten aus Gl. (5.13) enthält die Abb. 5.4. Daraus ist abzulesen, dass für minimale Verlustleistung die Versorgungsspannung U näherungsweise linear mit der geforderten Taktfrequenz f_0 zusammenhängt.

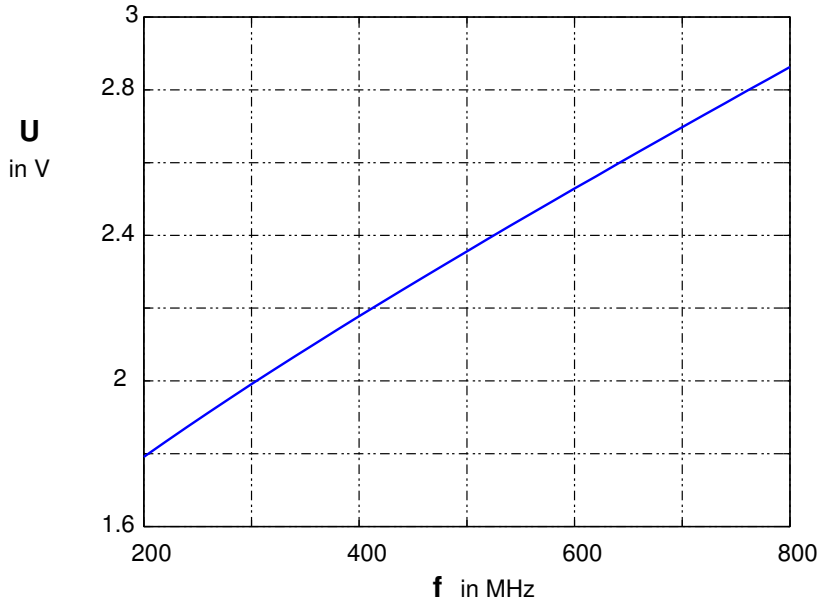


Abbildung 5.4: *Versorgungsspannung U für minimale bezogene Verlustleistung p bei gegebener Betriebsfrequenz f_0.* Der Verlauf dieser Kennlinie ist nahezu linear. Damit ist für eine Minimierung der bezogenen Verlustleistung einer CMOS-Logikschaltung die Versorgungsspannung näherungsweise linear zur Änderung der Taktfrequenz nachzuregeln.

5.3.2 Effiziente Nutzung digitaler Logikfunktionen

Die Verringerung der Gatteraktivität ist eine ebenenübergreifende Aufgabe. Ziel ist die möglichst effiziente Nutzung vorhandener Logikstrukturen für die Implementierung einer gegebenen Applikation. Dies ist gleichbedeutend mit der Minimierung der insgesamt oder je Zeiteinheit erforderlichen Schaltspiele in den Transistoren der zugrundeliegenden Hardwarebausteine. Als Schaltspiel ist dabei eine

Transition von logisch „0“ auf logisch „1“ oder von logisch „1“ auf logisch „0“ zu verstehen.

Sei τ die Periodendauer eines Taktsignals einer synchron[1] getakteten Logikschaltung und $s(k\tau)$ mit $k \in \mathbb{N}$ die Anzahl von Schaltspielen im Zeitraum $(k-1)\tau \leq t < k\tau$ innerhalb einer Taktperiode, so ergibt sich die bis zum Zeitpunkt $k\tau$ insgesamt ausgeführte Gesamtanzahl $S(k\tau)$ von Schaltspielen zu

$$S(k\tau) = \sum_{i=1}^{k} s(i\tau). \tag{5.16}$$

$S(k\tau)$ ist damit ein Maß für in einer Logikschaltung benötigten Gesamtenergie, während $s(k\tau)$ ein Maß für die momentane Verlustleistung zum Zeitpunkt $k\tau$ darstellt.

Das Bestreben, digitale Logikfunktionen möglichst effizient zu nutzen, bedeutet damit, eine gestellte Aufgabe (z. B. Berechnungsvorgang) innerhalb einer maximalen Zeitschranke $K\tau$ mit einer geringst möglichen Anzahl von Schaltspielen durchzuführen:

$$S(K\tau) \rightarrow \min. \tag{5.17}$$

Betrachtet man die von einem digitalen Schaltkreis erbrachten logischen Funktionen als eine Menge von ν übergeordneten, elementaren Berechnungsoperationen (z. B. Addition oder Multiplikation zweier Zahlen), so lässt sich das Minimierungsproblem auch als

$$S(K\tau) = \sum_{j=1}^{\nu} N_j(K\tau) S_j \rightarrow \min \tag{5.18}$$

anschreiben, wobei S_j die Anzahl der für die Berechnungsoperation j erforderlichen Schaltspiele und $N_j(K\tau)$ die Anzahl der bis zum Zeitpunkt $K\tau$ ausgeführten Berechnungsoperationen j darstellt.

Die Gl. (5.18) bedeutet eine wichtige Abstrahierung von der Transistorebene, indem einfache logische Funktionen (z. B. „Und“, „Oder“, „Nicht“) zu komplexeren, höherwertigen Maschinenbefehlen (z. B. Multiplikation) zusammengefasst werden. Diese Abstraktionsebene kann auch als Schnittstelle zwischen Hard- und Software aufgefasst werden.

Für die Minimierung der Gl. (5.18) lassen sich unmittelbar folgende Maßnahmen ableiten:

[1] Mikroprozessorsysteme werden heute wegen der besseren Beherrschbarkeit praktisch ausschließlich synchron getaktet.

- Auf der Hardwareebene sind jene Schaltungsvarianten vorzuziehen, die einzelne Berechnungsoperationen mit möglichst wenigen Schaltspielen S_j implementieren.
- Die zur Übersetzung von Anwendungsprogrammen verwendeten Compiler haben jene Maschinenbefehle gegenüber anderen vorzuziehen, die kleinere S_j besitzen. Heutige Compiler verfügen lediglich über die Möglichkeit, entweder besonders schnell ablaufenden oder besonders kompakten Maschinencode zu erzeugen.
- Effiziente Algorithmen können auf Applikations- bzw. Betriebssystemebene entscheidende Einsparungen in $S(K\tau)$ hervorrufen, da jegliche Programmänderung auf dieser Ebene mit einer enorm hohen Zahl von Schaltspielen einhergeht. Umgekehrt können aber auch schlecht implementierte Applikationen die Energieeffizienz eines Mikroprozessorsystems in einem negativen Sinn beeinflussen.

5.4 Konsequenzen und Entscheidungskriterien für die Realisierung von Low Power Mikroprozessorsystemen

Beim Entwurf von Mikroprozessorsystemen mit geringem Leistungsverbrauch sind einige wichtige Kriterien zu beachten und Fragen zu beantworten, die entscheidenden Einfluss auf die gesamte Hard- und Softwarearchitektur haben. Im Folgenden wird auf einige typische Fragestellungen näher eingegangen:

5.4.1 Spezialisierung auf die Anwendung

Wird ein Mikroprozessorsystem nicht für möglichst allgemeine Verwendbarkeit (wie z. B. Arbeitsplatz-PC) sondern - wie im Bereich der Embedded Systeme üblich - für einen ganz bestimmten Zweck entwickelt, kann eine genaue Analyse des Zusammenspiels zwischen Anwendungssoftware, Betriebssystem und Hardwarekomponenten durchgeführt und daraus Verbesserungsvorschläge hinsichtlich Verlustleistung und Performance abgeleitet werden. Als sehr vorteilhaft erweisen sich hierfür Simulationswerkzeuge, die es gestatten, das Systemverhalten genauer zu beobachten, als dies im echten Betrieb aufgrund von schwer messbaren Größen (z. B. Cache-Effizienz, Kapitel 8) möglich ist.

Durch die Spezialisierung auf eine bestimmte Anwendung können die im Mikroprozessorsystem ablaufenden Algorithmen einer genauen Untersuchung unterzogen werden, um daraus eine bestmögliche Abstimmung aller Softwarekomponenten untereinander zu gewinnen. Ziel ist eine energieeffiziente Gestaltung der Softwarearchitektur.

5.4.2 Maßgebliche verlustleistungsbestimmende Komponenten eines Mikroprozessorsystems

Für eine effiziente Reduktion von Verlustleistung ist die Kenntnis, welche Komponenten in einem Mikroprozessorsystem die dominante Rolle in der gesamten Verlustleistungsbilanz spielen, vorteilhaft. Eine grobe Einschätzung aus der industriellen Praxis besagt, dass der Mikroprozessor selber etwa die Hälfte der gesamten Systemleistung benötigt. Da dies von vielen Einflussfaktoren abhängt, ist eine genauere Untersuchung dieses Sachverhaltes sinnvoll. Für die in Kapitel 3 vorgestellten Mikroprozessorsysteme wurde daher eine Auflistung der Verlustleistung nach einzelnen Teilkomponenten in der Tab. 5.1 vorgenommen.

Produkt	Razor Blade	HS1600	Trizeps	iPAQ	gesamt
Prozessor	15	2.5	0.5	0.5	58%
RAM	2.1	1.3	0.3	0.32	13%
ROM	0	0	0.08	0.1	1%
Logikbaustein	3.8	1.3	0.25	0.26	18%
I/O	1.3	0.6	0	0.22	7%
Sonstiges	0.7	0.3	0.07	0.1	3%
gesamt	22.9	6	1.2	1.5	100%

Tabelle 5.1: *Anteil an der gesamten Verlustleistung für einzelne Komponenten eines Mikroprozessorsystems.* Die Tabelle enthält die in Watt angegebenen Verlustleistungen für die in Kapitel 3 vorgestellten Mikroprozessormodule und ihren Teilkomponenten [24, 26, 28, 29]. Die letzte Spalte gibt den prozentuellen, durchschnittlichen Anteil an der gesamten Verlustleistung für die jeweilige Teilkomponente an.

Darin wird der Anteil des Prozessors an der gesamten Systemleistung mit 58% ausgewiesen. Dieser Prozentsatz wird allerdings stark vom Mikroprozessormodul „Razor Blade“ geprägt, wo der eingesetzte Pentium III Prozessor 65% der gesamten Systemleistung in Anspruch nimmt. Die Mikroprozessoren der anderen Module benötigen dem gegenüber deutlich weniger als die halbe Systemleistung.

Die Konsequenz aus der Tab. 5.1 ist, das Hauptaugenmerk bei der Verlustleistungsoptimierung auf den Mikroprozessor selbst zu legen.

5.4.3 Verzicht auf Gleitkommaarithmetik

Wird ein Mikroprozessorsystem für eine bestimmte Aufgabe konzipiert, kann anhand der dazu notwendigen Berechnungsvorgänge abgeschätzt werden, ob eine

numerische Darstellung der Berechnungsvariablen im Rahmen der Gleitkommaarithmetik („Floating Point“) notwendig und sinnvoll ist. Im Sinne eines Low Power Entwurfs ist nach Möglichkeit auf Gleitkommaarithmetik zu verzichten und Festkommaarithmetik zu bevorzugen.

Die hardwaretechnische Implementierung einer Floating Point Unit (FPU) ist wesentlich aufwendiger als eine reine Integer Unit und benötigt daher auch eine Vielfaches an Chipfläche bzw. Transistoren [73]. Über die Gatteraktivität α aus Gl. (3.2) wirkt sich eine FPU unmittelbar negativ auf die Verlustleistung eines Mikroprozessors aus, da die aufwendigen Schaltungsstrukturen die Anzahl der Schaltspiele vergrößern.

5.4.4 „Hardware On Demand“ durch Betriebssystemunterstützung

Durch den Einsatz von FPGAs verfließt die Grenze zwischen fest vorgegebenen Hardwarestrukturen und flexibel austauschbaren Softwaremodulen. Da die Logikfunktionen eines FPGA durch Programmierung, also per Software festgelegt werden, lässt sich durch Wiederprogrammierung das Verhalten des FPGA völlig verändern [124]. Moderne FPGAs erlauben eine komplette oder auch teilweise Wiederprogrammierung während des Betriebes sodass man die Logikfunktionen des FPGA laufend bedarfsgerecht anpassen kann. Die Abb. 5.5 stellt dies grafisch dar.

Mithilfe einer solchen „Hardware On Demand“ Methode lässt sich beispielsweise die Anzahl von gleichzeitig in Betrieb befindlichen Hardware-Funktionsblöcken auf das jeweils zu einem bestimmten Zeitpunkt unbedingt benötigte Ausmaß reduzieren. Dies ist allein deshalb sinnvoll, da die gleichzeitige Verwendung sämtlicher in einem Mikroprozessorsystem vorhandenen Hardwarekomponenten sehr unwahrscheinlich ist. Die Allokation der Hardwareressourcen ist dabei vom Betriebssystem anhand einer Bedarfsermittlung durchzuführen.

Eine Wiederprogrammierung muss im Rahmen des „Hardware On Demand“ Konzeptes nicht notwendigerweise während des Betriebes erfolgen, sondern kann auch von Konfigurations-, Analyse- und Wartungswerkzeugen vorgenommen werden.

Von enormer Bedeutung ist die Wiederprogrammierbarkeit eines FPGA auch im Falle eines Software-Updates, um ein optimales Zusammenwirken zwischen der neu eingespielten Software und der für diese Software eventuell abzuändernden Hardware zu erhalten.

5.4.5 Performancegewinn durch breite Datenbusse

Bei der Auslegung des Speichersubsystems spielt die Datenbusbreite eine entscheidende Rolle. Breite Busse erfordern eine Parallelschaltung von mehreren

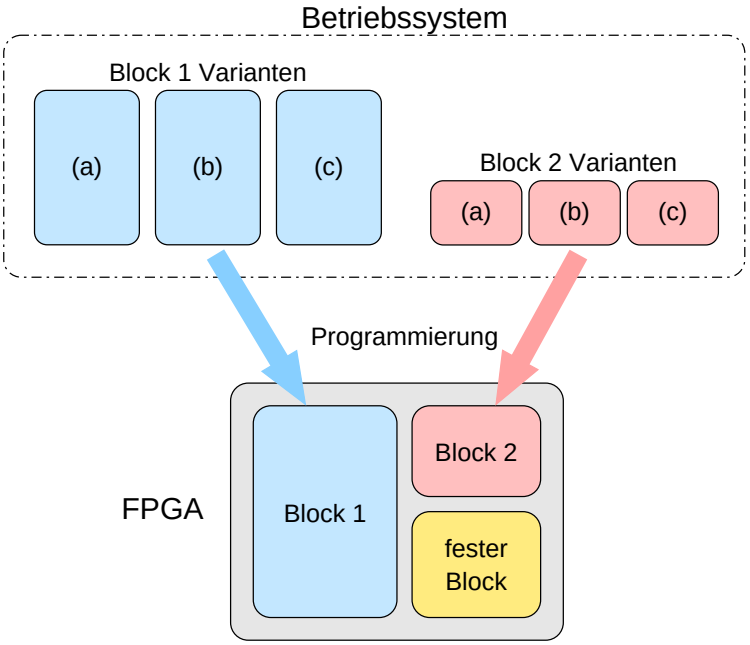


Abbildung 5.5: *"Hardware On Demand"*. Neben festen Anteilen kann durch Wiederprogrammieren von Funktionsblöcken das Verhalten des FPGA völlig neu bestimmt werden. Die Funktionalität des FPGA kann auf diese Weise je nach Bedarf flexibel durch das Betriebssystem angepasst werden.

Speicherbausteinen, deren Leistungsverbrauch sich addiert. Es ist daher eine möglichst geringe Anzahl von parallel geschalteten Speicherbausteinen anzustreben.

Man muss allerdings bedenken, dass bei halber Datenbusbreite für ein und dieselbe Datenmenge auch doppelt so viele Buszyklen erforderlich sind, was etwa denselben Energiebedarf bedeutet, unter der Voraussetzung, dass sich die Speicherbausteine in einem „Power-Save“ Zustand befinden wenn keine Buszyklen erfolgen.

Dieser Effekt läßt sich bei batteriebetriebenen Systemen ausnutzen, indem man bei konstantem Energieverbrauch (nur dieser ist für die Betriebsdauer einer Batterie hauptsächlich maßgebend) kurzzeitig höhere Verlustleistungen in Kauf nimmt, um durch parallel geschaltete Speicherbausteine die Systemperformance zu erhöhen. Eine derartige Maßnahme muss jedoch mit anderen Designkriterien (Baugröße, Kosten) abgestimmt werden.

5.4.6 Die Wahl des Betriebssystems

Für die Wahrnehmung verlustleistungssensitiver Aufgaben, wie z. B. das Schalten in verbrauchsarme Zustände, spielt das Betriebsystem eine zentrale Rolle, wie auch die Abb. 5.2 zeigt. Im Anwendungsfall ist daher stets das Betriebssystem auf

die gegebenen Hardwareverhältnisse sowie die anwendungsspezifischen Softwareprozesse anzupassen. Nicht offengelegte kommerzielle Betriebssysteme (z. B. VxWorks, Windows, QNX, OS-9, LynxOS, Virtuoso, etc.) bieten hier nur die Möglichkeit zusammen mit dem Hersteller derartige Anpassungen vorzunehmen. Freie, offene Betriebssysteme wie z. B. Linux liegen in Form der Quellcodes vor und erlauben so beliebige Eingriffe.

5.4.7 Modellierung von Betriebszuständen

Die Identifikation verschiedener Betriebszustände, die von einem Mikroprozessorsystem während der Bewältigung seiner Aufgaben eingenommen werden, ist für die Nutzung von verbrauchsarmen Modi einzelner Hardwarekomponenten von eminenter Bedeutung. Nach Modellierung dieses Zustandsgraphen kann für jeden dieser Zustände ein Minimum an erforderlichen Hardwareressourcen zugeteilt werden. Wie auch aus Abb. 5.2 erkennbar ist, kann unter Kenntnis des aktuellen Betriebszustandes auf allen Ebenen der Abb. 5.1 eine umfassende ebenenübergreifende Ressourceneinsparung erzielt werden.

Bei der Modellierung der Betriebszustandsgraphen sind auch die Übergänge zwischen den einzelnen Zuständen einer genauen Betrachtung zu unterwerfen. Muss der Übergang nämlich rasch vollzogen werden (z. B. rasche Reaktion auf ein Ereignis), ist das Einsparungspotential begrenzt, da die Zeitdauer einer Transition aus einem ressourcenschonenden Betriebszustand heraus mit zunehmenden Einsparungspotential wächst (z. B. Intel XScale Prozessor: Transition aus dem „Idle“-Modus: 10 Prozessortakte, Transition aus dem „Drowsy“-Modus: 2000 Prozessortakte, Transition aus dem „Sleep“-Modus: vollständiger Systemreset mit $> 10^6$ Prozessortakten).

5.4.8 Nutzung verbrauchsarmer Zustände

Zahlreiche Bausteine, die in Mikroprozessorsystemen zum Einsatz kommen, verfügen über verbrauchsarme Betriebsmodi. Deren rigorose Nutzung ist Grundvoraussetzung für einen Low Power Entwurf.

Da Energiesparmodi praktisch immer mit gewissen Einschränkungen verbunden sind, ist eine Aktivierung solcher Modi stets im Einklang mit dem Betriebs-Zustandsgraphen durchzuführen, was die Bedeutung der Modellierung von Betriebszuständen unterstreicht.

Durch den Einsatz von programmierbaren Logikbausteinen können verbrauchsarme Zustände anderer Hardwarekomponenten rasch und automatisch aktiviert werden, ohne zusätzlichen Eingriff durch Software. Innerhalb der Logikbausteine selbst ist durch gezieltes Abschalten („gated clock“ - Techniken) von momentan nicht benötigten Schaltungsteilen die Verlustleistung so gering als möglich zu halten.

5.4.9 Sorgfältige Auswahl von Bauteilen

In der Praxis besteht beim Entwurf eines Low Power Mikroprozessorsystems in der Regel keine unmittelbare Möglichkeit, auf die Chip- oder Transistorebene Einfluss zu nehmen. Vielmehr sind aus der Menge kommerziell erhältlicher Hardwarebausteine jene auszuwählen, die den gestellten Anforderungen kostenoptimal gerecht werden. Bei diesem Auswahlverfahren muss jedoch genau hinterleuchtet werden, unter welchen technologischen Randbedingungen ein Baustein beim Hersteller entwickelt und gefertigt wurde. Nur solche Bausteine, bei denen der Leistungsverbrauch besondere Berücksichtigung fand (z. B. geringe Versorgungsspannung, Energiesparmodi, variable Betriebsfrequenz, etc.), können in Low Power Mikroprozessorsystemen eingesetzt werden.

Kapitel 6

Entwicklung eines neuen Mikroprozessormoduls

Um die Anforderungen aus Kapitel 2 zu erfüllen, bedarf es der Methodik aus Kapitel 5, wo verdeutlicht wurde, dass nur durch eine vertikal vernetzte Gesamtbetrachtung mit weiten Systemgrenzen die Forderung nach hoher Rechenleistung mit gleichzeitig geringer Verlustleistung in Einklang gebracht werden kann.

In diesem Kapitel wird die praktische Realisierung eines neuen Mikroprozessormoduls gezeigt. Dabei werden - im Sinne der vernetzten Gesamtbetrachtung - die Möglichkeiten zur Verlustleistungsreduktion auf der Baugruppen- und Schaltkreisebene konsequent umgesetzt.

Hauptproblem bei einer Realisierung in Hardware stellen die Inkompatibilitäten zwischen einzelnen Bauelementen dar, indem entweder die Schnittstellen verschiedener Hersteller nicht aufeinander abgestimmt sind oder unterschiedliche Logikpegel für die Signalleitungen zum Einsatz kommen.

6.1 Bausteinauswahl

Eine zentrale Voraussetzung für Low Power Design ist die Auswahl geeigneter Bausteine, die nach den Grundsätzen aus Kapitel 3.3 für geringe Stromaufnahme konzipiert sind. Insbesondere sind viele Hersteller wegen der Anforderungen des Marktes für mobile Endgeräte (z. B. Handys, Organizer, PDAs) dazu übergegangen, verstärkt Low Power Produkte anzubieten.

6.1.1 Mikroprozessor

Der Mikroprozessor ist zentraler Baustein eines Systems und beeinflusst damit auch das gesamte Systemdesign nachhaltig. Eine Auswahl muss daher besonders

sorgfältig erfolgen. Selbst die Wahl des Betriebssystems ist stark von den Eigenschaften des eingesetzten Mikroprozessors abhängig. Für jedes Betriebssystem existiert eine beschränkte Liste unterstützter Mikroprozessoren.

Leistungsverbrauch

Die Erfahrung lehrt, dass der Anteil des Mikroprozessors an der gesamten Stromaufnahme eines Integrierten Sensorsystems, etwa die Hälfte beträgt. Aufgrund der Anforderung eines Leistungsbedarfes von $P = 2W$ für das Gesamtsystem darf der gewählte Mikroprozessor nicht mehr als $P = 1W$ Leistung verbrauchen.

Definition einer neuen Kenngröße für die Performance

Der Performancevergleich zwischen unterschiedlichen Mikroprozessorfamilien stellt ein grundlegendes Problem dar, da sich die architekturspezifischen Merkmale einzelner Mikroprozessoren nicht auf einfache Performancegrößen reduzieren lassen [19]. Einfache Größen wie z. B. M in MIPS gelten als Richtwert und sind von Fall zu Fall zu relativieren [14, 15].

Die lt. Datenblätter angegebenen MIPS stellen aus Marketinggründen stets einen theoretischen Maximalwert dar, unter Ausnutzung aller parallel ausführbaren Maschinenbefehle. Moderne DSPs sind aufgrund ihrer parallelen Architektur hier besonders auffallend. Sie verfügen über hohe MIPS-Werte, da sie besonders für die parallele Abarbeitung von signalverarbeitungstheoretischen Algorithmen (z. B. FFT - Fast Fourier Transform) konstruiert sind, wo die gleichzeitige Ausführung von Multiplikation, Addition, Subtraktion und Datamove in einem Befehlsschritt von enorm hohem Geschwindigkeitsvorteil sind.

DSP-Programme können im allgemeinen die speziellen Merkmale einer DSP-Architektur kaum ausnützen. In der Regel sind auch die verfügbaren C-Compiler (selbst im Optimierungsmodus) nicht in der Lage, die Parallelarchitektur effizient zu nützen, sodass man bei hochoptimierten Programmen auf händische Assembler-Codierung angewiesen ist.

Um einen objektiven Vergleich der Verarbeitungsgeschwindigkeit ohne architekturspezifische Merkmale zwischen Mikroprozessoren und DSPs zu ermöglichen, wird in dieser Arbeit eine neue Kenngröße M' eingeführt

> *Definition: Die Performancegröße M' in gpMIPS („General Purpose“ MIPS) ist gleich der Anzahl ausgeführter Maschinenbefehle in Millionen pro Sekunde, ohne Benutzung spezieller architekturspezifischer Prozessorstrukturen.*

Unter Verwendung dieser Kenngröße M' erhält man den bezogenen Leistungsverbrauch

$$p' = \frac{P}{M'} \tag{6.1}$$

Prozessor	Hersteller	Typ	Clock	MIPS	gpMIPS	P/W	MMU	FPU	Linux
StrongARM	Intel	ARM	206MHz	235	206	0,4	ja	nein	ja
SHARC 21160	AD	DSP	100MHz	600	100	2	nein	ja	nein
TigerSHARC	AD	DSP	150MHz	900	150	5	nein	ja	nein
XScale 80200	Intel	ARM	600MHz	750	600	0,5	ja	nein	ja
TM3200 Crusoe	TransMeta	X86	700MHz	700	700	2	ja	ja	ja
Pentium, mobile	Intel	X86	1000MHz	1000	1000	20	ja	ja	ja
TMS320 C6701	TI	DSP	167MHz	1000	167	1,8	nein	ja	nein
TMS320 C6203	TI	DSP	300MHz	2400	300	1,5	nein	nein	nein
TMS320 VC5510	TI	DSP	160MHz	320	160	0,08	nein	nein	nein
VR10000	NEC	µP	300MHz	360	300	30	ja	ja	nein
SH7750 (SH4)	Hitachi	µP	200MHz	360	200	3	ja	ja	nein
Carmel	Infineon	DSP	250MHz	3500	250	0,25	nein	nein	nein
MachZ	ZF	x86 SOC	100MHz	100	100	0,96	ja	ja	ja
ElanSC520	AMD	x86 SOC	133MHz	133	133	2	ja	ja	ja
Geode XLV	Nat Semi	x86 SOC	266MHz	266	266	2,5	ja	ja	ja
MPC7400	Motorola	PowerPC	500MHz	917	500	5	ja	ja	ja
MCF5407	Motorola	68k	167MHz	257	167	0,8	nein	nein	ja
MSC8101	Motorola	DSP	300MHz	1200	300	0,5	nein	nein	nein
EP7211	Cirrus	ARM	75MHz	100	75	0,17	ja	nein	ja

Tabelle 6.1: *Gegenüberstellung von Mikroprozessoren und DSPs.* [79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95] Angeführt sind handelsübliche Prozessoren, wobei jene hervorgehoben sind, die weniger als 1W Leistung benötigen. Die Tabelle zeigt Hersteller, Prozessorfamilie, Taktfrequenz (Clock), Performancegrößen (M in MIPS, M' in gpMIPS), Leistungsverbrauch in W, sowie Vorhandensein einer MMU bzw. FPU und einer etwaigen Unterstützung durch das Betriebssystem Linux.

Entscheidung für den Intel XScale 80200

Tab. 6.1 gibt einen Überblick über derzeit handelsübliche Mikroprozessoren und DSPs. Neben der Bezeichnung für den Prozessor ist die Herstellerfirma, die Prozessorfamilie, die Taktfrequenz (Clock), Performancegrößen M in MIPS und M' in gpMIPS, Leistungsverbrauch in W, Vorhandensein einer Memory Management Unit (MMU), Vorhandensein einer Floatingpoint Unit (FPU) und die Unterstützung durch das Betriebssystem Linux angegeben. Es sind jene Prozessoren mar-

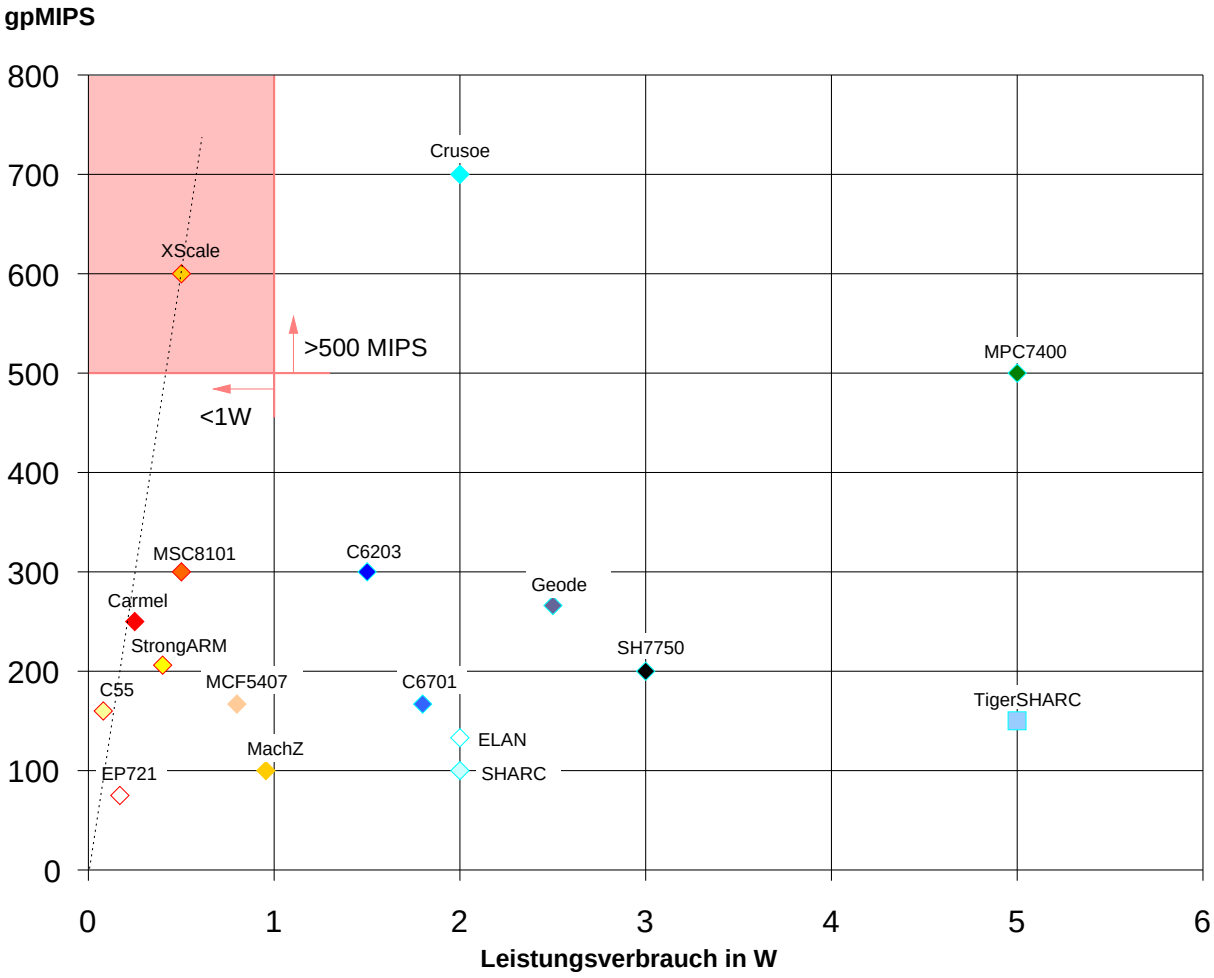


Abbildung 6.1: *Gegenüberstellung: Performance und Leistungsverbrauch für verschiedene Mikroprozessoren.* Die Abb. zeigt die Performancegröße gpMIPS und den Leistungsverbrauch für die Prozessoren aus Tab. 6.1. Die Steigung der punktierten Geraden zeigt für den Intel XScale 80200 die bezogene Performancegröße $1/p'$ in gpMIPS/mW

kiert, die weniger als 1W Leistung verbrauchen.

In der Abb. 6.1 sind Verlustleistung P und Performancegröße M' für die Prozessoren aus Tab. 6.1 gegenübergestellt. Zieht man für die Selektion eines Prozessors die Randbedingung aus Kap. 2, $M' \geq 500gpMIPS$, und einen möglichst geringen bezogenen Leistungsverbrauch p' heran, so fällt die Wahl auf den Intel XScale 80200 Prozessor, der als derzeit einziger Mikroprozessor den Forderungen der Aufgabenstellung gerecht wird.

Der Intel XScale ist die neueste Prozessorarchitektur der Fa. Intel, die speziell für den Einsatz in Low Power Systemen entwickelt wurde. Der XScale ist als Nachfolger zur früheren StrongARM-Familie zu sehen und stellt diese im Hinblick auf seine Kenngrößen (Taktfrequenz bis 733MHz, $P < 1$W, Versorgungsspannung 0.9-1.5V) weit in den Schatten [43, 44, 45, 46, 96, 97, 98].

6.1.2 RAM Speicher

Die in Kap. 2 geforderten 64Mbyte RAM Speicher machen den Einsatz von schnellem SRAM unmöglich, da diese einerseits in zu geringen Speicherkapazitäten vorliegen und andererseits SRAM-Zellen zu hohe statische Stromaufnahme aufweisen. Man ist daher gezwungen, die in ihrer Handhabung wesentlich aufwendigeren und komplexeren DRAM-Speicher einzusetzen. DRAM ist in großen Kapazitäten erhältlich und in Form synchroner DRAM-Speicher (SDRAM) wegen der hohen Stückzahlen am PC-Markt sehr kostengünstig erhältlich.

RAM−Baustein	Hersteller	Typ	Kapazität	Aufbau	Trd	Vcc	P
HYB25L128160AC−8	Infineon	Mobile SDRAM	128 Mbit	8Mx16	10ns	1.8/2.5V	375mW
HYB39S256160CT8	Infineon	SDRAM	256 Bbit	16Mx16	10ns	3.3V	550mW
K4S2A1634D−RS75	Samsung	Mobile SDRAM	256 Mbit	16Mx16	10ns	1.8/2.5V	165mW
K4S561632B	Samsung	SDRAM	256 Mbit	16Mx16	10ns	3.3V	480mW
HM52256457−B60	Hitachi	SDRAM	256 Mbit	1Mx64	7.5ns	3.3V	890mW
EDS2516APSA	Elpida(NEC)	SDRAM	256 Mbit	16Mx16	7.5ns	3.3V	480mW
TC59SM816BFT−75	Toshiba	SDRAM	256 Mbit	16Mx16	7ns	3.3V	330mW

Tabelle 6.2: *Gegenüberstellung: SDRAM Bausteine.* Die Tab. zeigt SDRAM Bausteine verschiedener Hersteller. Markiert sind jene Bausteine, die sich speziell für Low Power Systeme eignen. Neben Bauteilcode, Hersteller und Typ sind die Speicherkapazität in Mbit, die Speicherorganisation, Burst Read Time T_{rd}, Versorgungsspannung und Leistungsverbrauch angegeben. [99, 100, 101, 102, 103, 104, 105]

Die Tab. 6.2 stellt verschiedene SDRAM-Bausteine gegenüber. Neben der Bau-

teilnummer sind der Hersteller, SDRAM-Type, die Speicherkapazität in Mbit, die Speicherorganisation, Burst Read Time T_{rd} , Versorgungsspannung und Leistungsverbrauch angegeben.

In der Tab. 6.2 sind jene Bausteine markiert, die von den Herstellern speziell für den Einsatz in Low Power Systemen entwickelt wurden [99, 101]. Diese Bausteine weisen niedrige Versorgungsspannungen (1.8V bzw. 2.5V) und geringe Stromaufnahme auf. Zusätzlich verfügen sie über Low Power Modi und temperaturgesteuertes Refreshverhalten.

Aufgrund der hervorragenden Verbrauchswerte wird der Samsung-K4S2A1634D für die Entwicklung des neuen Mikroprozessormoduls herangezogen. Zur Zeit ist er allerdings nur in Form von Mustern erhältlich, da es sich um ein neues Produkt der Fa. Samsung handelt.

6.1.3 ROM/Flash Speicher

Zum permanenten Speichern von Programmen und Daten werden in Embedded Systemen üblicherweise nicht flüchtige Flash Speicher eingesetzt. Obwohl mittlerweile kleinste Festplatten mit Kapazitäten über 1Gbyte in der Größe einer Zündholzschachtel verfügbar sind [106], sind aufgrund der mechanischen Stoßfestigkeit und der relativ hohen Stromaufnahme, Flashdisks bzw. Compact Flash Karten [106] zu bevorzugen.

Man unterscheidet zwei verschiedene Typen von Flash Speicher [108]:

- NOR Flash: Die Speicherzellen liegen in Bezug auf die Bitleitungen parallel. Dadurch erhält man hohe Lesegeschwindigkeiten.
- NAND Flash: Es werden mehrere Speicherzellen in Bezug auf die Bitleitung in Serie geschalten. Dadurch sind im Vergleich zum NOR Flash höhere Integrationsdichten möglich (etwa eine Größenordnung). Die Lesegeschwindigkeit wird jedoch durch die Serienschaltung deutlich reduziert (etwa zwei Größenordnungen).

Für das neue Mikroprozessormodul wird Flash-Speicher direkt in den Adressraum des Prozessors eingebunden. Dies ermöglicht schnellstmöglichen, wahlweisen Direktzugriff auf die abgelegten Daten, im Gegensatz zum langsameren, seriellen Zugriff über IDE-Kanäle bei Festplatten bzw. Compact Flash Karten. Darüber hinaus wird dadurch auch das Ausführen von Programmcode direkt aus dem Flash Speicher (Execute In Place, XIP) ermöglicht, wodurch ein vorangehendes Laden in den RAM Speicher entfällt.

NAND Flashes werden wegen der geringen Lesegeschwindigkeit nur seriell ausgelesen (z. B. in Blöcken von 512byte). Sie kommen in erster Linie in Compact Flash Karten zum Einsatz. Für den wahlfreien Zugriff im Adressraum des Prozessors

sind NOR Flashes ideal, weshalb sie für das neue Mikroprozessormodul verwendet werden.

Flash−Baustein	Hersteller	Typ	Kapazität	Aufbau	Tacc	Trd	Vcc	P
28F128W18	Intel	NOR	128 Mbit	8Mx16	90ns	20ns	1,8V	12mW
28F128J3A	Intel	NOR	128 Mbit	16Mx8/8Mx16	150ns	25ns	2,7−3,6V	80mW
Am29BDS643D	AMD	NOR	64 Mbit	4Mx16	90ns	20ns	1,8V	
MBM29PDS322	Fujitsu	NOR	32 Mbit	2Mx16	120ns		1,8V	121mW
LHF64F09	Sharp	NOR	64 Mbit	4Mx16	90ns	35ns	2,7−3,6V	
K9K1G08U0M	Samsung	NAND	1 Gbit	128Mx8	12µs	50ns	2,7−3,6V	33mW
TH58100FT	Toshiba	NAND	1 Gbit	128Mx8	25µs	50ns	2,7−3,6V	33mW
TC58FVT641	Toshiba	NOR	64 Mbit	4Mx16/8Mx8	100ns	100ns	2,7−3,6V	100mW

Tabelle 6.3: *Gegenüberstellung: Flash Bausteine.* Die Tab. zeigt Flash Bausteine verschiedener Hersteller. Markiert sind jene Bausteine des NOR-Typs mit der höchsten Speicherkapazität. Neben Bauteilcode, Hersteller und Typ sind die Speicherkapazität in Mbit, die Speicherorganisation, Zugriffszeit T_{acc}, Burst Read Time T_{rd}, Versorgungsspannung und Leistungsverbrauch angegeben. [109, 110, 111, 112, 113, 114, 115, 116]

Die Tab. 6.3 stellt verschiedene Flash-Bausteine einander gegenüber. Aufgelistet sind Bauteilnummer, Hersteller, Type, die Speicherkapazität in Mbit, die Speicherorganisation, Zugriffszeit T_{acc} (für den ersten Zugriff), Burst Read Time T_{rd} (für Folgezugriffe), Versorgungsspannung und Leistungsverbrauch.

In der Tab. 6.3 sind jene Flash-Bausteine vom Typ NOR markiert, welche die höchsten Speicherkapazitäten aufweisen. Für das neue Mikroprozessormodul wird der Intel 28F128W18 mit einem hervorragend niedrigem Leistungsverbrauch, hoher Speicherkapazität und Zugriffsgeschwindigkeit ausgewählt. Grund für diese ausgezeichneten Kennwerte ist die spezielle Auslegung des Intel 28F128W18 für tragbare elektronische Geräte [117].

6.1.4 Logikbaustein - ASIC

Die Aufgabe eines Logikbausteines[1] ist, ein reibungsfreies Zusammenspiel von Prozessor, Speicherbausteinen und Ein-/Ausgabekomponenten zu gewährleisten. Für das neue Mikroprozessormodul hat der Logikbaustein folgende Aufgaben zu übernehmen:

- SDRAM Controller

[1]bei Arbeitsplatz PCs spricht man vom sog. Chipsatz

- Flash Controller
- DMA Controller
- Interrupt Controller
- Power Management Funktionen, Low Power Modi
- UART für serielle Schnittstellen
- Timer
- Pegelwandlung für unterschiedliche Logikfamilien (3.3V, 2.5V, 1.8V, 1.5V)
- Allgemeine Logikfunktionen (z. B. Test-Interfaces)

Für Prototypen- und Kleinserien werden diese Aufgaben von einem programmierbaren FPGA Baustein übernommen. Bei Stückzahlen ab etwa 2000 lohnt sich der Einsatz von Mask Grid Arrays (MGA). Für Großserien ab 20000 Stück sind Cell Based Integrated Circuits (CBIC) oder Full Customized ASICs wirtschaftlich sinnvoll, wie die Abb. 6.2 verdeutlicht [118, 119].

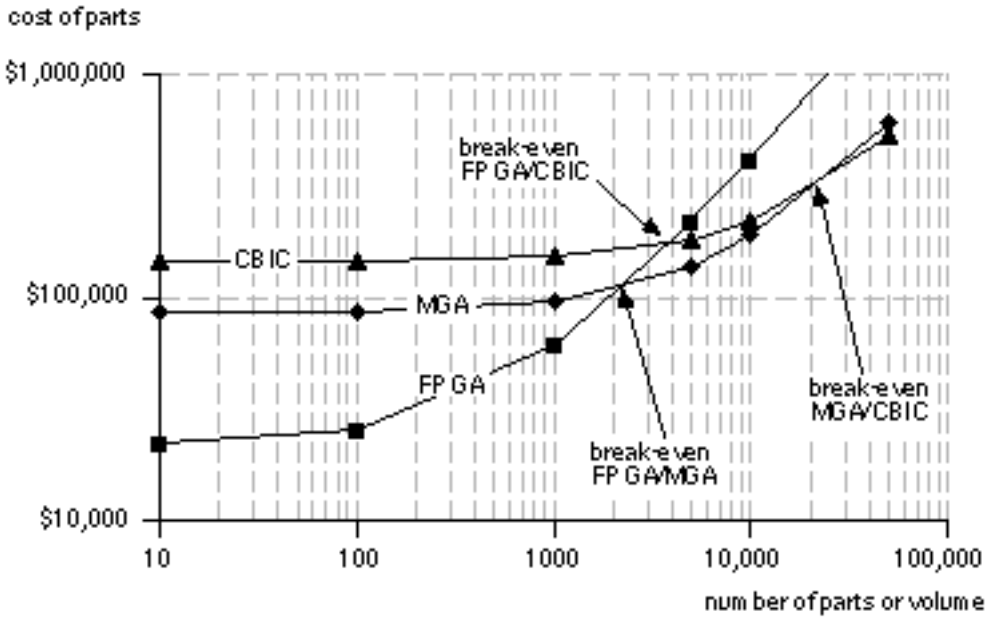


Abbildung 6.2: *Kostenvergleich für ASIC-Varianten aus [118].* Dargestellt sind typische Gesamtkostenverläufe für unterschiedliche ASIC-Varianten in Abhängigkeit von der Stückzahl. Eingetragen sind die Break-Even Punkte, anhand derer sich die Wirtschaftlichkeit einer Variante beurteilen lässt.

Der Einsatz eines FPGA ist durch die unverzichtbar hohe Flexibilität während der Prototypenphase notwendig. Nicht nur aus wirtschaftlichen Gründen, sondern auch im Hinblick auf die Stromaufnahme, sind für spätere Produktzyklen CBICs bzw. Full Customized ASICs zu bevorzugen, wie die quantitative Abschätzung aus Abb. 6.3 zeigt [120, 121, 122].

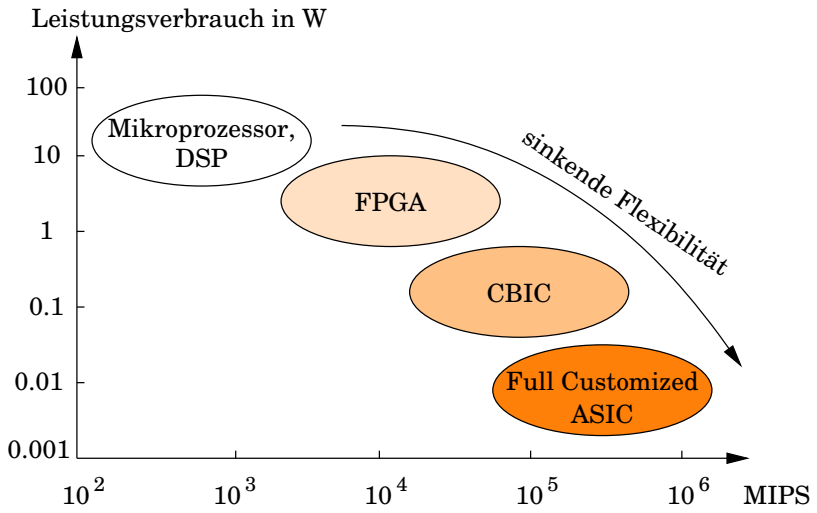


Abbildung 6.3: *Leistungsverbrauch und Performance für ASIC-Varianten.* Die Abb. zeigt eine quantitative Abschätzung für den Leistungsverbrauch in W und die zugehörige erreichbare Rechenleistung in MIPS.

Für die Auswahl eines geeigneten FPGA Bausteins ist aus der oben dargestellten Anforderungsliste einerseits eine Abschätzung des benötigten Logikaufwandes (z. B. Anzahl benötigter Logikgatter) und andererseits eine Abschätzung der notwendigen Ein- und Ausgänge (Anzahl I/O-Pins) durchzuführen.

Die Ermittlung der erforderlichen Anzahl von Logikgattern ist ein schwieriger Schätzprozess, der viel Erfahrung erfordert und durch bestehende Referenzimplementierungen unterstützt werden muss. Es ist ausreichend Spielraum für Schätzunsicherheiten und evtl. später implementierten Erweiterungen vorzusehen. Die Tab. 6.4 zeigt eine Abschätzung des Logikaufwandes für das neue Mikroprozessormodul.

Die Anzahl benötigter Ein- und Ausgänge des FPGA kann durch Betrachtung einzelner, überschaubarer Teilaufgaben bzw. Signalgruppen erfolgen, für die jeweils eine Einzelabschätzung der benötigten Pinanzahl erfolgt. In der Tab. 6.5 ist diese Abschätzung durchgeführt.

Neben der erforderlichen Anzahl von Gattern und Ein- bzw. Ausgängen sind gemäß der oben angeführten Anforderungen weitere Randbedingungen, wie Möglichkeit

Funktion	Anzahl Gatter
SDRAM−Controller	20k
Flash−Controller	15k
UART (2x)	20k
Div. Logikfunktionen	10k
Power Management	10k
Interrupt−Controller	5k
Gesamt	**80k**

Tabelle 6.4: *Abschätzung der erforderlichen Logikgatter.* Die Abschätzung erfolgt aufgrund von Erfahrungswerten und bestehenden Referenzimplementierungen.

Signalgruppe	Anzahl Pins
Datenbus Prozessor	64
Adressbus Prozessor	16
Requestbus Prozessor	16
Memory Datenbus	64
Memory Adressbus	23
Memory Controlsignale	26
UART	4
Standby Signale	4
Sonstige	8
Gesamt	**225**

Tabelle 6.5: *Abschätzung der erforderlichen Anzahl von Ein- und Ausgängen (Pins) am FPGA.* Die Abschätzung erfolgt durch Betrachtung von Teilaufgaben bzw. Signalgruppen, für die sich die Anzahl der Ein- und Ausgänge überschaubar ermitteln lässt.

der Pegelwandlung, Leistungsverbrauch und Versorgungsspannung bei der Bausteinwahl zu berücksichtigen.

Am Institut für Automatisierungs- und Regelungstechnik werden seit mehreren Jahren FPGAs der Fa. Xilinx mit Erfolg eingesetzt [123]. Um diese Erfahrung zu nutzen und somit die Entwicklung des neuen Mikroprozessormoduls zu beschleunigen, wird ein Logikbaustein aus dieser Produktfamilie gewählt. Insbesondere weisen Virtex-II FPGAs spezielle Eigenschaften wie Pegelwandlung und unterschiedlichste Versorgungsspannungsebenen auf. Die niedrige Kernspannung von 1.5V sorgt zusätzlich für einen geringen Leistungsverbrauch [124, 125].

Die Tab. 6.6 listet Xilinx Virtex-II FPGAs mit einer Gatteranzahl zwischen 80000 und einer Million. Gemäß der Abschätzung des Logikgatterbedarfes aus Tab. 6.4 wäre der XC2V80 ohne Reserve gerade ausreichend. Es zeigt sich aber, dass die Anzahl erforderlicher Ein- bzw. Ausgänge aus der Tab. 6.5 die maßgebende Größe für die Bausteinauswahl darstellt. Aus diesem Grunde muss der (gemessen an der Logikkapazität überdimensionierte) XC2V500 mit 264 Ein- bzw. Ausgängen und 500.000 Gattern gewählt werden.

Type / Gehäuse	Anzahl Ein− bzw. Ausgänge			
	XC2V80	XC2V250	**XC2V500**	XC2V1000
CS144	92	92		
FG256	120	172	172	172
FG456		200	**264**	324
FG676				
FF896				432

Tabelle 6.6: *Xilinx Virtex-II FPGA Bausteine.* Die Tabelle zeigt Logikbausteine mit einer Gatteranzahl zwischen 80.000 und einer Million. Verfügbare Gehäusevarianten bieten bis zu 432 Ein- bzw. Ausgänge.

6.1.5 Bluetooth Funkschnittstelle

Für die Übertragung von Berechnungsergebnissen und Diagnoseinformationen wird für das Mikroprozessormodul eine standardisierte Bluetooth Funkschnittstelle implementiert. Bluetooth bietet eine Reihe von Vorteilen gegenüber herkömmlichen drahtgebundenen Übertragungsverfahren:

- geringer Platzbedarf

- bequem und benutzerfreundlich da keine Kabel erforderlich sind; ein enormer Vorteil bei schwer zugänglichen Geräten
- preisgünstig
- hohe Datenübertragungsbandbreite von ca. 1Mbit/s (das Zehnfache einer seriellen Schnittstelle)
- geringer Leistungsverbrauch im mW-Bereich bei 10m Reichweite

Mit der Implementierung der Bluetooth-Schnittstelle beschäftigt sich die Diplomarbeit von Robert Jelinek [172].

6.2 Systemaufbau

6.2.1 Blockschaltbild

Der Aufbau des Mikroprozessormoduls ist in Form eines Blockschaltbildes in der Abb. 6.4 dargestellt. Man erkennt, dass dem eingesetzten Logikbaustein eine zentrale Rolle im System zukommt, da es seine Aufgabe ist, den Prozessor mit allen Speicherkomponenten und Ein/Ausgabeschnittstellen zu verbinden.

Der Logikbaustein stellt eine Reihe von Signalen zur Verfügung, die mittels Logikanalysator und Oszilloskop für Diagnosezwecke genutzt werden können. Zwei JTAG Schnittstellen [126] ermöglichen das Debuggen über den Prozessor einerseits und das Konfigurieren des FPGA andererseits. Weiters stehen für die Steuerung des Mikroprozessormoduls und den interaktiven Betrieb mit einem Benutzer zwei serielle Schnittstellen zur Verfügung.

Abhängig vom Betriebs- bzw. Belastungszustand greift der Logikbaustein direkt in die Spannungsversorgung ein, um die jeweils erforderliche elektrische Leistung bei optimalem Wirkungsgrad bereit zu stellen. Neben einer Anpassung der Versorgungsspannung können auf diesem Weg auch die Regelungsalgorithmen der Spannungssteller beeinflusst werden.

Der Logikbaustein stellt darüber hinaus auch Schnittstellen für Sensor- und Aktorsysteme zur Verfügung. Diese Schnittstellen sind aufgrund der Programmierbarkeit des FPGA auf das jeweilige Einsatzgebiet des Mikroprozessormoduls anzupassen und stellen so erhebliche Flexibilitäten hinsichtlich der Anschließbarkeit unterschiedlichster Sensor/Aktorsysteme bereit. Auf diese Weise ist sowohl der Anschluss einer Kamera mit hoher Datenübertragungsrate als auch der Anschluss einer einfachen Lichtschranke möglich.

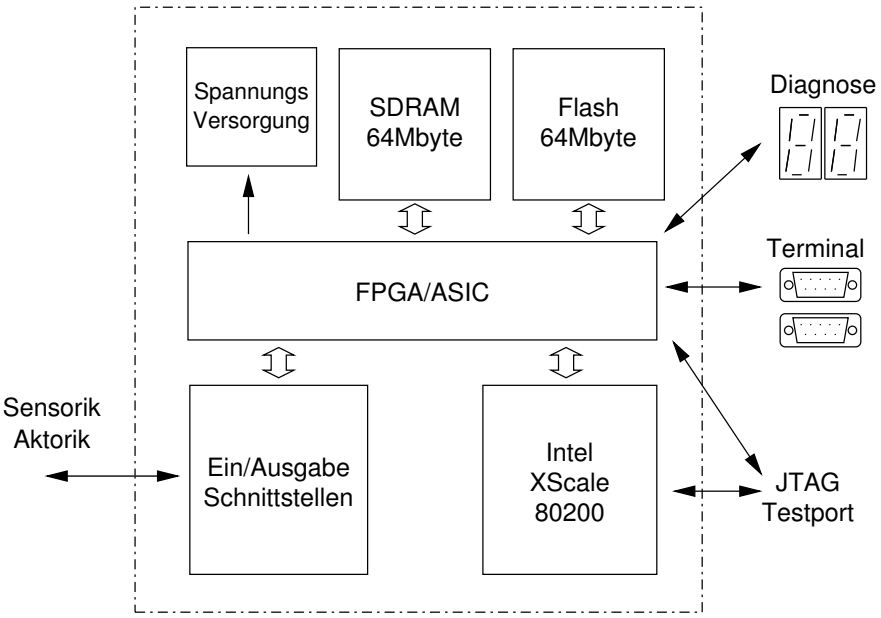


Abbildung 6.4: *Blockschaltbild des neuen Mikroprozessormoduls.* Dargestellt sind Prozessor (Intel XScale 80200), Speicher (SDRAM und Flash), Spannungsversorgung, Logikbaustein (FPGA) und externe Schnittstellen. Ein Terminal ermöglicht die Steuerung des Moduls. Zusätzliche Steckverbinder ermöglichen Tests und Fehlerdiagnosen.

6.2.2 Blockschaltbild FPGA

Der Logikbaustein (FPGA) übernimmt wichtige, zentrale Funktionen im Mikroprozessormodul. Diese sind schematisch im Blockschaltbild der Abb. 6.5 dargestellt.

Kernkomponente im FPGA ist der Memorycontroller. Er reiht Zugriffe auf die verschiedenen Speicherbereiche in eine Warteschlange ein, aus der sie der Reihe nach von funktionsspezifischen Request-Prozessoren (SDRAM, Flash und Ein-/Ausgabe) abgearbeitet werden.

Ein Pegelkonverter dient zur Umwandlung der Datensignale zwischen den unterschiedlichen Spannungsebenen (3.3V und 1.8V). Diese Pegelwandlung ist notwendig, da der Prozessor und die Speicherbausteine wegen des Energieverbrauchs auf unterschiedlichen Versorgungsspannungsebenen betrieben werden.

6.2.3 Versorgungsspannungen

Betrachtet man den dominierenden ersten Term aus Gl. (3.2)

$$P \sim \alpha f C U^2 \tag{6.2}$$

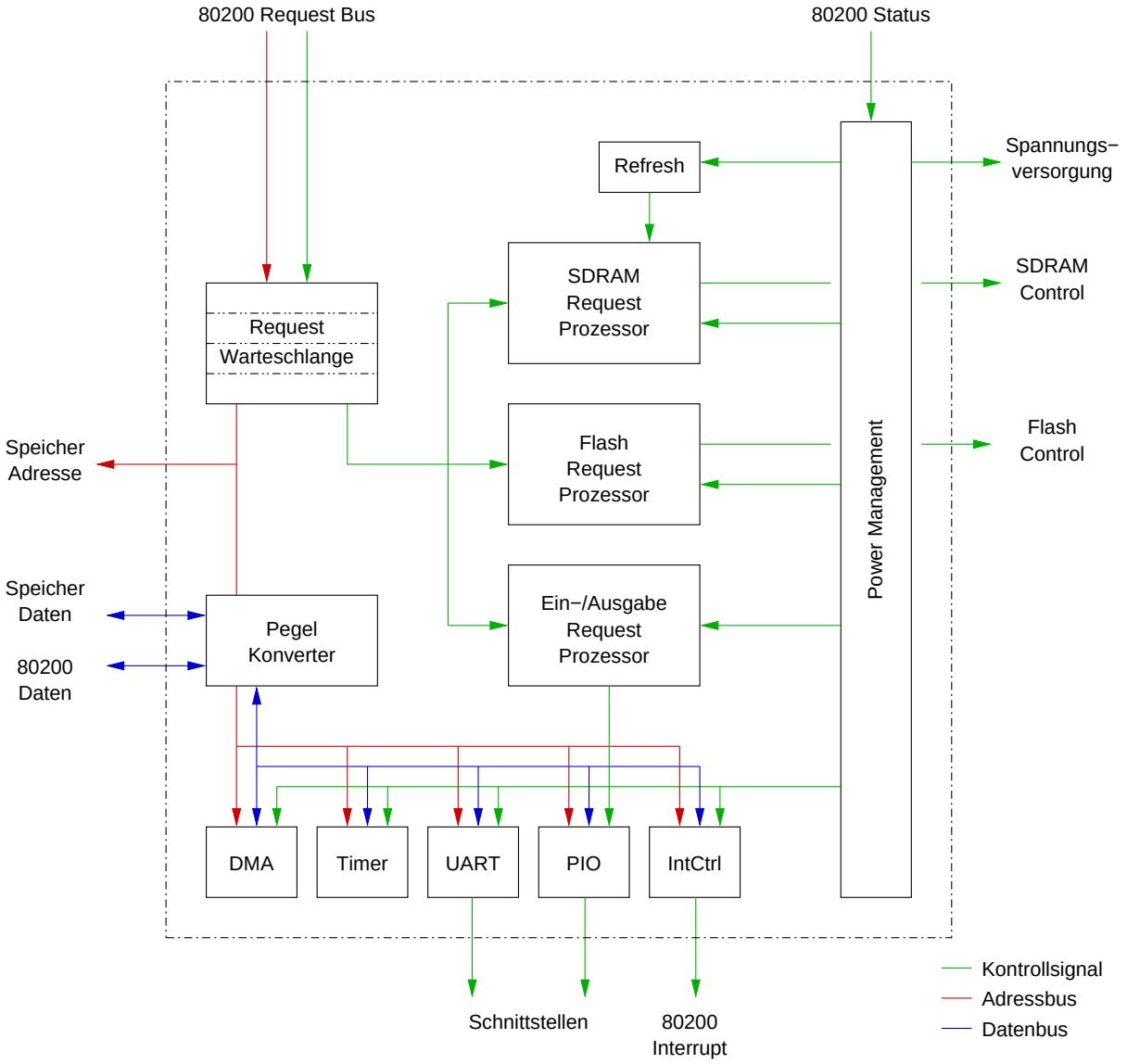


Abbildung 6.5: *Blockschaltbild des FPGA-Designs.* Speicherzugriffe des Prozessors gelangen in eine Warteschlange und werden in spezifischen Requestprozessoren abgearbeitet. DMA-Controller, Timer, UART, PIO und Interruptcontroller unterliegen der Kontrolle des Ein-/Ausgabe Requestprozessors. SDRAM und Flash sind an jeweils eigene Requestprozessoren angekoppelt. Ein Pegelkonverter ermöglicht das Umsetzen der für Prozessor und Speicher unterschiedlichen Logikpegel. Das Power Management bildet eine zentrale Komponente, die in Low Power Betriebszuständen für die Abschaltung nicht benötigter Komponenten sorgt.

so erkennt man, dass die Versorgungsspannung quadratisch in den Leistungsverbrauch einer CMOS Logikschaltung eingeht. Eine Reduktion der Versorgungsspannung im Rahmen der in Kapitel 3.3 dargestellten Grenzen ist daher eine sehr wirkungsvolle Maßnahme zur Einsparung von Verlustleistung.

Bezieht man das ebenenübergreifende Low Power Design Paradigma aus Abb. 5.1 in diese Überlegung mit ein, ergibt sich die Konsequenz, sämtliche Bausteine mit möglichst niedrigen Versorgungsspannungen zu betreiben. Am Markt befinden sich zur Zeit Bausteine mit Versorgungsspannungen zwischen 1.5V und 5V. Dies führt zwangsläufig dazu, dass am Mikroprozessormodul unterschiedliche Spannungsniveaus zur Verfügung gestellt werden müssen. Für das hochperformante Mikroprozessormodul kommen die in Tab. 6.7 angeführten Spannungsebenen zum Einsatz.

Baustein	3.3V	2.5V	1.8V	1.5V
Intel XScale Prozessor	x			x
Xilinx Virtex II FPGA	x		x	x
Samsung Mobile SDRAM		x	x	
Intel Wireless Flash			x	

Tabelle 6.7: *Versorgungsspannungen im neuen Mikroprozessormodul.* Um möglichst geringe Verlustleistungen zu erhalten, wird die Versorgungsspannung der einzelnen Bauteile möglichst niedrig gewählt. Prozessor und FPGA verwenden für den Betrieb interner Schaltkreise niedrigere Spannungen als für ihre externen Signale.

6.3 Low Power Maßnahmen

Um die aus der Aufgabenstellung geforderten Kenndaten, in erster Linie $P \leq 2$W und $M \geq 500$MIPS, für das neue Mikroprozessormodul zu erreichen, wurde eine Reihe von Maßnahmen ergriffen:

6.3.1 Auswahl von Low Power Hardwarebausteinen

Aufgrund der fehlenden technologischen Möglichkeiten konnte im Rahmen dieser Dissertation auf die im Ebenenmodell der Abb. 5.1 skizzierten Chip- und Transistorebenen kein direkter Einfluss ausgeübt werden. Es ist jedoch möglich, durch

sorgfältige Recherche Hardwarebausteine auszuwählen, in denen die Grundkonzepte eines hochperformanten Low Power Entwurfs durch den Hersteller berücksichtigt sind.

Die Untersuchung des anteiligen Leistungsbedarfes verschiedener Komponenten eines Mikroprozessorsystems (Tab. 5.1) hat gezeigt, dass besonderes Augenmerk auf die Auswahl des Prozessors zu legen ist. Dem wurde durch die Wahl des Intel XScale 80200 Prozessors entsprochen.

Betriebsmodus	Normal	Idle	Drowsy	Sleep
Verlustleistung	1W	0.3W	1mW	10μW
Prozessorstatus bleibt	erhalten	erhalten	erhalten	nicht erhalten
PLL-Zustand	On	On	Off	Off
Wake-Up Ursache		Interrupt	Interrupt	Reset
Wake-Up Dauer		<1μs	<100μs	>10ms

Tabelle 6.8: *Betriebsmodi des Intel XScale 80200 Prozessors.* In den vier Betriebsmodi „Normal", „Idle", „Drowsy" und „Sleep" benötigt der Intel XScale zwischen 1W und 10μW Leistung. Mit zunehmender Leistungseinsparung eines Modus steigt die Dauer einer Transition aus diesem Zustand [98].

Der Intel XScale 80200 ist speziell für den Einsatz in Low Power Applikationen entwickelt worden. Selbst bei einer Taktfrequenz von 600MHz benötigt dieser Prozessor lediglich eine Leistung von 1W. Für die Implementierung von energiesparenden Modi auf Betriebssystemebene bietet der XScale vier unterschiedliche Betriebszustände an, die in der Tab. 6.8 aufgelistet sind. Gegenüber dem „Normal"-Modus mit einem Leistungsbedarf von 1W kann im „Idle"-Modus 70% (Abschaltung der Verarbeitungspipeline) und im „Drowsy"-Modus sogar 99.9% (Abschalten der Taktversorgung durch eine PLL) der Verlustleistung eingespart werden, während der Prozessorzustand vollständig erhalten bleibt. Unvermeidbar ist in diesen sparsamen Betriebszuständen jedoch die angegebene Zeitdauer für eine Transition in den „Normal"-Zustand zurück (sog. wake-up Dauer). Diese ist umso höher, je mehr Leistung eingespart wird bzw. je tiefer der Schlafzustand des Prozessors ist.

Das besondere Leistungsvermögen des Intel XScale zeigt sich bei einer Betrachtung der zulässigen Intervalle für Versorgungsspannung und Taktfrequenz, mit denen dieser Prozessor betrieben werden kann (Tab. 6.9). Diese großen Intervalle reichen von 1.0V bis 1.5V und 200MHz bis 733MHz und haben dem Prozessor aufgrund dieser Skalierbarkeit seinen Namen gegeben.

Die Verlustleistung des Intel XScale teilt sich in zwei wesentliche Komponenten: die Verlustleistung des Prozessorkerns (Core-Verlustleistung) und die Verlustlei-

Prozessor-Variante	333MHz	400MHz	600MHz	733MHz
80200 M733	1.0-1.5V	1.1-1.5V	1.3-1.5V	1.5V
80200 M600	1.1-1.5V	1.3-1.5V	1.5V	
80200 M400	1.1-1.3V	1.3V		

Tabelle 6.9: *Versorgungsspannungen für Varianten des Intel XScale 80200 Prozessors.* Für die Varianten M733, M600 und M400 sind abhängig von der Taktfrequenz unterschiedliche Versorgungsspannungen für den Prozessorkern möglich. In den angegebenen Spannungsbereichen kann die Versorgungsspannung beliebig fein justiert werden [127].

stung in den Treiberstufen für die nach außen geführten Signalleitungen, der sog. I/O-Verlustleistung. Diese beiden Komponenten sind in der Abb. 6.6 in Abhängigkeit von den jeweiligen Versorgungsspannungen dargestellt. Als Parameter tritt die Taktfrequenz auf, mit der der externe Datenbus einerseits (66MHz oder 100MHz) bzw. der Prozessorkern andererseits betrieben werden (400MHz und 600MHz).

Durch Ablesen des in der Abb. 6.6 eingezeichneten Arbeitspunktes für einen 600MHz schnellen Prozessorkern und einen 100MHz getakteten externen Datenbus ergibt sich die Gesamtverlustleistung von rund 1W. Dabei ist anzumerken, dass für die Gewinnung dieser Diagramme hohe Prozessorauslastung während eines Dhrystone-Benchmarks und mittlere Busbelastung zugrundeliegen.

Durch die konsequente Auswahl verlustleistungsarmer Bauteile, bei denen die Hersteller Low Power Prinzipien auf Transistor- und Chipebene berücksichtigt haben, ergibt sich für das neue Mikroprozessormodul eine Gesamtleistungsbilanz gemäß Tab. 6.10, in der für jede Komponente der maximale und minimale Leistungsbedarf im aktiven bzw. energiesparenden Modus angegeben ist.

Damit ergibt sich unter Maximallast eine Verlustleistung von 2.1W für das neue Mikroprozessormodul („worst case“ - Abschätzung). Dabei ist anzumerken, dass im tatsächlichen Betrieb nicht alle Komponenten gleichzeitig ihre Maximallast erreichen können, da z. B. ein gleichzeitiger Zugriff auf SDRAM und Flash unmöglich ist. Daher ist die im realen Betrieb gemessene Verlustleistung von 1.48W geringer.

6.3.2 Steuerung der Versorgungsspannung

Wie in Abschnitt 3.4.1 erläutert, gehört die Steuerung der Versorgungsspannung zu den effektivsten Maßnahmen für die Reduktion von Verlustleistung. Diese Steuerung wird üblicherweise durch das Betriebssystem unter Beurteilung der aktuellen Rechenlast im Prozessor vorgenommen.

Im vorliegenden Anwendungsfall in der rechenintensiven Integrierten Sensorik (z. B. Bildverarbeitung) unterliegt das Rechenlastprofil einem „Stop & Go“ Ver-

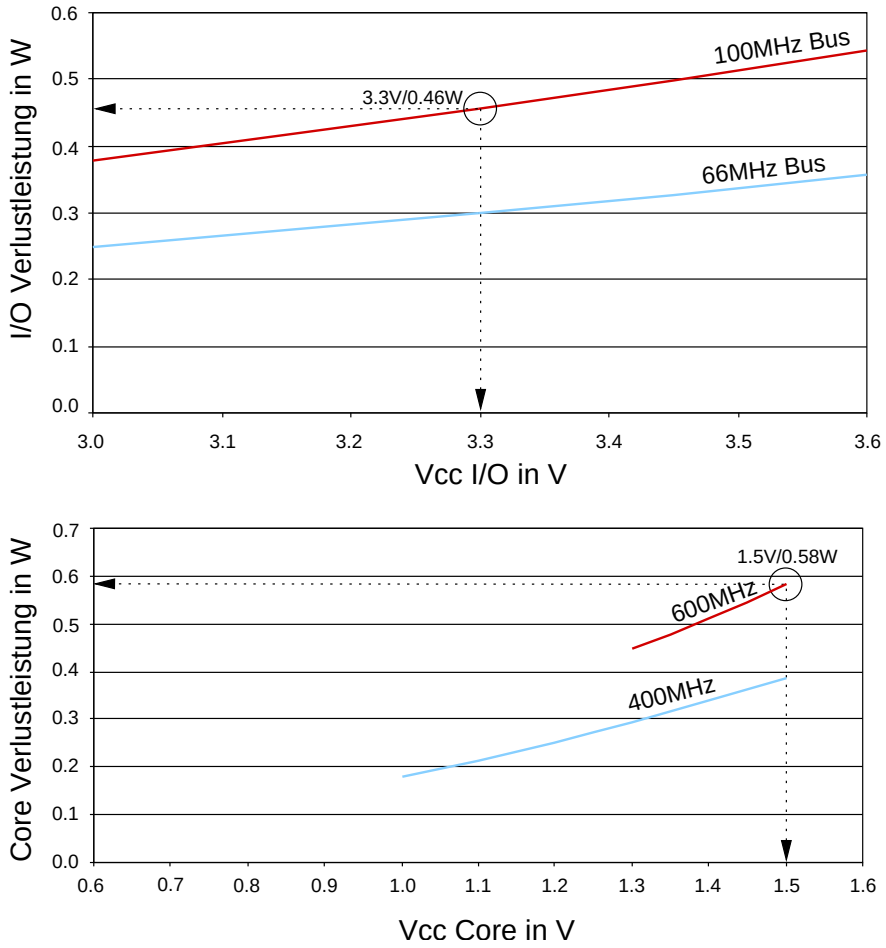


Abbildung 6.6: *Verlustleistung im Intel XScale Prozessor.* Das obere Diagramm stellt die Verlustleistung in den Ein-/Ausgabe-Treibern bei mittlerer Busauslastung, 10pF Leitungskapazität und einem 66MHz bzw. 100MHz Businterface in Abhängigkeit von der Versorgungsspannung „Vcc I/O" der Ein-/Ausgabetreiber dar. Das untere Diagramm zeigt die Verlustleistung im Prozessorkern in Abhängigkeit von der Kern-Versorgungsspannung „Vcc Core" für die Taktfrequenzen 400MHz und 600MHz. Für die eingetragenen Arbeitspunkte bei 600MHz Prozessortakt und 100MHz Businterface ergibt sich ein Gesamtleistungverbrauch von ca. 1W [127].

Komponente	maximal	minimal
Prozessor	1W	1mW
SDRAM	0.45W	17mW
Flash	50mW	50μW
FPGA	0.5W	30mW
sonstige	0.1W	3mW
gesamt	**2.1W**	**51mW**
Messung	**1.48W**	**42mW**

Tabelle 6.10: *Maximale und minimale Verlustleistung im neuen Mikroprozessormodul.* Für die jeweiligen Komponenten sind der Maximalwert im aktiven Betrieb und der Minimalwert im Low Power Modus angegeben. Die letzte Zeile enthält die nach der Realisierung des Mikroprozessormoduls tatsächlich gemessene Leistungsaufnahme.

lauf: entweder liegen Sensordaten vor und müssen schnellstmöglich ausgewertet werden oder das Sensorsystem ist gerade inaktiv. Eine Steuerung der Versorgungsspannung würde in diesem Fall keine Vorteile bringen. Belastungszustände mit mittlerer oder geringer Rechenlast, wie dies bei allgemein einsetzbaren Mikroprozessorsystemen häufig vorkommt (z. B. PC), sind in der Sensorik unbedeutend. Aus diesem Grund wurde auf eine Steuerung der Versorgungsspannung am neuen Mikroprozessormodul verzichtet. Dadurch ergeben sich auch Kostenvorteile und eine geringere Baugröße.

Wichtig ist hingegen, das Modulationsverfahren im DC/DC Versorgungsspannungsregler (Gleichspannungstiefsetzsteller) lastabhängig umzuschalten, um sowohl bei maximaler als auch bei minimaler Stromaufnahme einen bestmöglichen Wirkungsgrad zu erzielen. Bei hohen Strömen wird Pulsbreitenmodulation (Pulse Width Modulation, PWM), bei geringen Strömen Pulspausenmodulation (Pulse Skip Modulation, PSM) angewendet, die im jeweiligen Betriebsbereich den besseren Wirkungsgrad aufweisen (Abb. 6.7).

Das Umschalten zwischen PWM und PSM wird durch den FPGA vorgenommen. Dabei wird der Betriebszustand des XScale-Prozessors überwacht und im Falle eines energiesparenden Zustandes die Versorgungsspannungsregler in den PSM-Modus geschalten. Damit kann der Wirkungsgrad des Versorgungsspannungsreglers bei kleinen Ausgangsströmen erheblich verbessert werden.

6.3.3 Low Power Betriebszustände

Für die Aktivierung von verbrauchsarmen Modi der eingesetzten Hardwarebausteine ist die Modellierung eines Graphen für die Betriebszustände von eminenter

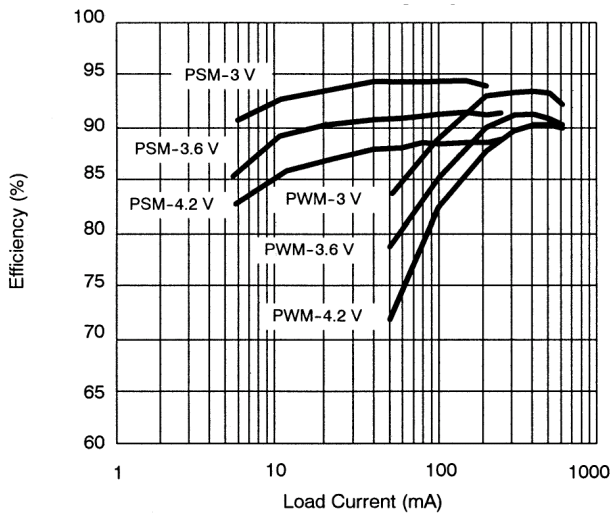


Abbildung 6.7: *Wirkungsgrad des Versorgungsspannungsreglers aus [128].* Dargestellt ist der Wirkungsgrad des DC/DC Schaltreglers in Abhängigkeit vom Ausgangsstrom für unterschiedliche Eingangsspannungen jeweils für die Modulationsarten PWM und PSM. Die geregelte Ausgangsspannung beträgt 2.7V.

Bedeutung. Für ein allgemeines integriertes Sensorsystem lässt sich ein Zustandsgraph gemäß Abb. 6.8 angeben.

Aus dem Betriebszustandsdiagramm lässt sich ohne Schwierigkeiten ableiten, in welchen Situationen das Betriebssystem und der eingesetzte Logikbaustein die Low Power Modi der zugrundeliegenden Hardwarekomponenten aktivieren müssen.

6.3.4 Pufferspeicher

Jede Verarbeitungsaktivität in einem Mikroprozessorsystem hat seinen Ursprung in einer Programmunterbrechnung („Interrupt“). Sind alle Interrupt-Anforderungen abgearbeitet, kehrt das Betriebssystem in den energiesparenden Wartezustand („Idle“-Modus) zurück.

Um Energie einzusparen ist es sinnvoll, mehrere Interrupt-Anforderungen zu puffern und anschließend gemeinsam zu verarbeiten. Typisches Beispiel ist das Puffern von Daten, die von einem Kommunikationskanal eingelesen werden. Es wäre zu aufwendig, den Idle-Modus für jedes einzelne Datenwort zu verlassen. Stattdessen sind in den Empfangseinheiten der Kommunikationskanäle (z. B. in den UARTs) entsprechend große Pufferspeicher vorzusehen, die erst bei Erreichen eines gewissen Füllstandes abgearbeitet werden.

Im Logikbaustein des neuen Mikroprozessormoduls werden für die UARTs der

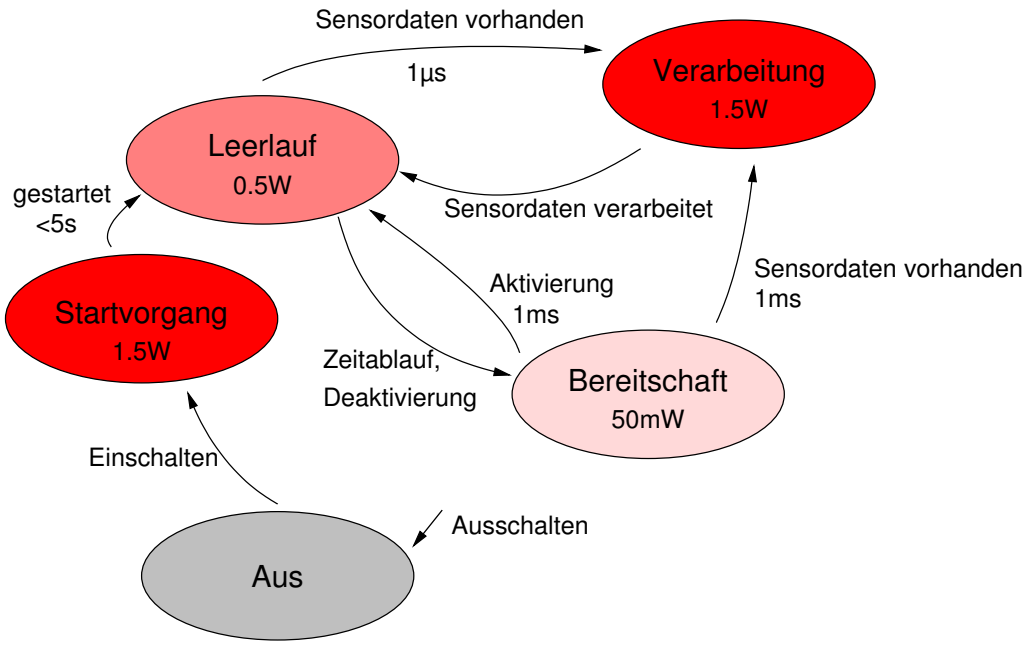


Abbildung 6.8: *Betriebszustände eines Integrierten Sensorsystems.* Nach dem Einschalten und dem Startvorgang befindet sich das Sensorsystem im Leerlauf. Bei Vorhandensein von Sensordaten werden diese verarbeitet und anschließend wieder der Leerlauf aktiviert. Erfolgte nach einer gewissen Zeit keine Verarbeitung wird in den Bereitschaftmodus geschalten. Für die Betriebsmodi ist die auftretende Verlustleistung und die Zeitdauer für eine Transition aus den verbrauchsarmen Zuständen heraus angegeben.

seriellen Schnittstellen Pufferspeicher in der Größe von 1kbyte angelegt (übliche UARTs besitzen 16byte lange Pufferspeicher). Damit kann die Interrupt-Belastung für den Mikroprozessor erheblich reduziert und der Energieverbrauch gesenkt werden.

6.4 PCB Layout

Die Aufgabenstellung eines maximal Scheckkarten-großen Mikroprozessormoduls erfordert den Einsatz von Hardwarekomponenten mit möglichst kleinem Gehäuse. Moderne Ball Grid Array (BGA) Gehäuse eignen sich hervorragend dafür, dieser Randbedingung gerecht zu werden.

Anschlussbelegung des FPGA

Der Einsatz von BGA Gehäusen hat allerdings auch nachhaltigen Einfluss auf den Entwurf eines PCB Layouts, da die extrem hohe Flächendichte von Anschlüssen (>1Pin/mm^2) eine Entflechtung der Signalleitungen auf dem PCB äußerst erschwert. Allein das Anschließen sämtlicher Signalleitungen bedarf genauester Planung um die am PCB vorhandene Fläche möglichst effizient zu nutzen. Die Abb. 6.9 zeigt für den eingesetzten FPGA mit einem 456-poligem BGA-Gehäuse einen Anschlussplan auf drei Signalebenen.

Trotz eines genauen Anschlussplans gibt es bei einer entsprechend hohen Anzahl von Signalleitungen erheblichen Entflechtungsbedarf. Dieser lässt sich verringern, indem die Anschlussbelegung am FPGA den Erfordernissen der Entflechtung angepasst wird, d. h. die Signalleitungen werden so am FPGA angeschlossen, dass ein Minimum an Auskreuzungen erforderlich wird. Am FPGA ergibt sich dadurch eine scheinbar wahllose Belegung der Anschlüsse, die aber wegen des flexiblen Routingkonzepts innerhalb des FPGA keine Rolle spielt.

Testgerechter Layout-Entwurf

Der Einsatz von BGA-Technologie erschwert den messtechnischen Zugang zu den Signalleitungen am PCB, da die Bauteile ihre Anschlüsse abdecken und somit nicht zugänglich sind. Für die Inbetriebnahme sind jedoch umfangreiche Messungen unumgänglich, um Probleme in der Funktionalität oder Signalintegrität lösen zu können.

Beim Entwurf des neuen Mikroprozessormoduls wurde auf diese Einschränkung durch die BGA-Gehäuse von vorne herein Rücksicht genommen, indem wichtige Signale in Form von Steckerreihen für messtechnische Zwecke zugänglich gemacht werden. Nachdem jeder zusätzliche Stecker Platz benötigt und nach der Inbetriebnahmephase ohnehin nicht mehr benötigt wird, werden alle diese Stecker am Rand

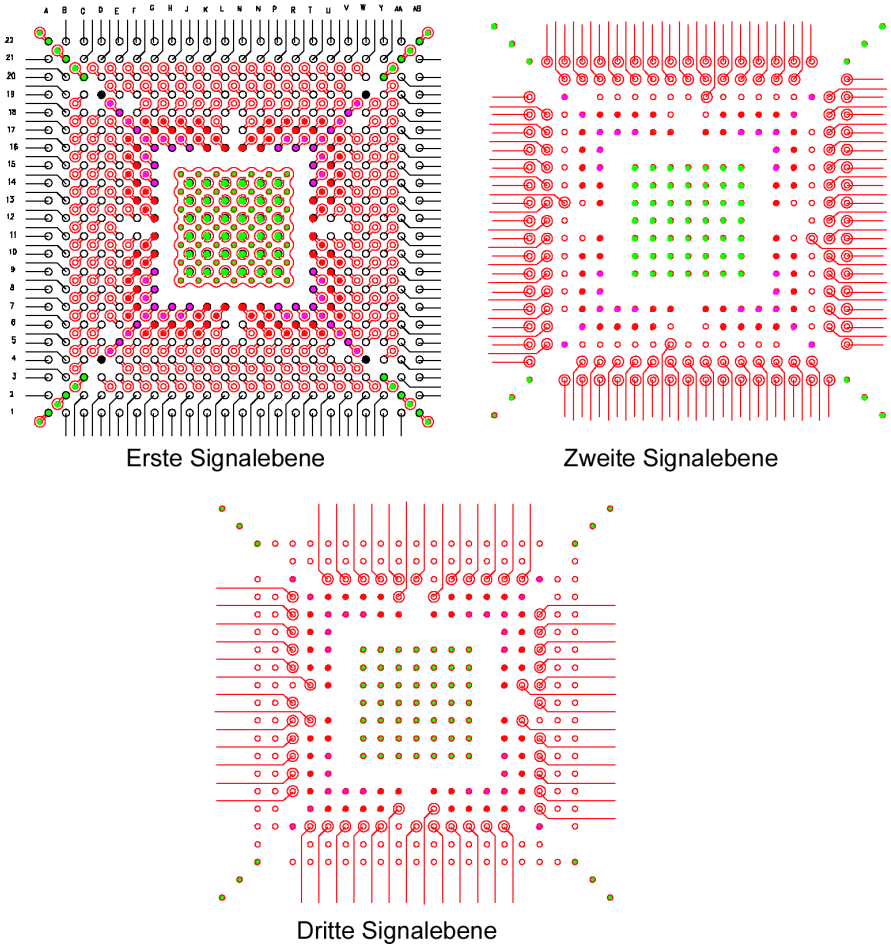


Abbildung 6.9: *FPGA Anschlussplan.* Die hohe Dichte von Anschlüssen an den FPGA im 1mm Raster des BGA-Gehäuses erfordert selbst ohne das Auskreuzen von Leitungen mindestens drei Signalebenen im PCB [129].

des PCBs dermaßen angeordnet, dass sie später durch einfaches Abschneiden eliminiert werden können. Dieses Abschneiden ist nicht nur rein mechanisch zu verstehen, sondern erfolgt auch innerhalb der CAD-Software durch löschen der Verbindungsleitungen zwischen den Steckern und den Signalen am eigentlichen inneren Kern des PCB. Damit ist es möglich, eine Endversion des PCB zu extrahieren, ohne ein wiederum fehleranfälliges Umordnen oder Umrouten am PCB vornehmen zu müssen. Die Abb. 6.10 verdeutlicht diese Idee.

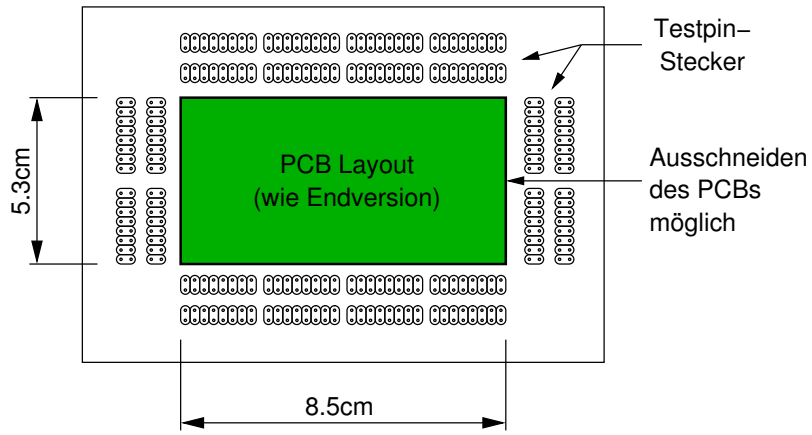


Abbildung 6.10: *Testgerechtes Layout.* Das eigentliche Layout im Zentrum der Platine wird von einer Vielzahl herausgeführter Testpins umrandet, die den Anschluss von Logikanalysatoren und Oszilloskopen ermöglichen. Nach Abschluss der Inbetriebnahme kann durch Weglassen der Testpins eine Endversion des PCB Layouts ohne weitere Abänderung gewonnen werden.

6.5 Bildmaterial

Das neue Mikroprozessormodul wurde in Form von zwei Prototypen auf jeweils zehnfach-Multilayer PCBs realisiert. Der erste Prototyp enthält gemäß Testkonzept eine Reihe von Steckern für den Anschluss von Messgeräten, was sich im Laufe der Inbetriebnahme als äußerst wertvoll herausstellte. Für den zweiten Prototypen konnte der scheckkartengroße Innenteil ohne wesentliche Änderungen übernommen werden. Anstelle der Testpins wurde eine zusätzliche Schnittstelle für den Anschluss einer Kamera sowie ein Netzwerkanschluss vorgesehen. Die folgenden Abbn. 6.11 bis 6.14 enthalten Aufnahmen der Vorder- und Rückseiten der beiden Prototypen.

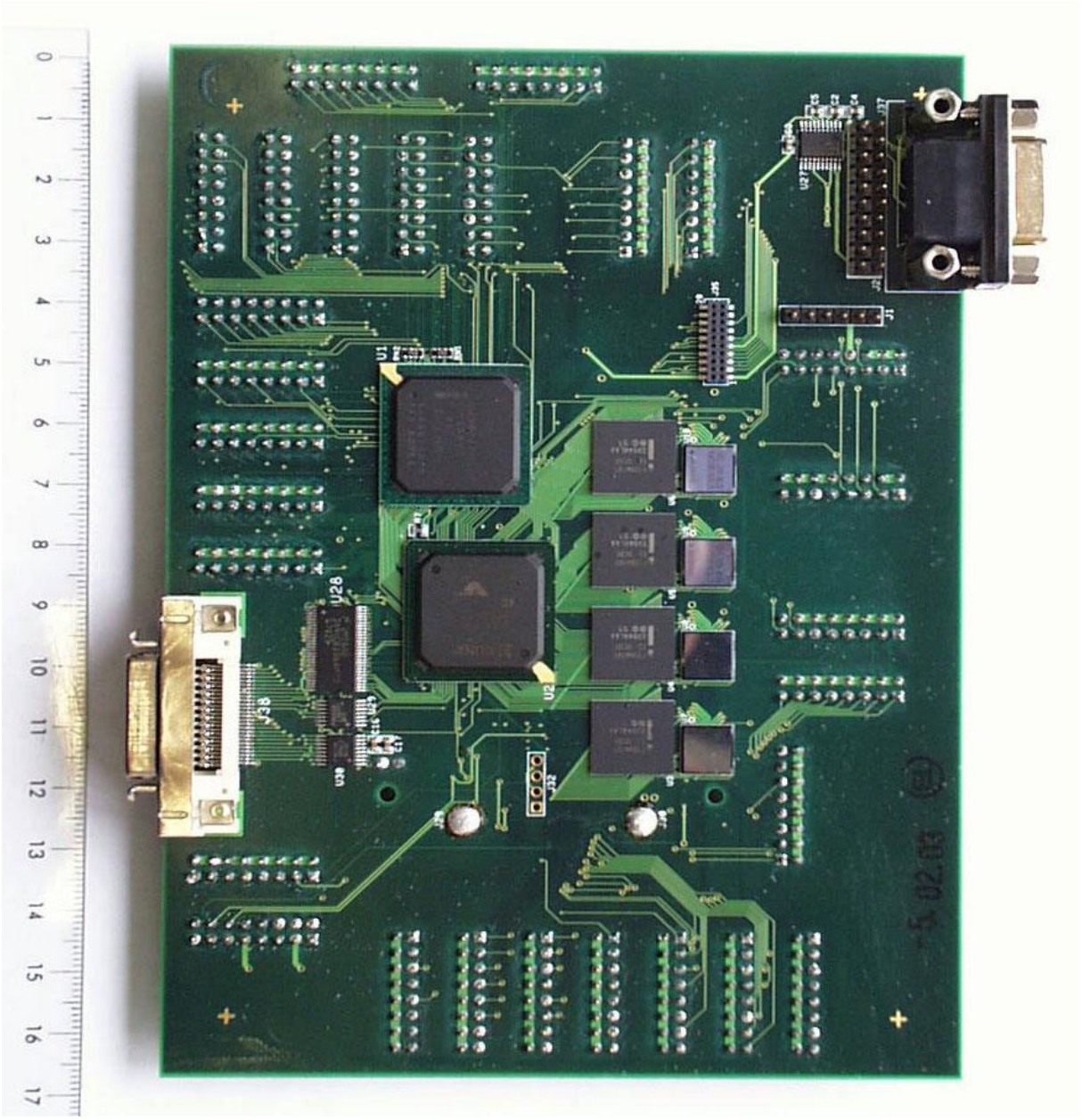


Abbildung 6.11: *Vorderseite des Mikroprozessormoduls, erster Prototyp.* Der scheckkartengroße Innenteil des Mikroprozessormoduls ist auf der Vorderseite mit dem Mikroprozessor, dem zentralen FPGA und den SDRAM- bzw. Flash-Speicherbausteinen bestückt. (Maßstab in cm)

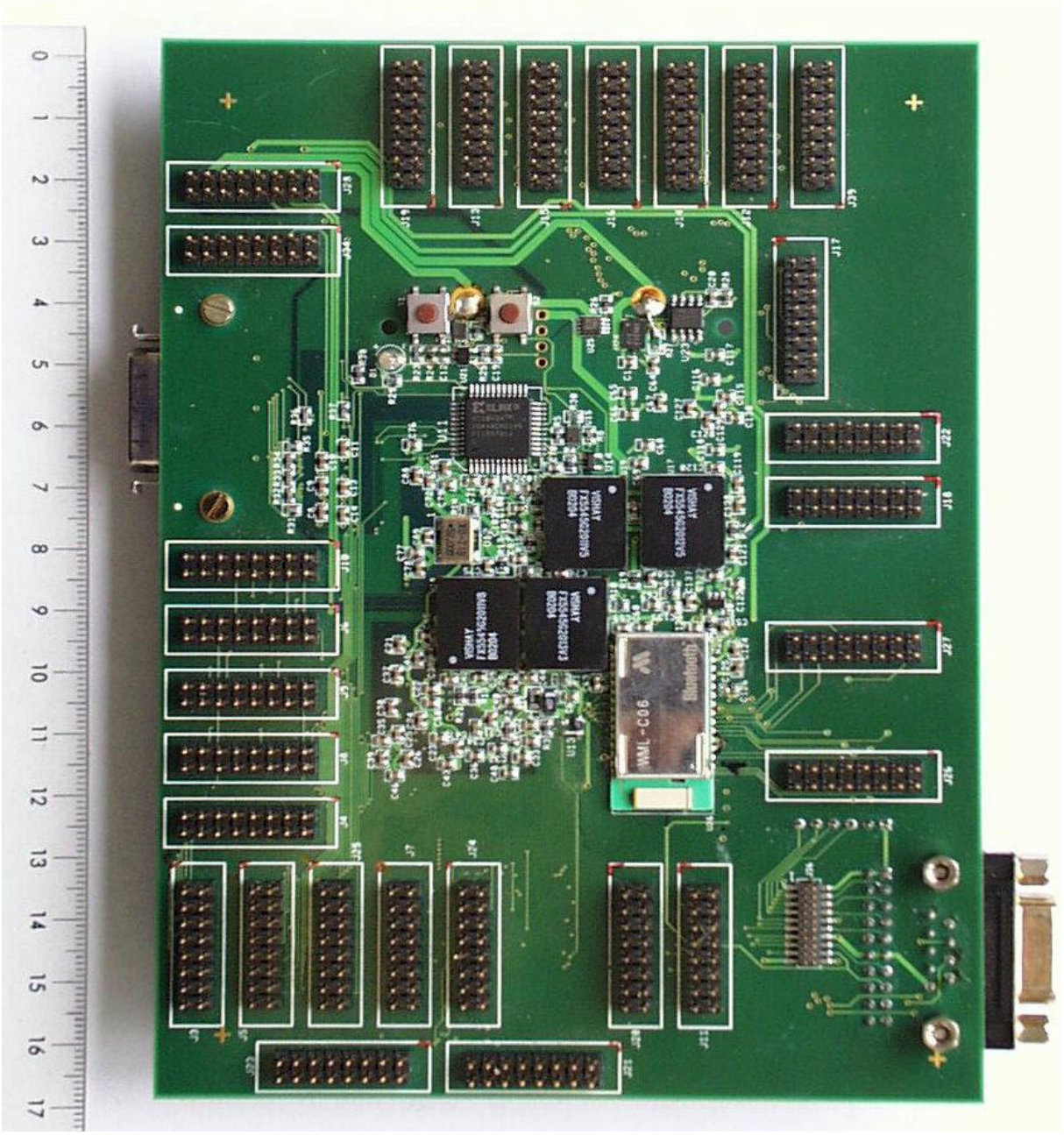


Abbildung 6.12: *Rückseite des Mikroprozessormoduls, erster Prototyp.* Auf der rückwertigen Seite des Mikroprozessormoduls befinden sich die Bausteine für die Spannungsversorgung, das Bluetooth-Funkmodul, der Konfigurationsspeicher für den FPGA sowie neben einer Reihe von Kleinbauteilen die Taster für Reset- und Ein/Ausschaltfunktion. Deutlich erkennbar sind die zahlreichen außen angeordneten Stecker für umfangreiche Mess- und Testmöglichkeiten. (Maßstab in cm)

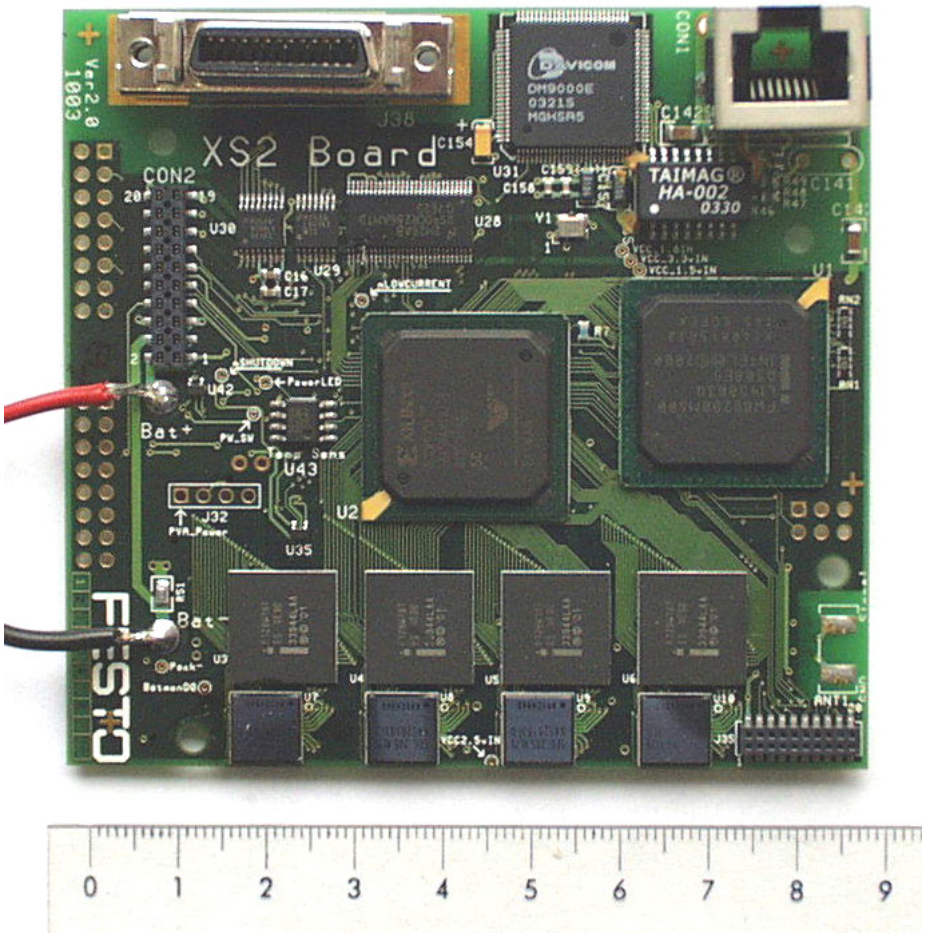


Abbildung 6.13: *Vorderseite des zweiten Prototypen.* Der scheckkartengroße Innenteil wurde vom ersten Prototypen übernommen. Als Erweiterung wurde eine zusätzliche 10/100Mbit Ethernet-Netzwerkschnittstelle (rechts oben) sowie einer Kamera-Schnittstelle (links oben) auf das Mikroprozessormodul integriert. Die zahlreichen Testpinstecker wurden am zweiten Prototypen entfernt. (Maßstab in cm)

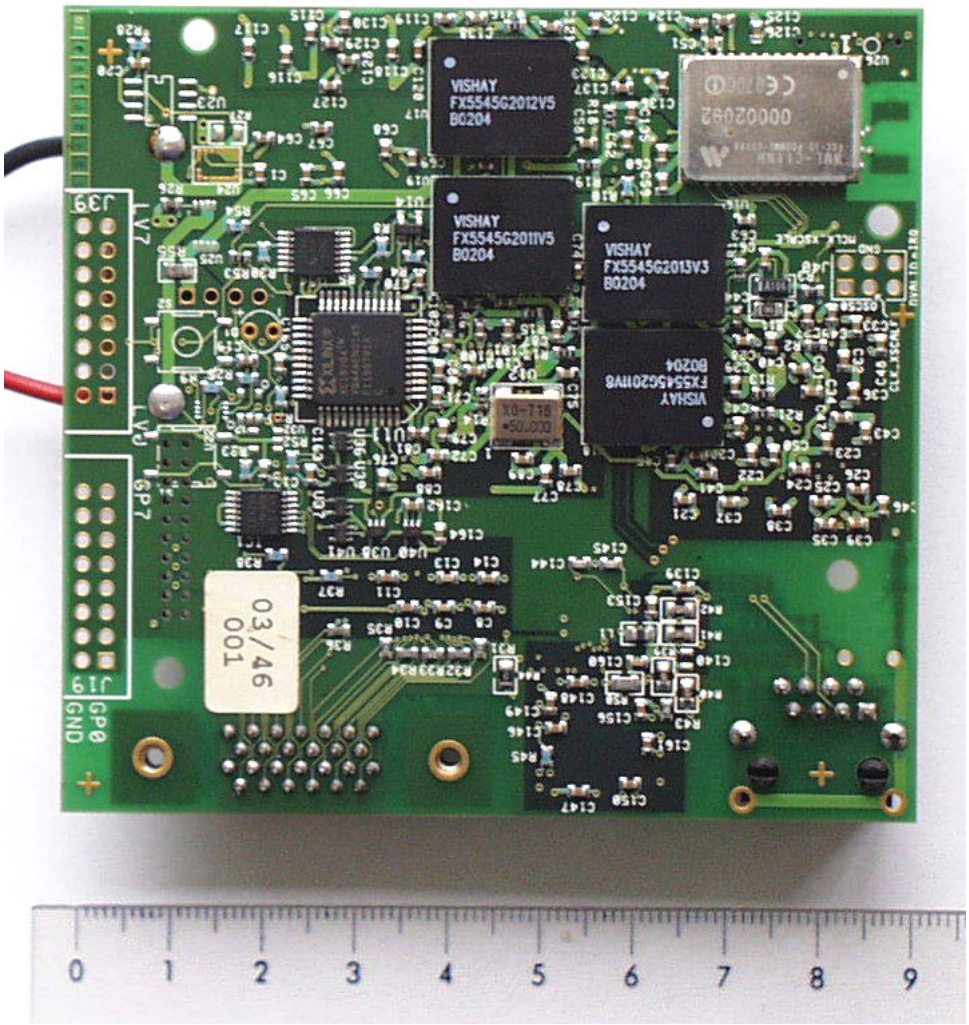


Abbildung 6.14: *Rückseite des zweiten Prototypen.* Wie beim ersten Prototypen befinden sich hier die Bausteine für die Spannungsversorgung, der Konfigurationsspeicher des FPGA sowie das Bluetooth Funkmodul. (Maßstab in cm)

Kapitel 7

Erstellung einer Entwicklungsumgebung für das neue Mikroprozessormodul

Nach dem hardwaretechnischen Entwurf ist für das neue Mikroprozessormodul eine leistungsfähige Umgebung für die Entwicklung von Software einzurichten.

Eine Entwicklungsumgebung ist grundsätzlich für jedes Mikroprozessorsystem gesondert einzurichten, da auf architekturspezifische Eigenschaften des Mikroprozessors selbst (z. B. Befehlssatz der Maschinensprache) als auch auf die Gegebenheiten des gesamten Mikroprozessorsystems (z. B. Adressraum, Speicherausstattung) Rücksicht zu nehmen ist. Im Falle dass bereits ähnlich aufgebaute Hardwarevarianten existieren, kann die Entwicklungsumgebung dieser Variante möglicherweise direkt übernommen oder leicht modifiziert werden.

Im vorliegenden Fall speziell konfektionierter Bausteine und der komplexen Anforderungen an verlustleistungsarmes Design reicht eine geringfügige Anpassung einer bestehenden Entwicklungsumgebung alleine nicht aus. Vielmehr sind vor allem das Betriebssystem samt der zugehörigen Treibersoftware speziell für die neu entwickelte Hardware zu erstellen.

Um das Portieren eines Betriebssystems überschaubar zu halten und die Hard- und Softwareentwicklung zeitgleich voranzutreiben wird ein neuer ereignisgesteuerter Simulator entwickelt, der es erlaubt, Betriebsystem und Anwendungsprogramme bereits vor Fertigstellung erster Hardwareprototypen im Simulator auszuführen und zu testen.

Insegesamt besteht die Entwicklungsumgebung aus folgenden Komponenten:

- Softwarewerkzeuge („Tools“) für die Erstellung von Programmen (z. B. Compiler, Assembler, Linker)
- Betriebssystem

- Treibersoftware (z. B. serielle Schnittstelle)
- Simulator
- Debugger

7.1 Einsatz des Betriebssystems Embedded Linux

Für den Betrieb eines Integrierten Sensorsystems erhebt sich die Frage, ob für die Bewältigung der Aufgaben und Anforderungen der Einsatz eines Betriebssystems notwendig ist. Für kleine, überschaubare Anwendungen ist ein Betriebssystem zweifellos nicht erforderlich. Für komplexere Applikationen, die über Kommunikationsschnittstellen und die dazu erforderlichen Protokolle verfügen, wo mehrere unterschiedliche Aufgaben gleichzeitig abzuarbeiten und Interaktionen mit Benutzern durchzuführen sind, erscheint ein Betriebssystem unerlässlich. Letztendlich entscheiden im Einzelfall immer die Kosten und die geplanten Stückzahlen, ob der Mehraufwand (Speicher, Overhead, Lizenzgebühren) für ein Betriebssystem in wirtschaftlich sinnvollem Verhältnis zu dessen Vorteilen (kürzere Entwicklungszeit, Stabilität, Qualität, Erweiterbarkeit) steht.

In typischen Anwendungsgebieten der Integrierten Sensorik (z. B. in der industriellen Bildverarbeitung) macht die sensorische Kernaufgabe meist nur einen Bruchteil des gesamten softwaretechnischen Aufwandes aus. Kommunikation auf unterschiedlichsten Bussystemen sowie Benutzerinterfaces stellen erfahrungsgemäß größere Aufwände dar. Dies ist auch die Begründung für den Einsatz eines Betriebssystems auf dem neuen Mikroprozessormodul.

7.1.1 Entscheidungskriterien für den Einsatz von Linux

Bei der Umsetzung verlustleistungsreduzierender Maßnahmen spielt das Betriebssystem eine wesentliche Rolle, da sämtliche Hardwarebausteine unter dessen Kontrolle liegen. Eine Eingriffsmöglichkeit in das Verhalten und die Strukturen des Betriebssystems ist daher eine Grundvoraussetzung für eine effiziente Implementierung von softwaretechnischen Low Power Methoden. Diese Eingriffsmöglichkeit bieten nur Betriebssysteme, deren Quellcode offengelegt und auch dokumentiert ist.

Der Quellcode kommerzieller Betriebssysteme ist wertvolles Know How des Herstellers und wird daher nicht oder nur durch hohes Entgelt offengelegt. Anpassungen für Low Power Maßnahmen sind allein schon aus Gewährleistungsgründen wenn überhaupt nur im Einvernehmen mit dem Hersteller möglich und sind teuer zu bezahlen.

Unbegrenzte Flexibilitäten erlauben hingegen frei erhältliche (sog. „Open Source“) Betriebssysteme, die als Quellcode vorliegen und beliebige Anpassungen ermöglichen. Die Tab. 7.1 listet eine Reihe von solchen Betriebssystemen. Für jedes Betriebssystem ist angegeben, welche Mikroprozessoren unterstützt werden.

Betriebs−system	**Unterstützte Prozessoren**	**Kurzbeschreibung**
GNU/Hurd	x86	Ein von GNU entwickelter Ersatz für Unix−Betriebssysteme
E.R.I.K.A	x86	Realtime Betriebssystem für die Automobilbranche
eCos	EP−72xx, ARM, SPARClite, 68k, SH, StrongARM, VR4300, PowerPC, TX39	Konfigurierbares, schlankes Betriebssystem von RedHat für Branchen mit hohen Stückzahlen (Consumer−Elektronik, Telekom)
Fiasco	x86	Kleiner Mikrokernel der TU Dresden
FreeDOS	x86	Freie Implementierung eines MS−DOS kompatiblen Betriebssystems
Linux	x86, 68k, IA64, Alpha, ARM, MIPS, HP PA−RISC, SH, SPARC, PowerPC	Freier Unix−Klon von Linus Torvalds mit umfangreicher Funktionalität, voll netzwerkfähig, Vielzahl von zusätzlichen Programmen vorhanden
NetBSD	Alpha, Amiga, ARM, HP300, x86, 68k, SH3, SPARC, SUN3, VAX	Von einer weltweiten Gemeinschaft entwickeltes Unix−ähnliches, freies Betriebssystem
Oberon	x86, DEC Shark	Modulares, Single−User Betriebssystem der ETH Zürich
PowerOS	PowerPC	Speziell für PowerPC entwickeltes, schlankes Betriebssystem
QNX	x86	Ursprünglich kommerzielles, skalierbares Echtzeitbetriebssystem
RTEMS	29K, SH, i386, i960, MIPS64, 68k, PowerPC, SPARC, HP PA−RISC	Realtime Betriebssystem für den Militärbereich
S.Ha.R.K	x86	Dynamisch konfigurierbarer Kernel mit einer Vielzahl an möglichen Scheduling−Prinzipien
V2_OS	i386	Sehr schnelles, in Assembler codiertes Betriebssystem

Tabelle 7.1: *Frei verfügbare „Open Source” Betriebssysteme [130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142].* In der Tabelle sind jene Betriebssysteme hervorgehoben, die Unterstützung für den Intel XScale bzw. dessen ARM-Architektur bieten.

Das im Rahmen dieser Arbeit entwickelte Mikroprozessormodul ist mit einem Intel XScale 80200 ausgestattet, welcher zur Familie der ARM-Prozessoren zählt. In der Tab. 7.1 sind daher jene Betriebssysteme hervorgehoben, die diese Prozessorfamilie unterstützen: eCos, Linux und NetBSD. Unter diesen drei Betriebssystemen geht Linux als jenes mit dem höchsten Bekanntheits- und Verbreitungsgrad hervor [143, 144]. Aufgrund seiner Vielseitigkeit und des großen Funktionsumfangs wird Linux als Betriebssystem für das neue Mikroprozessormodul ausgewählt.

Vor- und Nachteile von Linux

Die Wahl von Linux als Betriebssystem hat weitreichende Konsequenzen im Hinblick auf die Gestaltung der Entwicklungsumgebung und der Entwicklung von Software für das Mikroprozessormodul generell. Die Verwendung von Linux bietet dabei eine Reihe von Vorteilen:

- **Offene Quellcodes.** Linux ist in Form eines großen Paketes[1] an Quellcode erhältlich [145], wobei die Verbreitung hauptsächlich über das Internet erfolgt. Der offengelegte Quellcode ermöglicht beliebige Eingriffe in das Verhalten des Betriebssystems.
  Durch die Verfügbarkeit der Quellcodes ist man in der Lage, Fehler zu erkennen und auszubessern ohne die Mitwirkung des Software-Herstellers.
  Durch die weltweite Beteiligung vieler Programmierer am Linux-Projekt können rasch Informationen und Expertenmeinungen zu bestimmten Programmsequenzen eingeholt werden.
  Unter Kenntnis des Quellcodes läßt sich die Funktionweise des Betriebssystems genau nachvollziehen.
  Verbesserungen und Erweiterungen am Linux-Kernel können rasch berücksichtigt werden, ohne ein komplettes Betriebssystem zu tauschen.

- **Linux ist kostenlos.** Für das Betriebssystem fallen keinerlei Kosten oder Lizenzgebühren an. Linux unterliegt der GPL (GNU General Public License), die eine freie Distribution und Verwendung ermöglicht. Bei Verwendung kommerzieller Betriebssysteme fallen neben hohen Fixkosten für die Entwicklungsumgebung in der Regel auch Stückkosten für jede Laufzeitlizenz („Royalties“) an.

- **Viele unterstützte Architekturen.** Linux ist für viele unterschiedliche Prozessorfamilien verfügbar. Daraus ergibt sich ein hoher Verbreitungsgrad und ein hohes Synergiepotential beim Einsatz ein und desselben Betriebssystems auf unterschiedlichen Prozessoren.

- **Stabilität.** Durch den weit verbreiteten Einsatz als Serverbetriebsystem gilt Linux als ausgesprochen stabil. Diese Eigenschaft kommt besonders in sicherheitskritischen Anwendungen zum Tragen.

[1] Der Linux Kernel in der Version 2.4.13 umfasst 10233 Dateien

- **Modularer Aufbau.** Linux ist modular aufgebaut und kann durch ein hierarchisches Konfigurationskonzept den Gegebenheiten einer zugrundeliegenden Hardware optimal angepasst werden.
- **Netzwerkfähigkeit.** Schon seit der Geburtsstunde von Linux, wo die Netzwerkfähigkeit der Unix-Systeme als Vorbild diente, ist die Unterstützung von einer Reihe wichtiger Netzwerkprotokolle (allen voran TCP/IP) intrinsische Eigenschaft von Linux. Gerade im Bereich der Netzwerkprotokolle kommen die Vorteile eines Betriebsystems voll zum Tragen, da Protokollstacks Mechanismen zum Multitasking, zur Synchronisation und Timersteuerung erfordern [146].
- **Echtzeitfähigkeit.** Linux ist auf möglichst hohe Verarbeitungsgeschwindigkeit (Durchsatz) optimiert und daher nicht von vorne herein echtzeitfähig. Es existieren jedoch mit RT-Linux bzw. RTAI Erweiterungen, die den Linux-Kernel echtzeitfähig machen und damit garantierte Antwortzeiten des Betriebssystems sicherstellen.

Durch den Einsatz von Linux ergeben sich aber auch zwangsläufig einige Nachteile, die bei einer Entscheidung für dieses Betriebssystem beachtet werden müssen:

- **Mehraufwand bei Installation.** Linux ist aufgrund seiner allgemeinen Verwendbarkeit für den jeweiligen Anwendungsfall anzupassen und einzurichten. Dies bedeutet einen Mehraufwand zu Beginn einer Entwicklungstätigkeit. Kommerzielle Betriebssysteme werden bereits angepasst geliefert und sind nach ihrer Installation sofort voll einsatzfähig.
  Manche Softwarehersteller haben hier eine Marktlücke erkannt und vertreiben für verschiedene Anwendungsfälle spezielle Linux-Distributionen, bei denen der Anpassungsaufwand bereits vorgeleistet ist (z. B. Monta Vista Linux [147], Embedix [148], BlueCat Linux [149]).
- **Lizenzmodell - kein Schutz von proprietärem IP.** Linux unterliegt einem Lizenzmodell welches vorsieht, dass Änderungen, die an Linux vorgenommen werden wiederum vollkommen frei zugänglich sein müssen. Ein Schutz von proprietärem Know How ist in einem solchen Fall nicht möglich. Der Abschnitt 7.1.4 geht näher auf das Lizenzmodell von Linux ein.
- **Kein dezitierter Ansprechpartner.** Nachdem die Entwicklung von Linux durch eine weltweite Gemeinschaft von Programmierern vorangetrieben wird, gibt es zwar für einzelne Softwareteile zuständige Experten, einen dezitierten Ansprechpartner oder Gewährleistungsträger gibt es aber nicht. Der Anwender von Linux trägt dadurch Mitverantwortung über das Betriebssystem.

7.1.2 Embedded Linux

Im Sprachgebrauch der facheinschlägigen Literatur findet sich sehr häufig der Begriff „Embedded Linux“. Darunter sind Varianten von Linux zu verstehen, deren Fokus nicht auf Arbeitsplatzrechner und Serversystemen gerichtet ist, sondern die für kleine, ressourcenbegrenzte Hardwareplattformen, wie sie im Bereich der Embedded Systeme vorkommen, ausgelegt sind. Diese Spezialisierung wird einerseits durch minimale Konfiguration des Linux-Kernels und andererseits durch gezieltes Weglassen nicht benötigter Teilkomponenten erreicht.

Die Minimalanforderungen an einen Arbeitsplatzrechner, der mit einer neuen Version von Linux ausgestattet werden soll, belaufen sich zur Zeit auf einen Pentium-kompatiblen PC mit 200MHz Taktfrequenz, 64Mbyte Hauptspeicher und 650Mbyte Festplattenspeicher [150]. Eine Embedded Linux Variante, wie z. B. Monta Vista Linux, findet im Gegensatz dazu im Minimalfall mit 500kbyte Speicher das Auslangen [147].

Eine weitere Eigenschaft von Embedded Linux ist, dass das Betriebssystem direkt aus einem ROM-Speicher ausgeführt werden kann, ohne es zuvor in den RAM-Speicher zu laden. Man bezeichnet diese Eigenschaft als XIP-fähig (Execute In Place).

7.1.3 Der Aufbau von Linux

Für den Einsatz von Linux und für die Entwicklung von Applikationen sind genaue Kenntnisse über den Aufbau dieses Betriebssystems erforderlich. Linux ist schichtweise organisiert (Abb. 7.1). Sämtliche Zugriffe auf die Hardware können ausschließlich vom Kernel und den darin enthaltenen Gerätetreibern durchgeführt werden. Nach außen stellt der Kernel eine Schnittstelle in Form von Systemaufrufen zur Verfügung. Assemblerprogramme können diese Schnittstelle direkt nutzen. Programme, die in Hochsprache, vorzugsweise in C/C++ abgefasst sind, nutzen die Applikationsschnittstelle der C-Bibliothek, welche umfangreiche Funktionen zur Verfügung stellt, die ihrerseits auf Systemaufrufe des Kernels abgebildet werden.

Neben den Applikationen sind damit die C-Bibliothek und der Kernel die wesentlichen Komponenten eines Linux-Systems. Das bedeutet für das neue Mikroprozessormodul, dass am Kernel einerseits Anpassungen an die Hardware vorzunehmen sind und andererseits wegen der Schnittstelle zur Hardware die Low Power Funktionalität (z. B. Ansteuerung von energiesparenden Zuständen) direkt in den Kernel eingearbeitet werden muss.

Aber auch die C-Bibliothek ist von großer Bedeutung, da diese im Regelfall erheblich mehr Speicherplatz in Anspruch nimmt als der Kernel. Die am neuen Mikroprozessormodul eingesetzte GNU C-Bibliothek[2] [151] benötigt beispielsweise 1.55Mbyte Speicher, während der Kernel dagegen nur 846kbyte belegt.

[2]Die verwendete Version 2.2.3 der GNU C-Bibliothek besteht aus 8944 Dateien

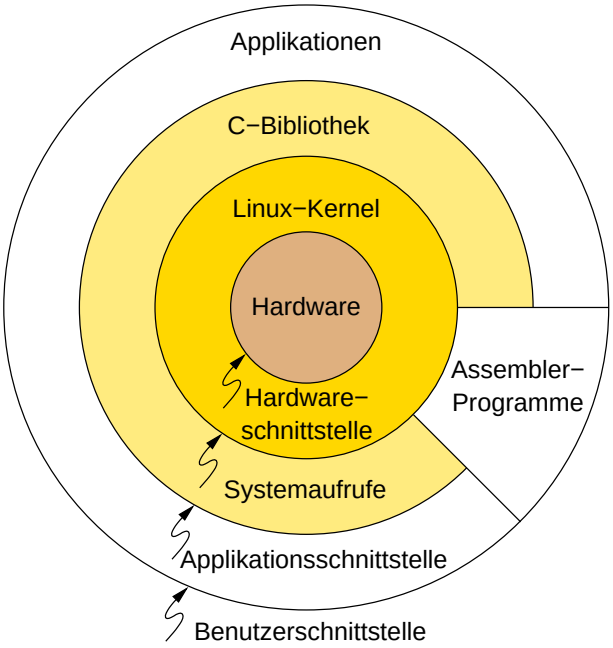


Abbildung 7.1: *Der Schichtenaufbau eines Linux-Systems mit Schnittstellen.* Die komplette Hardware unterliegt der Kontrolle durch den Linux-Kernel. Zwischen Kernel und Applikationen stellt die C-Bibliothek wichtige Funktionen und Schnittstellen zur Verfügung. Assemblerprogramme können den Linux-Kernel auch direkt über Systemaufrufe ansprechen.

Alternativ zur GNU C-Bibliothek sind in den letzten Jahren auch Alternativen entstanden, die sich aufgrund ihres ressourcenschonenden Aufbaus besser für den Einsatz in Embedded Systemen eignen, z. B. ucLibc [152], Newlib [153] oder dietlibc [154]. In [143] wird ein Größenvergleich dieser C-Bibliotheken vorgenommen, der in Tab. 7.2 wiedergegeben ist.

C-Bibliothek	Dateigröße
GNU Libc	1.3 Mbyte
ucLibc	339 kbyte
Newlib	269 kbyte
dietlibc	200 kbyte

Tabelle 7.2: *Größenvergleich von C-Bibliotheken.* [143] Der Vergleich wurde auf einer i386-Plattform durchgeführt und zeigt die jeweilige Dateigröße der Bibliotheken ohne Symbolinformation.

7.1.4 Das Lizenzmodell von Linux

Linux unterliegt der wichtigsten Lizenz für freie Software, der GPL (GNU General Public License, [155]). Diese räumt dem Benutzer einer GPL-Software das Recht ein, die Software frei weiterzuverteilen, auf Anfrage den Quellcode zu erhalten und den Quellcode nach seinen Wünschen und Vorstellungen abzuändern. Werden Änderungen an einer GPL-Software vorgenommen, so sieht die Lizenz vor, dass diese Änderungen wiederum der GPL unterliegen. Auf diese Weise wird verhindert, dass in kostenpflichtiger, proprietärer Software Teile von freier Software eingearbeitet werden können.

In vielen Fällen ist es im Sinne von Geheimhaltung oder Sicherheit erforderlich bzw. aus wirtschaftlichen Gründen notwendig, Software gegen Veräußerung des Quellcodes zu schützen. Um die Verbreitung freier Software nicht durch diese Aspekte zu hemmen, wurde die LGPL (GNU Lesser General Public License, [156]) entwickelt. Im Prinzip gelten für die LGPL dieselben Grundsätze wie für die GPL mit der Ausnahme, dass Software bei bloßer Verwendung einer unter der LGPL stehenden Programmbibliothek nicht veröffentlicht werden muss. Damit ist es möglich, LGPL-Software in proprietäre Projekte einzubinden. Änderungen, die an der Bibliothek selbst vorgenommen werden, unterliegen jedoch wiederum der LGPL.

7.2 GNU Entwicklungswerkzeuge

Der Einsatz von Linux als Betriebssystem für das neue Mikroprozessormodul bedingt automatisch die Verwendung von GNU Entwicklungswerkzeugen, da diese z. B. für das Übersetzten des Linux-Kernels Voraussetzung sind. Zu diesen Entwicklungswerkzeugen (auch als Tools bezeichnet) zählen neben einer Reihe von kleineren Programmen folgende große Pakete:

- Compiler
- Assembler
- Linker
- Archiver
- Debugger

Diese Entwicklungswerkzeuge sind Voraussetzung für die Erstellung von ausführbaren Programmen (Abb. 7.2). Abhängig davon, ob diese Werkzeuge auf einem Entwicklungs-PC oder direkt auf der Zielhardware eingesetzt werden, spricht

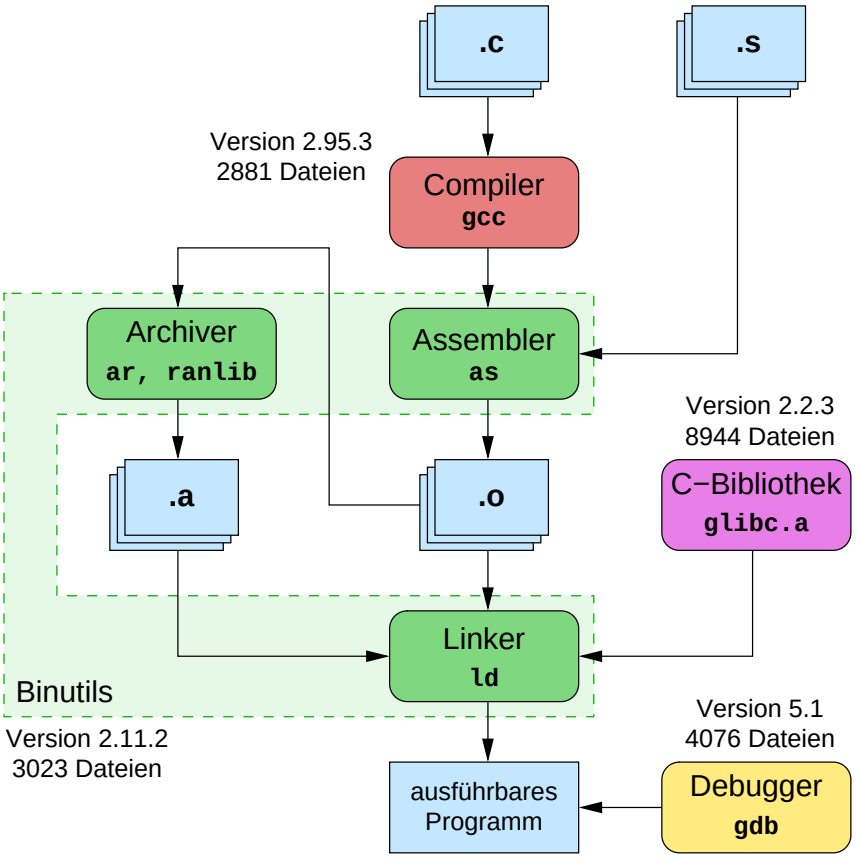


Abbildung 7.2: *GNU Entwicklungswerkzeuge für das neue Mikroprozessormodul mit Anzahl der Quelldateien.* Die Entwicklung ausführbarer Programme beginnt mit der Übersetzung von Quelldateien in C (Dateiendung .c) oder Assembler (Dateiendung .s) durch den Compiler und Assembler zu Objektdateien (Dateiendung .o). Gruppen von Objektdateien lassen sich mittels Archiver zu Bibliotheken (Dateiendung .a) zusammenfassen. Der Linker erstellt schließlich das ausführbare Programm, das im Debugger getestet werden kann. Für die jeweiligen Entwicklungswerkzeuge sind die verwendete Version sowie die Anzahl von Quelldateien angegeben, aus denen sie bestehen.

man von einer „Cross“- bzw. „Native“-Entwicklung. Üblicherweise wird Cross entwickelt, da die Ressourcen der Zielhardware meist für die Ausführung eines Compilers nicht ausreichen.

GNU Entwicklungswerkzeuge haben den Vorteil, dass sie ebenso wie Linux auf vielen verschiedenen Plattformen eingesetzt werden und dadurch sehr stabil und ausgereift sind. Sie sind kostenlos im Rahmen der GPL-Lizenz erhältlich [157, 158] und weit verbreitet.

Als Nachteil muss der beträchtliche Aufwand für das Übersetzen der GNU Entwicklungswerkzeuge angesehen werden[3]. Dieser Thematik ist die Diplomarbeit von Robert Kroiß [159] gewidmet. Um die Komplexität des Übersetzungsvorganges zu veranschaulichen, ist in der Abb. 7.2 die Anzahl von Dateien angegeben, aus denen die Entwicklungswerkzeuge bestehen.

7.3 Entwicklung eines Hardware-Simulators für das neue Mikroprozessormodul

Unter dem Druck des Marktes mit immer kürzer werdenden Produktentwicklungszyklen ist man gezwungen, die Entwicklungstätigkeiten für Hard- und Software weitgehend zu parallelisieren. Hauptproblem dabei ist, dass wichtige Komponenten der Basissoftware (Betriebssystem und Gerätetreiber) erst dann getestet werden können, wenn erste funktionstaugliche Prototypen der zugrundeliegenden Hardware zur Verfügung stehen. Dieses Problem kann wesentlich entschärft werden, indem das Verhalten der realen Hardware in einem Simulationsmodell nachgebildet und die Basissoftware gemeinsam mit diesem Modell in einem Simulator getestet wird. Für die Entwicklung des neuen Mikroprozessormoduls wird diese Vorgangsweise aufgegriffen, um das Betriebssystem Linux schon frühzeitig auf das parallel dazu entwickelte Mikroprozessormodul zu portieren.

7.3.1 Eventgesteuerte Simulation mit SID

Für den Aufbau eines Simulationsmodells für das vorliegene Mikroprozessormodul existiert mit dem Open-Source Programmpaket „SID“ [160] bereits ein gut gelungener Ansatz, der im Rahmen dieser Arbeit weiterverfolgt wird.

SID stellt ein Gerüst zur Verfügung, mit dem es möglich ist, aus einer Reihe von einzelnen Komponenten durch Zusammenschalten eine komplette Eventgesteuerte Simulation aufzubauen. Komponenten können in C oder C++ programmiert [161, 162] werden und besitzen nach außen einfache Schnittstellen in Form

[3] Es sind zwar binäre, bereits übersetzte Distributionen erhältlich, die allerdings völlig inflexibel bezüglich Versionsänderungen sind.

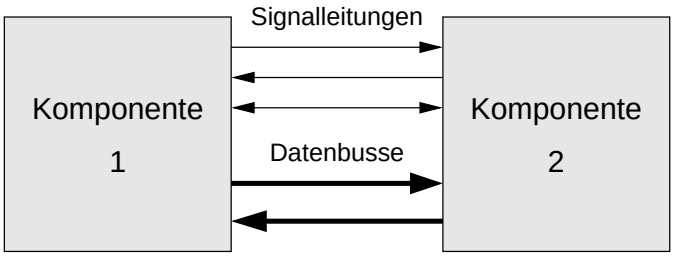


Abbildung 7.3: *Komponentenorientierte Simulation mit SID.* Einzelne Komponenten werden in C/C++ programmiert und kommunizieren durch Signale und Busse miteinander. Signale können uni- oder bidirektional, Busse ausschließlich unidirektional sein.

von Signalleitungen und Datenbussen (Abb. 7.3). Eine Konfigurationsdatei beschreibt die Art und Weise des Zusammenschaltens und stellt Parameter innerhalb der Komponenten ein. Nach dem Start von SID werden sämtliche Komponenten der Reihe nach initialisiert und die Simulation durch Einschleusen eines Reset-Events gestartet. Die Komponenten reagieren in der Folge auf ihre Eingangssignale und generieren daraus die entsprechenden Ausgangssignale an ihren Schnittstellen.

7.3.2 Erstellung eines neuen XScale Simulationsmoduls

SID beinhaltet eine Reihe bereits vorgefertigter Komponenten, wie z. B. Prozessormodule, Speicher und div. Ein-/Ausgabebausteine. Auch die Anbindung an einen GNU-Debugger ist bereits vorgeleistet, was für den Softwaretest von besonderer Bedeutung ist.

Für die Simulation des neuen Mikroprozessormoduls müssen Simulationsmodelle für die folgenden Komponenten neu entworfen werden:

- XScale-Prozessor
- Serielle Schnittstelle UART
- Interruptcontroller

Der XScale-Prozessor basiert auf einer ARM-Architektur und realisiert zusätzlich zur Version ARMv5TE [163] des ARM-Maschinenbefehlssatzes eine Reihe weiterer Befehle für digitale Signalverarbeitungsalgorithmen. Das bedeutet, dass die in SID enthaltene Simulationskomponente für ARM-CPUs zwar verwendet werden kann, allerdings um sämtliche ARMv5TE- und XScale-spezifische Eigenschaften erweitert werden muss. Dazu zählen in erster Linie die vom XScale gegenüber ARM zusätzlich implementierten Maschinenbefehle.

Eine weitere Herausforderung besteht darin, die im XScale enthaltenen Koprozessoren in das Simulationsmodell einzubinden. Dazu gehören z. B. der gesamte Cache-Controller und die komplexe Memory Management Unit (MMU), die für eine Simulation von Linux-Software unbedingt Voraussetzung ist [164].

Ausgehend von der Komponente für ARM-CPUs wurde im Rahmen der vorliegenden Arbeit eine neue, vollständige Simulationskomponente für den XScale-Prozessor erstellt.

Die weiteren Komponenten für den UART der seriellen Schnittstelle sowie der Interruptcontroller weisen im Vergleich mit dem XScale geringere Komplexität auf und wurden ebenfalls in Form einer SID-Komponente für die vollständige Simulation des neuen Mikroprozessormoduls entwickelt.

Die Abb. 7.4 zeigt das Blockschaltbild einer kompletten Simulationsumgebung für das neue Mikroprozessormodul. Die Simulation besteht aus den Komponenten für den XScale-Prozessor, den UART und Interruptcontroller sowie der Komponenten für RAM und ROM. Ein Adressmapper sorgt für die richtige Zuordnung der Prozessorzugriffe auf die verschiedenen Speicherbereiche des Adressraums. Ein Debuggerinterface gestattet den Zugriff bzw. die Steuerung der Simulation durch die Entwicklungsumgebung. Sämtliche Ein- und Ausgaben erfolgen über die serielle UART Schnittstelle, die auf ein Terminalfenster des Entwicklungs-PCs umgeleitet wird.

Das in Abb. 7.4 dargestellte Modell stellt somit einen vollständigen Simulator für das neue Mikroprozessormodul dar. Dadurch ist es möglich, die softwaretechnische Entwicklung bzw. Portierung des Linux-Kernels bereits in einer frühen Designphase voranzutreiben, ohne Vorliegen einer tatsächlichen Hardware.

Neben den verbesserten Parallelisierungsmöglichkeiten von Hard- und Softwareentwicklung bietet ein Simulator noch weitere Vorteile:

- Beliebige Duplizierbarkeit und damit beliebig viele Testplätze
- Genaue Analyse jeglicher Abläufe im Zusammenspiel von Hard- und Software möglich, da normalerweise nicht zugängliche Informationen zur Verfügung stehen (z. B. Auslastung von Caches oder Prozessorpipelines)
- Einfacher Test von zeitkritischen Vorgängen, da die Zeit im Simulator beliebig anhaltbar ist
- Da durch den Simulator die Hardware beliebig steuerbar ist, können selten auftretende Fehlerfälle einfach getestet werden

Im Zusammenhang mit Simulatoren ist die Simulationsgeschwindigkeit stets von Interesse. Im vorliegenden Fall der Simulation eines Mikroprozessormoduls mit einem 600MHz-Prozessor konnte bei der Ausführung von Programmen ein Geschwinigkeitsunterschied von etwa 1:400 zwischen echter Hardware (600MHz XScale) und Simulation (Linux-PC mit 1.2GHz Athlon-Prozessor und 512Mbyte Speicher) gemessen werden, d. h. die Simulation lief um den Faktor 400 langsamer.

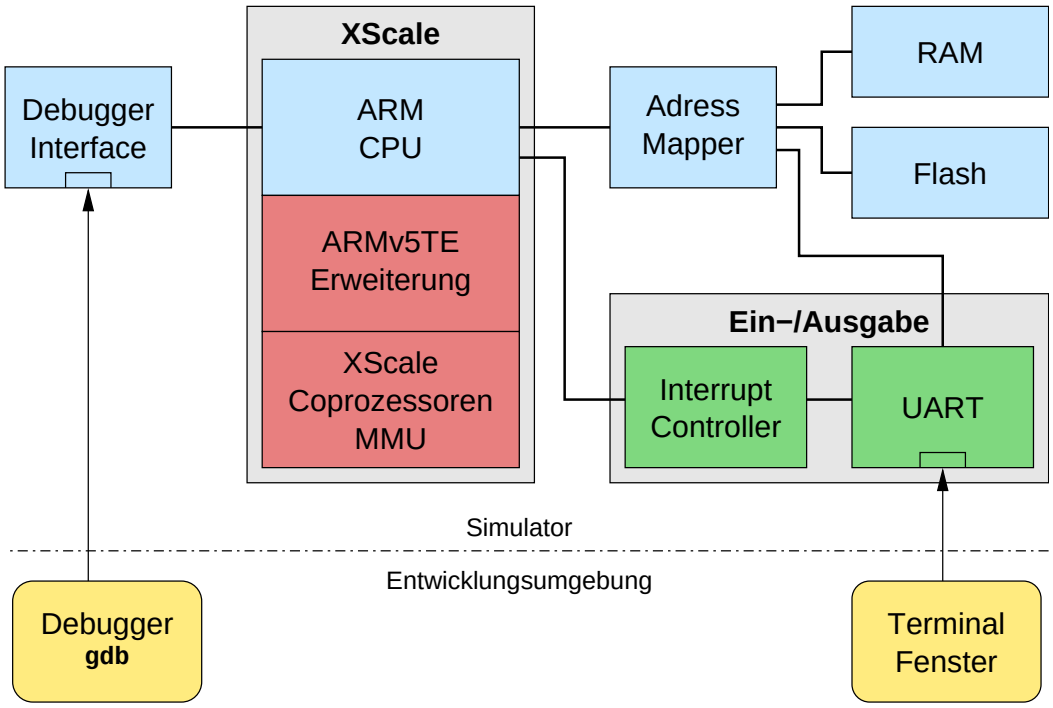


Abbildung 7.4: *Komponentenorientierter Aufbau des XScale-Simulators für das neue Mikroprozessormodul.* Die Simulationskomponente für den XScale-Prozessor wurde auf Basis der vorhandenen Komponente für ARM-CPUs neu entwickelt. Die Komponenten UART und Interruptcontroller simulieren die Ein-/Ausgabeschnittstelle des Mikroprozessormoduls. Der Speicher wird durch die Komponenten RAM und Flash nachgebildet. Der Simulator ermöglicht über Schnittstellen den Einsatz eines Debuggers sowie ein Terminalfenster für die Ein- und Ausgaben.

7.4 Portierung des Linux-Kernels auf das neue Mikroprozessormodul

Für die Inbetriebnahme von Linux auf dem neuen Mirkoprozessormodul ist eine Portierung des Linux-Kernels auf dieses Hardware erforderlich. Wie aus Abb. 7.1 ersichtlich, greifen ausschließlich die innersten Schichten des Kernels direkt auf die Hardware zu und sind somit Gegenstand der Portierungsarbeit.

Linux ist bereits auf viele Plattformen portiert worden, sodass bereits die grundsätzliche Unterstützung für ARM-Prozessoren in den ARMLinux-Kernel (eine Variante des Linux-Kernels) eingearbeitet wurde [165]. Da der XScale eine ARM-Architektur aufweist, ist damit bereits ein großer Teil der Portierungsarbeit geleistet, wenngleich auch damit keine speziellen XScale-spezifischen Eigenschaften ausgenutzt werden.

Der Portierungsaufwand liegt in der Anpassung der hardwarespezifischen Anteile des ARMLinux-Kernels an die Gegebenheiten der vorliegenden Hardware sowie der Entwicklung von Treibern [166] für die darauf befindlichen Ein-/Ausgabebausteine.

Gerhard Khüny hat in seiner Diplomarbeit [167] Linux auf das neue Mikroprozessormodul portiert und seine Arbeit im Simulator getestet. Das Ergebnis (Abb. 7.5) war ein erfolgreicher Bootvorgang und ein lauffähiges Linux-Betriebssystem rein auf Basis eines Hardwaresimulators.

```
Uncompressing Linux..... done, booting the kernel.
Linux version 2.4.13-ac5-rmk2-pva.1 (gery@hal) (gcc version 2.95.2
19991024 (release)) #274 Fri Jun 14 11:34:53 CEST 2002
Processor: Intel XScale-80200 revision 2
Architecture: PVA board
On node 0 totalpages: 16384
zone(0): 16384 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: root=/dev/mtdblock1 console=ttyPV1
init=/bin/init load_ramdisk=1 ramdisk_start=0 prompt_ramdisk=0
ramdisk_size=16384
Using XScale PMU as timer source
PVA uart console ttyPV1 registered (baud=38400 parity=n)
Calibrating delay loop... 599.65 BogoMIPS
Memory: 64MB = 64MB total
Memory: 63464KB available (717K code, 192K data, 36K init)
Dentry-cache hash table entries: 8192 (order: 4, 65536 bytes)
Inode-cache hash table entries: 4096 (order: 3, 32768 bytes)
Mount-cache hash table entries: 1024 (order: 1, 8192 bytes)
Buffer-cache hash table entries: 4096 (order: 2, 16384 bytes)
Page-cache hash table entries: 16384 (order: 4, 65536 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Starting kswapd v1.8
JFFS2 version 2.1. (C) 2001 Red Hat, Inc., designed by Axis Communications AB.
block: queued sectors max/low 42106kB/14035kB, 128 slots per queue
RAMDISK driver initialized: 16 RAM disks of 16384K size 1024 blocksize
Creating 4 MTD partitions on "PVA flash":
0x00000000-0x00080000 : "Kernel (zImage)"
0x00080000-0x01080000 : "Root Filesystem"
0x01080000-0x01280000 : "User Filesystem"
0x01280000-0x04000000 : "User Data"
NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 16384 blocks [1 disk] into ram disk... done.
VFS: Mounted root (ext2 filesystem) readonly.
Freeing init memory: 36K
serial console detected. Disabling virtual terminals.
init started: BusyBox v0.60.3 (2002.06.02-19:44+0000) multi-call binary

Please press Enter to activate this console.

BusyBox v0.60.3 (2002.06.06-09:48+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

# cat /proc/cpuinfo
Processor : Intel XScale-80200 rev 2 (v5l)
BogoMIPS : 599.65
Features : swp half thumb fastmult edsp

Hardware : PVA board
Revision : 0000
Serial : 0000000000000000
#
```

Abbildung 7.5: *Booten von Linux im Simulator*. [167]. Der Bootvorgang beginnt mit dem Dekomprimieren des Kernels in den RAM. Anschließend erfolgt die Initialisierung des Kernels. Durch den Start des Init-Prozesses (hier die BusyBox-Shell) ist der Bootvorgang abgeschlossen, sodass Kommandos vom Benutzer eingegeben werden können (z. B. `cat /proc/cpuinfo`).

Kapitel 8

Weitere Verbesserung von Performance und Verlustleistung durch ein neues Verfahren zur energieoptimierten Cache-Zuteilung

In den Kapiteln 6 und 7 wurde die Realisierung eines neuen Mikroprozessormoduls und zugehöriger Entwicklungsumgebung nach der Methodik des Kapitels 5 dargestellt. In diesem Abschnitt wird die Optimierung von Performance und Verlustleistung konsequent weiterverfolgt, indem ein neues Verfahren vorgestellt wird, das eine Steigerung der Rechengeschwindigkeit bei gleichzeitiger Verringerung des Leistungsverbrauches ermöglicht. Grundkonzept dabei ist die energetisch günstigere Nutzung vorhandener Speicher-Caches durch Umordnen des ausgeführten Programmcodes. Eine wesentliche Eigenschaft des neuen Verfahrens ist, dass es weder die Manipulation von Quellcodes noch die Modifikation von Hardware erforderlich macht.

8.1 Gleichzeitige Verbesserung von Performance und Verlustleistung durch Cache-Speicher

Die Gestaltung der Speicherhierarchie, bestehend aus Prozessor, Cache-Speicher und Hauptspeicher, spielt seit Jahrzehnten eine große Rolle bei der Steigerung der Performance eines Mikroprozessorsystems. Heutige PC-Systeme besitzen sogar mehrere hintereinandergeschaltete Cache-Speicher (man spricht von Level-1 bis Level-3 Caches) um die mittlere Zugriffsgeschwindigkeit auf Daten, die im Hauptspeicher liegen, zu erhöhen.

Die Speicherhierarchie trägt wesentlich zu einem verringerten Leistungsverbrauch eines Mikroprozessorsystems bei, da die benötigte Energiemenge E_c für den Zugriff auf einen Cache-Speicher wesentlich geringer ist als die Energiemenge E_s für einen Zugriff auf den Hauptspeicher (z. B. SDRAM). Dies ist dadurch zu begründen, dass Cache-Speicher in der Regel direkt in den Mikroprozessor integriert sind (kurze Signalwege), aufgrund ihrer geringeren Speicherkapazität energetisch effizienter aufgebaut werden können [168] und in ihrer Ansteuerung wesentlich weniger komplex sind als SDRAM (geringere Gatteraktivität). Der Zugriff auf SDRAM-Speicher erfolgt stets über einen SDRAM-Controller, der für die richtige Ansteuerung und den zyklischen Refresh der SDRAM-Bausteine sorgt, wodurch zusätzliche Energie benötigt wird.

Wenn es gelingt, durch bessere Ausnutzung des vorhandenen Caches die Anzahl der Hauptspeicherzugriffe zu verringern, kann der Energiebedarf eines Programmes signifikant reduziert werden. Darüber hinaus bedeutet eine effizientere Nutzung des Caches eine gleichzeitige Steigerung der Performance bei verringertem Energiebedarf. Jegliche Methode, die eine Steigerung in der Cache-Effizienz herbeiführt wirkt damit unmittelbar dem Kompromiss zwischen Performance und Verlustleistung entgegen und ist ein wertvolles Instrument zur Realisierung von hochperformanten Low Power Mikroprozessorsystemen.

Die Untersuchungen im Rahmen dieses Kapitels zur energieoptimierten Cache-Zuteilung erfolgen anhand der Speicherhierarchie des neu entwickelten XScale-Mikroprozessormoduls, weshalb auf diese im folgenden Abschnitt eingegangen wird.

8.2 Cache und Speicherhierarchie des neuen Mikroprozessormoduls

Das XScale-Mikroprozessormodul verfügt über einen 32kbyte großen Befehlscache („Instruction“ Cache oder „I“-Cache) und einen 32kbyte großen Datencache („D“-Cache), die direkt in den XScale-Mikroprozessor integriert sind. Die zeitlich und energetisch aufwendigen Zugriffe auf den Hauptspeicher erfolgen über den SDRAM-Controller (VHDL-Implementierung im FPGA), der für die richtige Ansteuerung und den Refresh der SDRAM-Bausteine sorgt (Abb. 8.1).

I-Cache und D-Cache besitzen denselben strukturellen Aufbau. Jeder dieser Caches verfügt über 32 Sets bei 32-facher, voller Assoziativität. Bei einer Cache-Größe von 32kbyte sind die im Cache-Speicher abgelegten Datenpakete (Cache-Zeilen) 32byte bzw. 8word lang (Abb. 8.2). Die 32 assoziativen Einträge innerhalb eines Sets unterliegen einer Round-Robin Verdrängungstrategie, d. h. ein neuer Cache-Eintrag überschreibt den jeweils ältesten. Die zugrundeliegende Verdrängungsstrategie ist eine wesentliche Randbedingung bei der Entwicklung von Cacheoptimierungsverfahren.

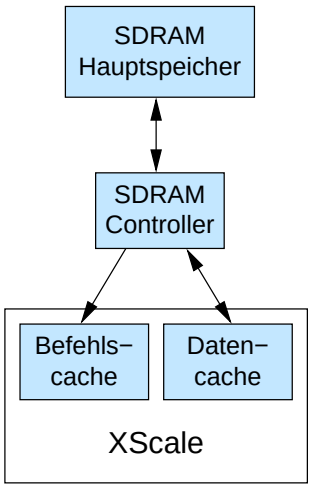


Abbildung 8.1: *Speicherhierarchie des neuen Mikroprozessormoduls.* Der Intel XScale besitzt je einen 32kbyte großen Cache für Befehle und Daten. Informationen, die nicht im Cache enthalten sind, müssen über den SDRAM-Controller aus dem Hauptspeicher nachgeladen werden.

Die Adressierung einzelner Speicherworte erfolgt anhand unterschiedlicher Portionen der Speicheradresse. Die Bits fünf bis neuen selektieren einen der 32 möglichen Sets des Cache-Speichers. Die Sets sind als inhaltsadressierbarer, assoziativer Speicher aufgebaut, wobei die Adressierung durch die Tag-Portion (Bit zehn bis 31) der Speicheradresse vorgenommen wird. Aus einer kompletten Cache-Zeile werden schließlich durch die Adressbits zwei bis fünf einzelne Datenwörter durch einen Multiplexer ausgewählt.

Fordert der XScale-Mikroprozessor ein bestimmtes Datenwort aus dem Speicher an und ist es bereits im Cache enthalten, kann es direkt aus dem Cache bereitgestellt werden („cache hit“). Befindet sich das Datenwort nicht im Cache, muss eine komplette Cache-Zeile (32byte) aus dem SDRAM vom SDRAM-Controller angefordert werden („cache miss“). Nach einer gewissen Zugriffszeit (etwa 100ns) werden die angeforderten Daten aus dem Speicher in den Cache transferiert, wobei jene Cache-Zeile überschrieben wird, die zu diesem Zeitpunkt bereits am längsten im Cache verweilt hat. Aufgrund dieser komplexeren Vorgänge ist ein „cache miss“ energetisch wesentlich aufwendiger als ein „cache hit“.

8.3 Experimentelle Ermittlung des Energieaufwandes für Speicherzugriffe

Für die Entwicklung von Methoden zur effizienteren Nutzung eines Cache-Speichers ist die Kenntnis des energetischen Aufwandes für Zugriffe auf den Cache

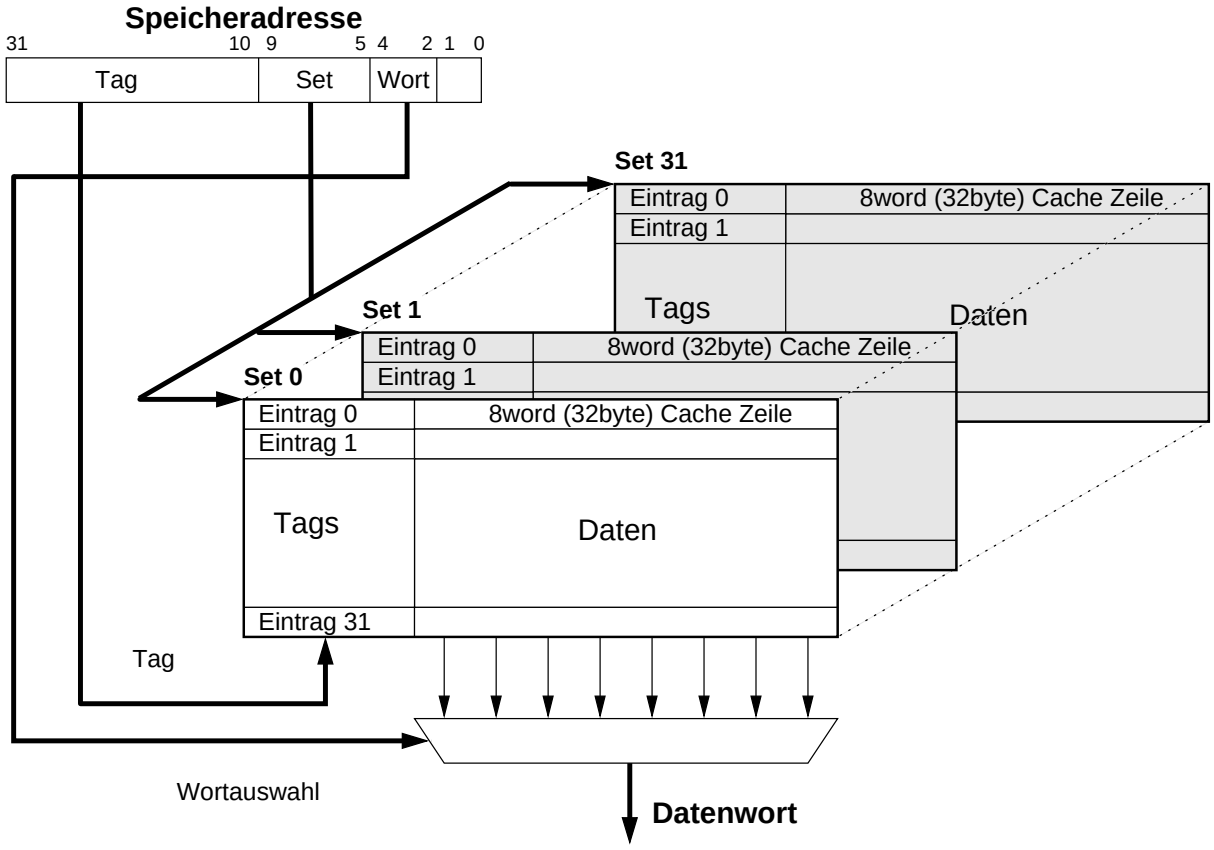


Abbildung 8.2: *Cacheorganisation im Intel XScale.* Die Adressierung einzelner Datenworte erfolgt anhand unterschiedlicher Portionen der Speicheradresse. Die höherwertigen Bits der Speicheradresse bilden die sog. Adresstags, die beim Zugriff auf den Cache mit den bereits abgelegten Tags verglichen werden müssen (inhaltsadressierbarer Speicher). Sowohl Befehls- als auch Datencache des Intel XScale sind 32kbyte groß, verfügen über 32 Sets mit jeweils 32-facher Assoziativität. Cache-Zeilen haben eine Länge von 32byte und werden nach dem Round-Robin Prinzip durch neue Einträge ersetzt.

bzw. auf den Hauptspeicher notwendig. Je höher der Energieaufwand für Hauptspeicherzugriffe ist, desto effizienter greifen die Methoden der Cache-Optimierung. Der Energiebedarf wird anhand spezieller Testprogramme bei gleichzeitiger Leistungs- und Laufzeitmessung ermittelt.

8.3.1 Messvorgang und Testprogramme

Für die Messung des Energiebedarfes E_Z einzelner Speicherzugriffe wurden drei unterschiedliche Programme mit den folgenden Eigenschaften entwickelt:

- `memtest` ist ein Programm, das laufend Zugriffe auf dieselbe Speicherstelle durchführt. Damit kommt die Wirkung des Caches voll zum Tragen, sodass 100% der Speicherzugriffe - mit Ausnahme des zu vernachlässigenden ersten Zugriffes - direkt vom Cache abgedeckt werden können.
- `noptest` ist ein Programm, das ausschließlich `nop`-Maschinenbefehle (No Operation) ausführt.
- `linetest` ist ein Programm, das Speicherzugriffe so durchführt, dass keiner der Zugriffe durch den Cache abgedeckt werden kann, also 100% Cache-Misses auftreten. Dies kann durch Aufspreizung der angeforderten Speicheradressen über das Cache-Fassungsvermögen hinaus erreicht werden.

Die drei oben genannten Programme werden der Reihe nach sowohl bei aktiviertem als auch bei deaktiviertem Cache ausgeführt und die Stromaufnahme des Mikroprozessorsystems gemessen. Als Testsystem wurde ein Intel XScale 80200 Evaluationboard mit der Bezeichnung IQ80310 [170] verwendet, da die Inbetriebnahme des neuen Mikroprozessormoduls zum Zeitpunkt der Tests noch nicht abgeschlossen war.

Die Messergebnisse enthält die Tab. 8.1. P_0 ist die Leistungsaufnahme im Leerlauf, d. h. ohne Ausführung eines Programmes. Nach dem Starten eines Programmes ergibt sich für die Programmabarbeitung ein zusätzlicher Leistungsbedarf ΔP. Die Ausführungszeit t der Programme wird ebenfalls gemessen. Die Programme `memtest`, `noptest` und `linetest` sind so aufgebaut, dass sie jeweils eine willkürlich gewählte, übereinstimmende Anzahl von

$$Z = 1.934 \cdot 10^9 \tag{8.1}$$

Programmschritten ausführen. Im Falle von `memtest` und `linetest` besteht ein Programmschritt aus einem Speicherzugriff, wohingegen ein Programmschritt von `noptest` aus einem `nop`-Befehl besteht. Aufgrund der identischen Anzahl von Programmschritten ist anhand der Ausführungszeit t ersichtlich, dass Speicherzugriffe wesentlich länger dauern als `nop`-Befehle. Während dieser Speicherzugriffe ist

	ohne Cache			mit Cache		
P_0	5.23W			5.29W		
	`noptest`	`memtest`	`linetest`	`noptest`	`memtest`	`linetest`
ΔP	0.435W	1.44W	1.43W	0.432W	0.541W	1.69W
t	3.46s	383s	385s	3.42s	10s	417s
E_Z		199nJ	198nJ		0.56nJ	271nJ

Tabelle 8.1: *Messung des Energiebedarfes für Speicherzugriffe.* Es wird die Leistungsaufnahme P_0 des Testsystems im Leerlauf und die zusätzliche Leistungsaufnahme ΔP bei Ausführung der Programme `noptest`, `memtest` und `linetest` mit und ohne Cache gemessen. Weiters wird die Laufzeit der Programme gemessen und daraus die für einen Speicherzugriff benötigte Energiemenge E_Z bestimmt.

der Mikroprozessor gezwungen, auf die erfolgreiche Ausführung eines Lese- bzw. Schreibzugriffes auf den langsameren Hauptspeicher zu warten. Beim Intel XScale entspricht diese Wartestellung der wiederholten Ausführung von `nop`-Befehlen [97]. Um die benötigte Energiemenge E_Z für einen Speicherzugriff zu ermitteln, muss daher die im Wartezustand verbrauchte Energie berücksichtigt werden. Dies kann anhand der Messungen am Programm `noptest` erfolgen, sodass sich der folgende Zusammenhang für E_Z ergibt:

$$E_Z = \frac{(\Delta P - \Delta P_{\texttt{noptest}})t}{Z}. \tag{8.2}$$

8.3.2 Diskussion der Ergebnisse

Ist der Cache deaktiviert, werden alle Speicherzugriffe durch den Cache-Controller direkt an den Hauptspeicher weitergereicht. In diesem Fall ergibt sich also praktisch kein Unterschied im Energiebedarf E_Z in den Programmen `memtest` und `linetest`. Der Energiebedarf für die Hauptspeicherzugriffe beträgt $E_s = 198$nJ.

Eine andere Situation liegt bei Aktivierung des Cache-Speichers vor. Das Programm `memtest` nutzt die Eigenschaften des Cache voll aus, sodass sich keinerlei Zugriffe auf den Hauptspeicher ergeben, mit Ausnahme des allerersten, der in Anbetracht der großen Anzahl von Zugriffen (siehe Gl.(8.1)) vernachlässigt werden kann. E_Z beträgt in diesem Fall 0.56nJ, was dem Energiebedarf für einen Zugriff auf den Cache entspricht („cache hit“). Damit ergibt sich

$$E_c = E_{hit} = 0.56\text{nJ}. \tag{8.3}$$

Können Speicherzugriffe nicht durch den Cache abgedeckt werden („cache miss“), muss die angeforderte Speicherstelle vom Hauptspeicher in den Cache transferiert werden. Während dieses Transfers lädt der Cache eine komplette Cache-Zeile aus dem Hauptspeicher nach. Das entspricht im Falle des Intel XScale dem Laden eines 32byte langen Speicherblocks, in dem die angeforderte Speicherstelle enthalten ist. Dieses Nachladen ist folglich aufwendiger als der Transfer eines einzelnen Speicherwortes und beträgt

$$E_{miss} = 271\mathbf{nJ}. \tag{8.4}$$

Damit ergibt sich für die erforderlichen Energiemengen bei aktiviertem Cache ein Miss/Hit-Verhältnis (MHV) von

$$MHV = \frac{E_{Z,\texttt{linetest}}}{E_{Z,\texttt{memtest}}} = \frac{E_{miss}}{E_{hit}} = \frac{271\mathbf{nJ}}{0.56\mathbf{nJ}} = 484. \tag{8.5}$$

Dieses hohe MHV widerspiegelt den energetischen Mehraufwand für „cache misses“. Wenn es gelingt, die Anzahl der Cache-Misses deutlich zu senken, ergibt sich ein erheblich geringerer Energiebedarf für die Speicherzugriffe des Prozessors.

Auch der Mehraufwand M_{line} für das Laden einer gesamten Cache-Zeile gegenüber dem Laden von einzelnen Speicherwörtern kann aus der Tab. 8.1 ermittelt werden:

$$M_{line} = \frac{E_{Z,\texttt{linetest}, \text{ mit Cache}}}{E_{Z,\texttt{linetest}, \text{ ohne Cache}}} = \frac{E_{miss}}{E_s} = \frac{271\mathbf{nJ}}{198\mathbf{nJ}} = 1.37. \tag{8.6}$$

Das ist ein überraschendes Ergebnis: Das Laden einer 32byte langen Cache-Zeile erfordert lediglich um 37% mehr Energie als das Laden eines einzelnen, 4byte langen Speicherwortes. Es ist dies eine unmittelbare Folge aus den Eigenschaften von SDRAM Speicher, der einen Großteil der insgesamt für einen Zugriff aufzuwendenden Energie für die Ansteuerung des ersten Datenwortes verbraucht, während das anschließende, sequentielle Laden weiterer Datenwörter im Vergleich dazu mit geringerem Energieaufwand möglich ist (sog. „burst“-Sequenz).

8.4 Untersuchung der Cache-Effizienz durch Einsatz eines Hardware-Simulators

Eine notwendige Voraussetzung für die Ermittlung der Auslastung einer Cache-Struktur bei gegebenem Programm ist die genaue Kenntnis über die laufende Belegung der Cache-internen Ressourcen während der Ausführung des Programmes. In einem realen Mikroprozessor ist diese Information nicht zugänglich. Eine Alternative stellen Hardware-Simulatoren dar, in denen das Programm abgearbeitet wird. Der Simulator kann dabei so gestaltet werden, dass an beliebigen Stellen des

Simulationsmodells während der Programmausführung Informationen mitprotokolliert werden (Gewinnung von sog. „Traces“).

8.4.1 Methodik

Der im Abschnitt 7.3 beschriebene, neu entwickelte XScale-Simulator wird für die Untersuchung der Cache-Effizienz derart modifiziert, dass an der Schnittstelle zwischen Mikroprozessor und Hauptspeicher sämtliche Speicherzugriffe aufgezeichnet werden können. Nach erfolgter Programmausführung sind diese Aufzeichnungen offline durch entsprechende Analysesoftware auszuwerten.

Der Simulator generiert während der Programmabarbeitung zwei Protokolldateien: Die erste Datei enthält die Zugriffe auf den Programmspeicher bzw. Befehlscache, während die zweite Datei alle Zugriffe auf den Datenspeicher enthält. Eine neu entwickelte Analysesoftware bildet die Cache-Speicher des XScale (Abb. 8.2) durch ein Softwaremodell nach und simuliert das genaue Verhalten des Caches unter Zuhilfenahme der Protokolldateien. Daraus lassen sich wichtige Kennzahlen ermitteln, die eine Beurteilung der Cache-Effizienz ermöglichen:

- Gesamtanzahl K_{ges} der Zugriffe des Mikroprozessors auf den Speicher. Diese Größe dient in erster Linie als Bezugswert.
- Anzahl der „cache hits“ K_{hit} : Wie oft kann ein Datenwort durch den Cache zur Verfügung gestellt werden, ohne auf den Hauptspeicher zurückgreifen zu müssen?
- Anzahl der „cache misses“ K_{miss} : Wie oft muss ein Datenwort aus dem Hauptspeicher (durch Anfordern einer kompletten Cache-Zeile) nachgeladen werden?
- Transferierte Speicherworte K_{trans} : Gesamtanzahl von 32bit-Speicherworten, die zwischen Cache und Hauptspeicher transferiert werden.
- Geladene Speicherworte K_{load} : Anzahl der tatsächlich vom Mikroprozessor aus dem Cache geladenen Speicherworte. Mit dieser Kennzahl lässt sich der Overhead, der aufgrund des ausschließlichen Ladens kompletter Cache-Zeilen entsteht, abschätzen.

Aus diesen absoluten Kennzahlen lassen sich die relativen Kenngrößen k_{hit}, k_{miss} sowie k_{load} ermitteln:

$$\begin{aligned} k_{hit} &= \frac{K_{hit}}{K_{ges}} 100\% \\ k_{miss} &= \frac{K_{miss}}{K_{ges}} 100\% \\ k_{load} &= \frac{K_{load}}{K_{trans}} 100\% \end{aligned} \tag{8.7}$$

8.4.2 Auswahl von Testprogrammen

Für die Aufzeichnung der Speicherzugriffe wird ein Satz von Testprogrammen ausgewählt, der das Aufgabenspektrum der zugrundeliegenden Hardware möglichst gut abdeckt. Die Anzahl der Testprogramme sollte gering sein, um die Ausführungszeiten wegen des hohen Simulationsaufwandes in vertretbaren Grenzen zu halten.

Die Tab. 8.2 enthält die fünf Programmpakete, welche für die Untersuchung der Cache-Effizienz ausgewählt wurden. Die gesamte Ausführungsdauer dieser Programme im Simulator beträgt je nach Rechnerausstattung etwa drei bis fünf Minuten.

Paket	Applikation
`cjpeg`	Konvertierung eines Bildes in das JPEG-Format (DCT)
`sigproc`	Signalverarbeitungsalgorithmen: FFT, Filter, Faltung
`mpeg`	Codierung einer Sprachsequenz im MPEG-Format
`gzip`	Komprimieren einer Datei durch GNU-Zip
`dhry`	Dhrystone v2.1 Benchmark Programm

Tabelle 8.2: *Testprogramme für die Untersuchung der Cache-Effizienz.* Die angegebenen fünf Programmpakete werden der Reihe nach im Simulator ausgeführt und die dabei entstehenden Speicherzugriffe mitprotokolliert.

Die Testprogramme wurden mithilfe der in Kapitel 7 beschriebenen Entwicklungsumgebung übersetzt und unter dem Betriebssystem Linux unmittelbar aufeinanderfolgend im Simulator ausgeführt.

8.4.3 Effizienzkennzahlen für den Cache ohne energieoptimierte Cachezuteilung

Nach Ausführung der Testprogramme aus Tab. 8.2 im XScale-Simulator wurden die erzeugten Protokolldateien für Daten- und Befehlscache von der für diesen Zweck entwickelten Analysesoftware ausgewertet. Die Ergebnisse sind in der Tab. 8.3 wiedergegeben.

Anhand von K_{ges} ist aus Tab. 8.3 ersichtlich, dass Zugriffe auf den Datencache (bei Ausführung der Testprogramme aus Tab. 8.2) wesentlich häufiger stattfinden

	Datencache		Befehlscache	
	Anzahl	*Anteil*	*Anzahl*	*Anteil*
Gesamtanzahl Zugriffe K_{ges}	$26.1 \cdot 10^6$	100%	$1.54 \cdot 10^6$	100%
Anzahl Cache-Hits K_{hits}, k_{hits}	$25.7 \cdot 10^6$	98.74%	$1.48 \cdot 10^6$	95.9%
Anzahl Cache-Misses K_{miss}, k_{miss}	$329 \cdot 10^3$	1.26%	$63.8 \cdot 10^3$	4.15%
transferierte Speicherworte K_{trans}	$2.63 \cdot 10^6$	100%	$511 \cdot 10^3$	100%
davon verwendet K_{load}, k_{load}	$1.33 \cdot 10^6$	50.65%	$374 \cdot 10^3$	73.2%

Tabelle 8.3: *Zugriffsstatistik für Befehls- und Datencache.* Die Tab. zeigt Effizienzkennzahlen in absoluten und prozentuellen Größen getrennt für Befehls- und Datencache. Die Auswertungen wurde von einer Analysesoftware vorgenommen, die für diesen Zweck entwickelt wurde.

als auf den Befehlscache. Der Unterschied beläuft sich etwa auf das Verhältnis 1:17, sodass die Optimierung des Datencaches zunächst als wesentlich wichtiger erscheint.

Bei Betrachtung von k_{hit} bzw. k_{miss} fällt allerdings auf, dass die Anzahl der „cache-misses“ im Falle des Befehlscaches (4.15%) wesentlich höher ist als beim Datencache (1.26%). Deshalb ist der Unterschied zwischen Befehls- und Datencache in der Anzahl der transferierten Speicherworte weniger drastisch.

Interessant ist der tatsächlich vom Mikroprozessor verwendete Anteil k_{load} an der Gesamtanzahl der in den Cache transferierten Datenworte. Während 73.2% der Datenworte aus dem Befehlscache tatsächlich Verwendung finden, sind dies im Falle des Datencaches lediglich 50.65%. Dieser vehemente Unterschied ist eine Folge der wahlfrei verteilten Zugriffe auf Programmdaten im Unterschied zur linearen, sequentiellen Abarbeitung von Programmcode .

Mit der Kenntnis des Energiebedarfes für Speicherzugriffe aus Tab. 8.1 lässt sich der gesamte Energiebedarf E_{ges} für sämtliche Speicherzugriffe auf den Daten- bzw. Befehlscache bei Abarbeitung der Testprogramme mit

$$\begin{aligned} E_{ges} &= K_{hit}E_{hit} + K_{miss}E_{miss} \\ \tfrac{E_{ges}}{K_{ges}} &= k_{hit}E_{hit} + k_{miss}E_{miss} \end{aligned} \tag{8.8}$$

in absoluter bzw. in bezogener Form berechnen. Unter Verwendung der Gln. (8.3) und (8.4) folgt mit den Werten aus der Tab. 8.3

$$\frac{E_{ges}}{K_{ges}} = \underbrace{0.959 \cdot 0.56\mathbf{nJ}}_{0.537\mathbf{nJ}} + \underbrace{0.0415 \cdot 271\mathbf{nJ}}_{11.2\mathbf{nJ}} = 11.8\mathbf{nJ} \tag{8.9}$$

für den Befehlscache und

$$\frac{E_{ges}}{K_{ges}} = \underbrace{0.9874 \cdot 0.56\mathbf{nJ}}_{0.553\mathbf{nJ}} + \underbrace{0.0126 \cdot 271\mathbf{nJ}}_{3.41\mathbf{nJ}} = 3.97\mathbf{nJ} \tag{8.10}$$

für den Datencache. *Daraus ist deutlich erkennbar, dass selbst für sehr kleine k_{miss} von 0.55% im Falle des Datencaches der gesamte Energiebedarf in erster Linie durch die „cache-misses“ bestimmt wird und dass die Anzahl der „cache-hits“ dabei eine untergeordnete Rolle spielt.* Für eine Steigerung der Cache-Effizienz ist damit das Hauptaugenmerk auf eine Reduktion von K_{miss} zu legen, wobei eine gleichzeitige Erhöhung von K_{hit} bis zur Grenze von $|\Delta K_{hit}/\Delta K_{miss}| < MHV$ sogar in Kauf genommen werden kann.

Durch die Offlineanalyse der aus dem XScale-Simulator gewonnenen Protokolldateien besteht auch die Möglichkeit, einzelne Komponenten des Caches einer genauen Betrachtung zuzuführen. So gibt die Untersuchung der Ausnutzung einzelner Cache-Zeilen wichtige Aufschlüsse über Möglichkeiten zur Steigerung der Cache-Effizienz. Mithilfe der Analysesoftware konnte so ermittelt werden, wie viele Speicherworte aus den 32byte (= 8word) langen Cache-Zeilen im Mittel tatsächlich vom Mikroprozessor verwendet werden.

Das Ergebnis dieser Analyse ist in der Abb. 8.3 getrennt für Befehls- und Datencache dargestellt. Auf der Abszisse findet sich der Verwendungsgrad einer Cache-Zeile (Anzahl tatsächlich benutzter Speicherwörter) von einem bis zu acht Speicherworten. Auf der Ordinate ist die zugehörige Häufigkeit für den jeweiligen Verwendungsgrad aufgetragen. Folgende Kernaussagen sind aus dieser Darstellung abzuleiten:

- Im Schnitt werden komplette Cache-Zeilen nur in 35% der Fälle beim Datencache und in 45% der Fälle beim Befehlscache vollständig verwendet, d. h. der Prozessor fordert in diesen Fällen alle acht Datenworte einer Cache-Zeile auch tatsächlich im Zuge der Programmausführung an. Die bessere Ausnutzung einer Befehlscache-Zeile ist auf das lineare, sequentielle Ausführen von Programmcode zurückzuführen. Beim Datencache wird eine hohe Ausnutzung der Cache-Zeilen dann erzielt, wenn große Datenpakete sequentiell abgearbeitet werden müssen (z. B. Verarbeitung von Datenströmen oder langen Zeichenketten).

- Beim Datencache liegt eine auffällige Häufung von Cache-Zeilen vor, von denen nur ein einziges Speicherwort verwendet wird. Mit 38% ist dies sogar häufiger der Fall, als die Verwendung aller acht Speicherworte. Dies ist dadurch zu erklären, dass auf Programmdaten in der Regel wahlfrei und sporadisch und nicht sequentiell zugegriffen wird.

- Für den Befehlscache ist (abgesehen von der vollständigen Verwendung aller acht Speicherwörter) die Anzahl der verwendeten Speicherwörter etwa

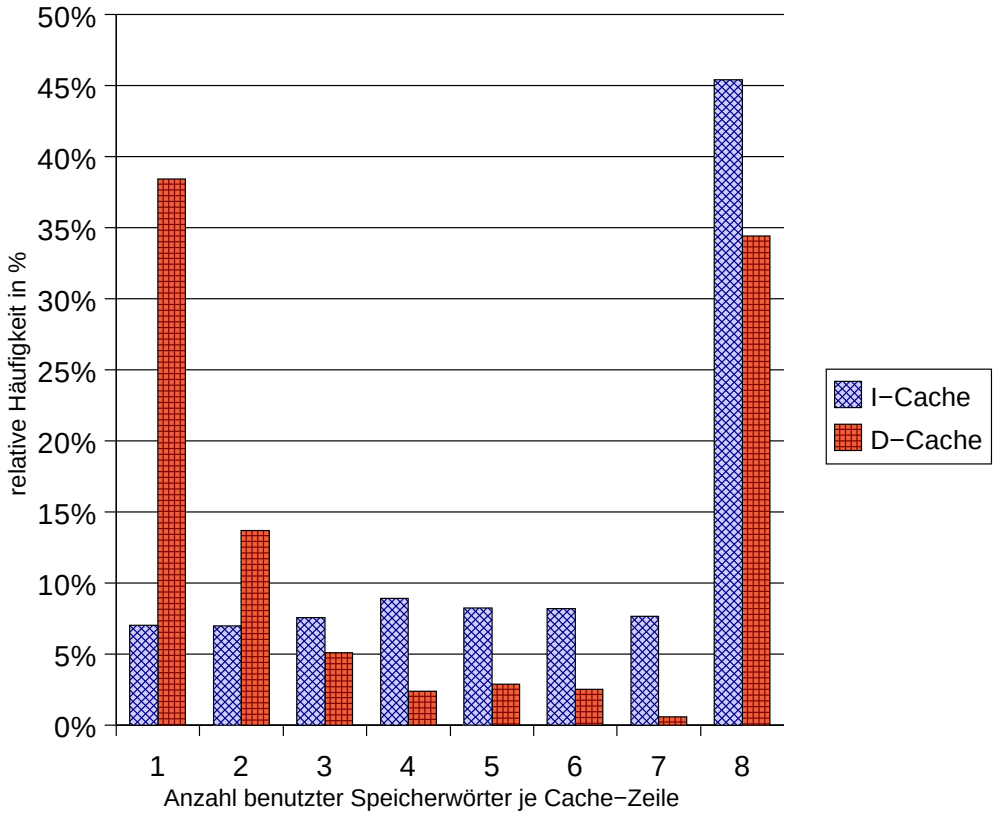


Abbildung 8.3: *Mittlere Ausnutzung von Cache-Einträgen.* Im Befehlscache werden in 45% der Fälle alle acht Speicherworte der Cache-Zeile ausgenutzt. Ansonsten ist die Anzahl der verwendeten Speicherwörter je Cache-Zeile beim Befehlscache etwa gleichverteilt. Beim Datencache wird von einer Cache-Zeile in den häufigsten Fällen entweder nur eines oder alle acht Speicherwörter verwendet.

gleichverteilt. Das ist eine Konsequenz aus der Beschaffenheit der ausgeführten Programmblöcke und Prozeduren, die vom Compiler bzw. Linker linear im Adressraum des Programmes platziert werden und dadurch in ihren jeweils modulo acht genommenen Anfangs- bzw. Endadressen der Programmblöcke eine Gleichverteilung vorliegt.

In Anbetracht des hohen $k_{hit} = 0.9945$ für den Datencache verlangt die Häufung der nur in einem einzigen Speicherwort verwendeten Datencache-Zeilen nähere Untersuchung. Ein aufschlussreiches Ergebnis liefert die Analysesoftware bei Auswertung der Anzahl von Zugriffen auf Cache-Zeilen, aus denen nur ein einziges Speicherwort verwendet wird. Damit kann festgestellt werden, ob der Mikroprozessor das einzelne Speicherwort mehrmals oder nur einmal lädt.

Die Abb. 8.4 stellt das Ergebnis dieser Analyse grafisch dar. Untersucht wurden ausschließlich Cache-Zeilen mit einem Verwendungsgrad von einem einzigen Speicherwort. Auf der Abszisse ist die Anzahl von Zugriffen durch den Mikroprozessor auf dieses Speicherwort aufgetragen. Obwohl in Einzelfällen wesentlich mehr als zehn Zugriffe erfolgen (z. B. auf wichtige globale Variablen eines Programmes), sind die numerischen Werte für die zugehörigen Häufigkeiten ab einer Anzahl von zehn Zugriffen unbedeutend.

Das wesentliche Ergebnis ist die hohe Anzahl (>60%) von Einzelwortzugriffen, d. h. aus einer Cache-Zeile wird ein einziges Speicherwort auch nur ein einziges mal verwendet. Diese Konstellation ist besonders ungünstig für eine effiziente Nutzung vorhandener Cache-Strukturen und gibt berechtigten Anlass zu einer Optimierung in diesem Bereich.

8.5 Neue Strategien zur Steigerung der Cache-Effizienz

Eine effizientere Nutzung bestehender Cache-Ressourcen ermöglicht aufgrund der geringeren Anzahl von Zugriffen auf den Hauptspeicher eine Reduktion der für eine bestimmte Aufgabe erforderlichen Energiemenge bei gleichzeitiger Steigerung der Ausführungsgeschwindigkeit von Programmen.

Für die im Mikroprozessorsystem auftretende Verlustleistung bedeutet dies, dass die Einsparungen bei den Speicherzugriffen teilweise durch die höheren Verarbeitungsgeschwindigkeiten wieder kompensiert werden. Tatsächlich führt eine höhere Performance aber wiederum zu längeren Pausenzeiten („idle“-Zustand), da in Embedded Systemen der Umfang an insgesamt auszuführenden Berechnungsaufgaben genau vorgegeben ist. Im Mittel sinkt daher die auftretende Verlustleistung durch eine gesteigerte Effizienz der Cache-Speicher.

Eine neue Strategie zur Steigerung der Cache-Effizienz besteht in der Modifikation des Speicherlayoutes eines Programmes in der Weise, dass bei Abarbeitung des

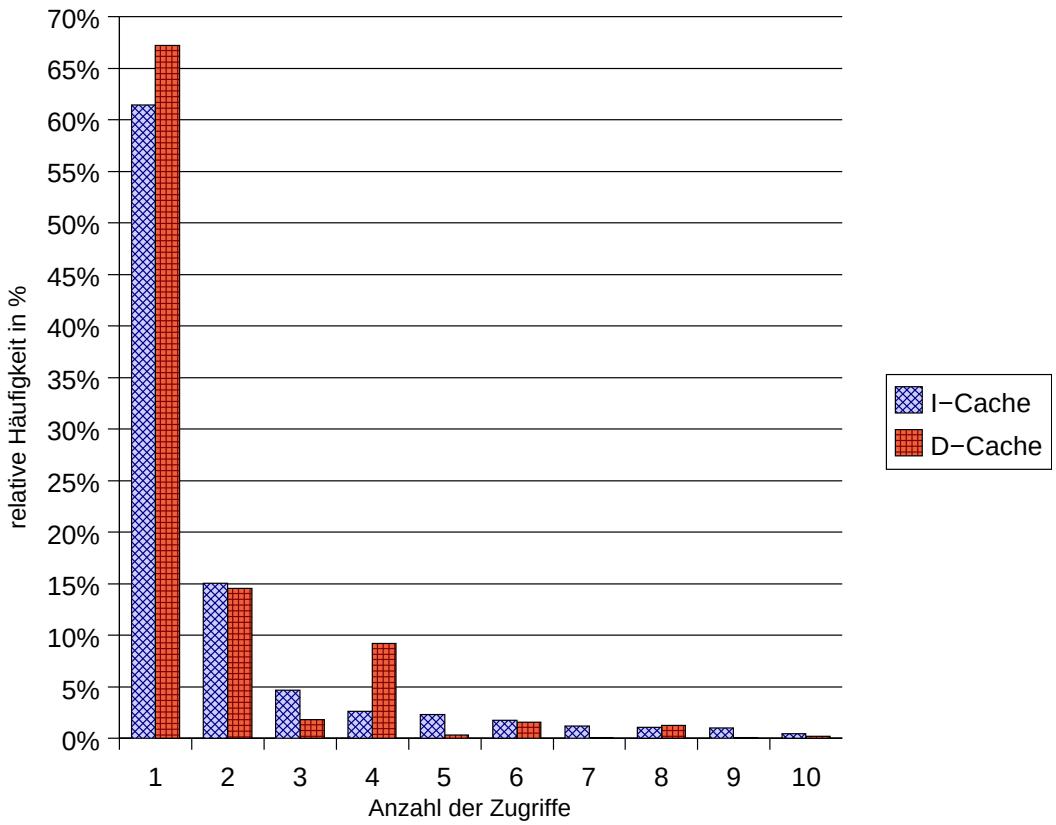


Abbildung 8.4: *Anzahl der Zugriffe auf Cache-Zeilen in denen nur ein Speicherwort verwendet wird.* Sowohl im Befehls- als auch im Datencache wird in mehr als 60% der Fälle ein einziges Speicherwort einer Cache-Zeile auch nur ein einziges mal verwendet, was sich besonders ungünstig auf die Effizienz des Caches auswirkt. Mehr als vier Zugriffe sind kaum wahrscheinlich.

Programmes die vorgegebenen Strukturen der Cache-Speicher energieoptimal ausgenutzt werden.

Die spezielle Gestaltung des Speicherlayoutes stellt ein Optimierungsproblem dar, das durch rein kombinatorisches Durchprobieren möglicher Varianten nicht gelöst werden kann.

Als Eingangsdaten für das Optimierungsproblem dient die bei Ausführung eines Programmes (inkl. Betriebssystem) vom Mikroprozessor durchgeführte Folge von Speicherzugriffen. Die Abfolge dieser Speicherzugriffe (Adressraumspur) wird mithilfe eines Simulators gewonnen, in dem das Programm inkl. Betriebssystem zur Ausführung gelangt.

8.5.1 Gesamtheitliche Betrachtung von Applikation und Betriebssystem

Bei Ausführung eines Programmes bildet die Reihe der dabei auftretenden Speicherzugriffe eine komplizierte, nicht zusammenhängende Adressraumspur durch den kompletten Speicher eines Mikroprozessorsystems. Je nach Programmgröße und Eigenschaften der ausgeführten Algorithmen wird durch diese Spur ein mehr oder weniger großer Adressbereich überstrichen. Je kompakter diese Spur gestaltet ist, desto besser kann ein Cache-Speicher ausgenutzt werden.

Eine gezwungenermaßen vorgegebene Unstetigkeit in der Adressraumspur bedeuten die von einem Programm ausgeführten Betriebssystemaufrufe. Dabei wird die Programmausführung an der Betriebssystemschnittstelle („system call“, Abb. 8.5) in den geschützten Adressraum des Betriesbssystems verlagert. Trotz der getrennten virtuellen Adressräume wird bei Abarbeitung von Programmcode des Betriebssystem auf denselben Befehls- und Datencache zugegriffen wie im Anwenderprogramm.

Daraus ergibt sich die unmittelbare Konsequenz, dass für die Steigerung der Cache-Effizienz bei einer gegebenen Aufgabe wegen der gemeinsamen Verwendung derselben Cache-Strukturen stets die Gesamtheit aus Betriebssystem und Anwenderprogrammen in Betracht zu ziehen ist.

Bestehende Methoden [52, 53, 54, 55] versuchen durch alleinige Modifikation der Anwenderprogramme (Änderung des Speicherlayouts, der Algorithmen und Datenstrukturen) eine Effizienzsteigerung herbeizuführen. Im Falle von Programmen, die auf zahlreiche Funktionen des Betriebssystems zurückgreifen ist die Wirksamkeit dieser Methoden jedoch beschränkt, da jeder Systemaufruf den Inhalt der Caches in einem unbekannten Ausmaß verändert.

Mithilfe des im Rahmen der vorliegenden Arbeit entwickelten Simulators ist die gesamtheitliche Betrachtung von Betriebssystem und Anwenderprogrammen ohne Mehraufwand problemlos durchführbar, da die vom Simulator vorgenommene

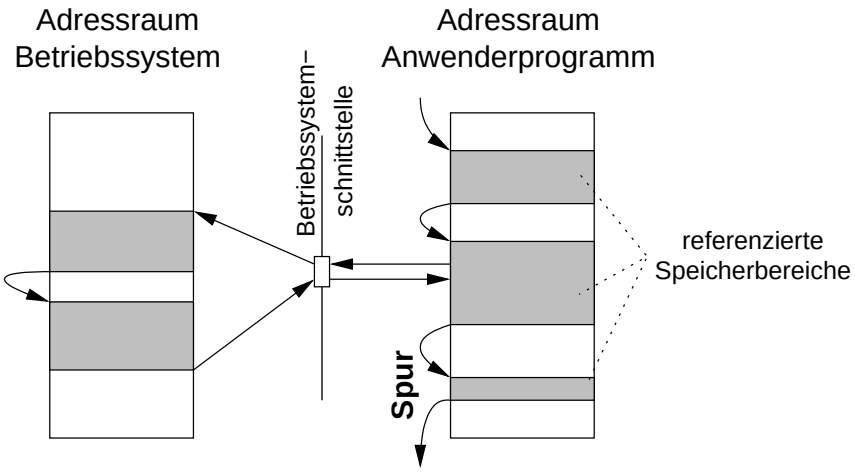


Abbildung 8.5: *Adressraumspur.* Die Ausführung eines Programmes erzeugt durch die Reihe von Speicherzugriffen eine Spur im Adressraum. Durch Aufruf von Funktionen des Betriebssystems entstehen Unstetigkeiten in dieser Spur.

Protokollierung der Speicherzugriffe uneingeschränkt auch für das Betriebssystem gilt.

8.5.2 Statische Blockoptimierung

Grundprinzip der statischen Blockoptimierung ist eine Steigerung der räumlichen Lokalität eines Programmes, indem häufig hintereinander benutzte Programmteile in unmittelbarer Nähe zueinander im Adressraum angeordnet werden.

Die Verbesserung der räumlichen Lokalität wirkt sich positiv auf die Ausnutzung von Caches aus. Diese Methode ist als statisch zu bezeichnen, weil die Anordnung der Blöcke während des Programmablaufes nicht abgeändert werden kann, also statisch vorgegeben ist.

Methodik der statischen Blockoptimierung

Für die vollständige Ausführung eines Programmes fordert der Mikroprozessor insgesamt N Speicherblöcke B_i ($i = 1, 2, \ldots, N$) vom Hauptspeicher an, wobei die Blöcke B_i eine Länge l_i in byte aufweisen und im nicht optimierten Speicherlayout die Adressintervalle $[a_i\,,\, a_i + l_i[$ belegen. Die Programmausführung entspricht somit einer Kette von M aufeinanderfolgend angeforderten einzelnen Blöcken, deren Indizes zur Ausführungsreihenfolge J zusammengefasst werden:

$$J = \{j_1, j_2, j_3, ..., j_M\} \text{ mit } 1 \leq j_k \leq N, k = 1, \ldots, M. \tag{8.11}$$

Durch Analyse dieser Ausführungsreihenfolge J lässt sich die Übergangshäufigkeit p_{xy} zwischen zwei Blöcken B_x und B_y durch Abzählen der Übergänge $x \rightarrow y$ in J einfach ermitteln. Für alle N Blöcke ergibt sich somit eine Übergangsmatrix

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1N} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{NN} \end{pmatrix}. \tag{8.12}$$

Die Elemente der Diagonale p_{xx} geben dabei die Anzahl der Schleifendurchläufe des Blocks B_x an. Im Allgemeinen gilt $p_{xy} \neq p_{yx}$, sodass P nicht symmetrisch ist.

Aus der Übergangsmatrix P lässt sich die gegenseitige Übergangsmatrix Q entwickeln, deren Elemente q_{xy} die gegenseitige oder wechselseitige Übergangshäufigkeit

$$\begin{aligned} q_{xy} &= p_{xy} + p_{yx} \\ q_{xx} &= p_{xx} \\ q_{yx} &= q_{xy} \end{aligned} \tag{8.13}$$

darstellen. Im Gegensatz zu P ist Q eine symmetrische Matrix und kann somit in einer softwaretechnischen Implementierung als obere oder untere Dreiecksmatrix ressourcenschonender abgesetzt werden.

Der statischen Blockoptimierung liegt das Prinzip zugrunde, dass Blöcke, die vom Mikroprozessor häufig nacheinander angefordert werden auch physikalisch im Speicher aufeinanderfolgend zu liegen kommen. Dadurch wird die Lokalität der Speicherzugriffe gesteigert und vorhandene Cache-Speicher effizienter genutzt.

Für einen Optimierungsvorgang bedeutet dies, aus den N Blöcken zunächst jene zwei Blöcke B_μ und B_ν auszuwählen, für die die gegenseitige Übergangshäufigkeit $q_{\mu\nu}$ ein Maximum ist:

$$q_{\mu\nu} = \max_{x=1\ldots N, y=1\ldots N, x\neq y} q_{xy}. \tag{8.14}$$

Anschließend werden die beiden Blöcke B_μ und B_ν zu einem gemeinsamen Block B_ξ zusammengefasst, wobei die gegenseitigen Übergangshäufigkeiten gemäß den Zusammenhängen

$$\begin{aligned} q_{x\xi} &= q_{x\mu} + q_{x\nu} \\ q_{\xi\xi} &= q_{\mu\mu} + q_{\nu\nu} + q_{\mu\nu} \\ q_{\xi x} &= q_{x\xi} \end{aligned} \tag{8.15}$$

zu modifizieren sind. Das Verfahren wird so lange fortgesetzt, bis schließlich alle Blöcke zu einem einzigen, gemeinsamen Block zusammengefasst sind.

Die sukzessive Zusammenfassung der Blöcke ist in der Abb. 8.6a-e graphisch anhand eines einfachen Beispiels mit fünf Blöcken veranschaulicht. Unterhalb der

einzelnen Graphen ist die zugehörige wechselseitige Übergangsmatrix Q angegeben, in der jenes Element hervorgehoben wurde, das der Gl. (8.14) genügt. Nach vier Iterationsschritten endet das Verfahren.

Implementierung in Software

Die Umsetzung der statischen Blockoptimierung in Software ist z. T. mit erheblichen Schwierigkeiten verbunden, da beispielsweise die Protokolldatei für die Aufzeichnung der Datencachezugriffe über 200Mbyte groß ist und dabei etwa 480000 Speicherblöcke umfasst. Allein die wechselseitige Übergangsmatrix Q würde, ein Byte je Matrixelement gerechnet, eine Größe von 115Gbyte aufweisen.

Eine Implementierung wird nur dadurch möglich, indem von der konventionellen Darstellungsweise einer Matrix in Form eines zweidimensionalen Arrays abgegangen wird. Die Tatsache, dass einzelne Speicherblöcke zumeist in einem relativ engen Programmkontext verwendet werden, verursacht mangels vorhandener Übergänge zwischen vielen Blöcken zahlreiche Nullelemente in der Matrix Q. Dies ermöglicht ein wesentlich ressourcenschonenderes Ablegen der Matrixelemente in assoziativen, sortierten Container-Strukturen [169]. Ein schneller wahlfreier Zugriff auf einzelne Elemente geht dadurch zwar verloren, wird aber durch das sortierte Ablegen der Daten in vertretbarem Ausmaß kompensiert. Durch diese Maßnahme lässt sich der erforderliche Speicherbedarf für die Implementierung der statischen Blockanalyse auf etwa 300Mbyte reduzieren, was auf heutigen Arbeitsplatzrechnern kein gravierendes Problem mehr darstellt.

Die für eine softwaretechnische Umsetzung der statischen Blockoptimierung notwendigen Einzelschritte sind in der Tab. 8.4 aufgelistet:

- *Schritt 1: Einlesen der Protokolldatei und Extrahieren von Speicherblöcken.* Die aus der Protokolldatei gelesenen Speicherzugriffe sind einer Analyse zu unterwerfen, um sequentiell adressierte Blöcke daraus zu ermitteln. Wegen der zahlreichen bedingten Verzweigungen bei Ausführung eines Programms kommt es häufig vor, dass zunächst sequentiell durchlaufene große Speicherblöcke in mehrere kleinere Blöcke unterteilt werden müssen. Das Resultat aus Schritt 1 ist jedenfalls die Menge der Blöcke B_i mit der Ausführungsreihenfolge J.

- *Schritt 2: Berechnung der wechselseitigen Übergangsmatrix Q.* Mithilfe der Ausführungsreihenfolge J und den Berechnungsvorschriften aus Gln. (8.12) und (8.13) kann die sehr ressourcenintensive Matrix Q ermittelt werden. Aufgrund der Symmetrie wird Q als rechte obere Dreiecksmatrix implementiert.

- *Schritt 3a: Suchen der maximalen Übergangshäufigkeit gemäß Gl. (8.14).* Die maximale wechselseitige Übergangshäufigkeit $q_{\mu\nu}$ wird mittels Durchsuchen der Matrixelemente von Q festgestellt. Da ein lineares Suchen enorm viel

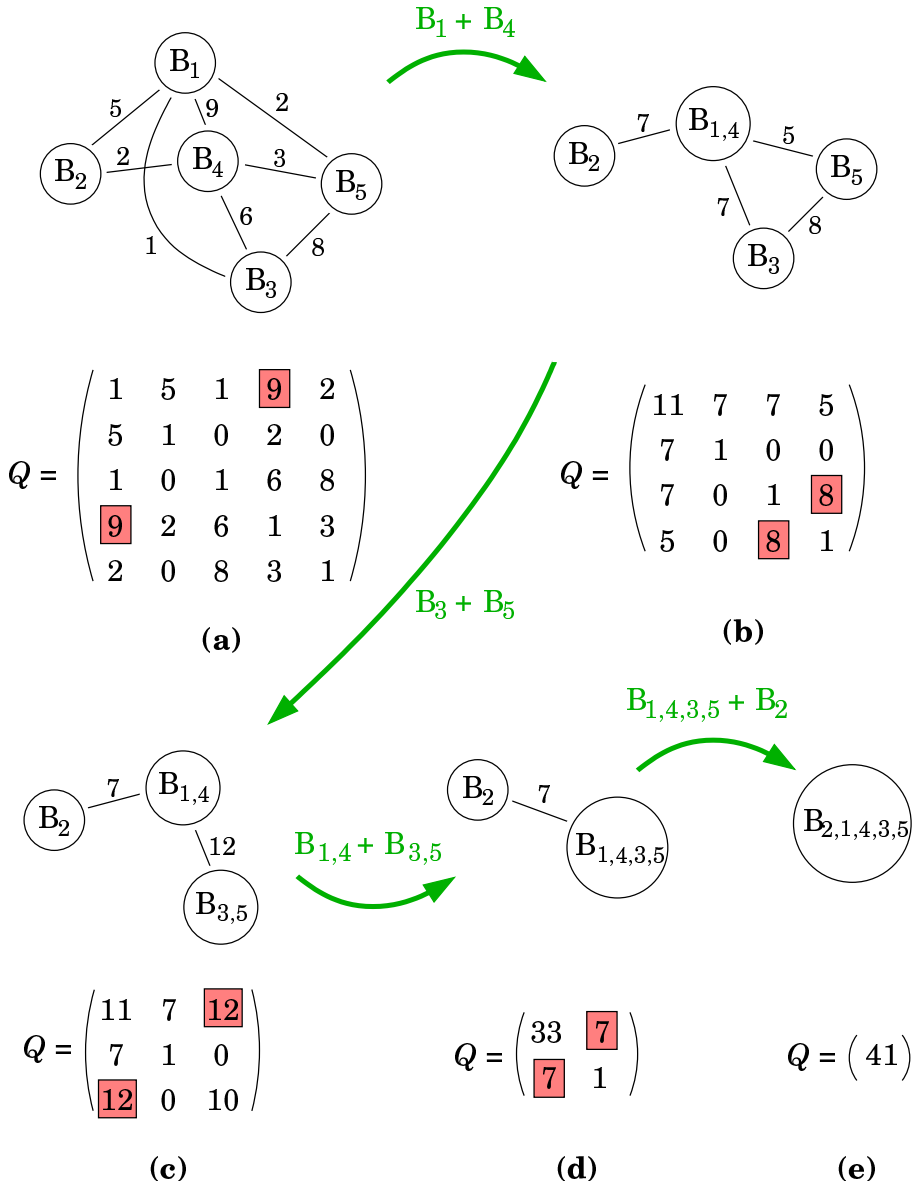


Abbildung 8.6: *Statische Blockoptimierung durch Zusammenfassen von Programmblöcken.* Die fünf Programmblöcke B_1 bis B_5 in (a) werden durch vier Iterationsschritte zu einem einzigen Block in (e) zusammengefasst. Die Blöcke sind in Form eines Graphen mit den zugehörigen Übergangshäufigkeiten dargestellt. Unterhalb der einzelnen Graphen ist die zugehörige wechselseitige Übergangsmatrix Q angegeben, wobei jene Elemente mit der maximalen wechselseitigen Übergangshäufigkeit hervorgehoben sind. Die Pfeile zwischen den Teilabb. geben an, welche Blöcke zusammengefasst werden.

Schritt	Aufgabe
1	Einlesen der Protokolldatei und Extrahieren von Speicherblöcken
2	Berechnung der wechselseitigen Übergangsmatrix Q
3	Statische Blockoptimierung a) Suchen der maximalen Übergangshäufigkeit gemäß Gl. (8.14) und b) Zusammenfassen der beiden Blöcke c) Speichern des zusammengefassten Blocks und Modifikation von Q wiederhole Schritt 3 bis Q die Dimension 1×1 aufweist
4	Feststellung der optimierten Blockreihenfolge
5	Zuteilung von Adressen an die einzelnen Blöcke

Tabelle 8.4: *Einzelschritte des Algorithmus zur softwaretechnischen Umsetzung der statischen Blockoptimierung.*

Rechenzeit in Anspruch nimmt, wird stattdessen eine vorsortierte Liste zum Auffinden des Maximums herangezogen.

- *Schritt 3b: Zusammenfassen der beiden Blöcke.* Jene zwei Blöcke mit der maximalen Übergangshäufigkeit $q_{\mu\nu}$ werden zusammengefasst. Dies erfolgt durch Summieren der entsprechenden Elemente aus der Matrix Q gemäß der Gl. (8.15). Um die zur Berechnung notwendigen Summanden in der Matrix Q einfach aufzufinden, erweist sich ein Durchlaufen der Indizes nach dem Muster von Abb. 8.7a als vorteilhaft.

- *Schritt 3c: Speichern des zusammengefassten Blocks und Modifikation von Q.* Jener an der Zusammenfassung beteiligte Block mit dem kleineren Spaltenindex in der Matrix Q wird durch den neuen zusammengefassten Block überschrieben (Abb. 8.7b). Der verbleibende Block mit dem größeren Spaltenindex wird aus der Matrix Q gelöscht, wodurch sich die Dimension von Q um eine Zeile und eine Spalte verringert (Abb. 8.7c).

- *Schritt 4: Feststellung der optimierten Blockreihenfolge.* Die im Schritt 3 paarweise zusammengefassten Blöcke werden unter Zuhilfenahme einer Baumstruktur aneinandergereiht (Abb. 8.8).

- *Schritt 5: Zuteilung von Adressen an die einzelnen Blöcke.* Die im Schritt 4 aneinandergereihte Kette aus Blöcken wird von vorne nach hinten durchlau-

fen und die Anfangsadressen der einzelnen Blöcke festgelegt. Damit ist das optimierte Speicherlayout eines Programmes fixiert.

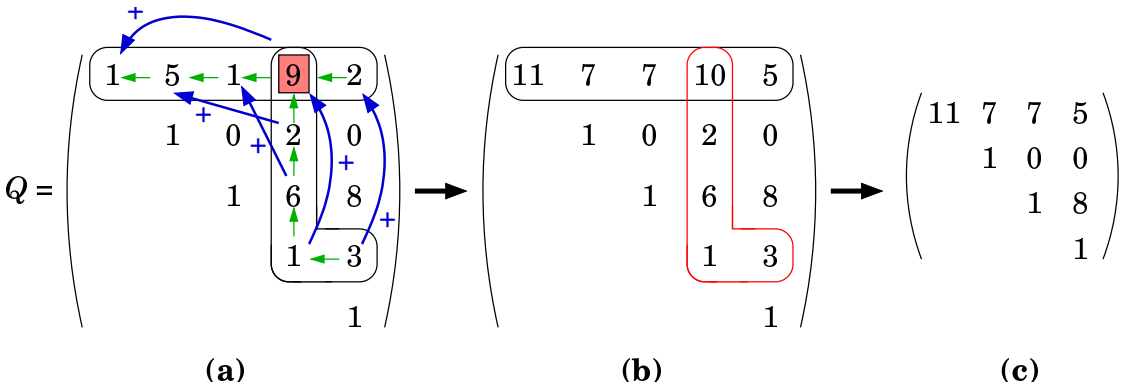


Abbildung 8.7: *Veranschaulichung der Blockzusammenfassung aus Abb. 8.6 a und b.* (a) Durchlaufreihenfolge der Indizes für die Summierung nach Gl. (8.15). Es wird die Zeile und die Spalte des Matrixelementes mit der maximalen Übergangshäufigkeit richtung fallende Indizes durchlaufen (kleine grüne Pfeile). Ragt die markierte Zeile bzw. Spalte über die Diagonale der Dreiecksmatrix hinaus, wird sie unter Ausnutzung der Symmetrie in transponierter Richtung fortgesetzt. Die größeren blauen Pfeile kennzeichnen die vorgenommenen Additionen. (b) Die Ergebnisse der Additionen stellen die Übergangshäufigkeiten der zusammengefassten zwei Blöcke dar und überschreiben die vorhandenen Matrixelemente des ersten Blocks. Die rot eingefassten Matrixelemente des zweiten Blocks werden entfernt. (c) Resultierende Matrix Q nach erfolgter Blockzusammenfassung.

Verbesserung der Cache-Effizienz durch statische Blockoptimierung

Die bei Ausführung der Testprogramme aus Tab. 8.2 mitprotokollierten Speicherzugriffe werden der statischen Blockoptimierung unterzogen, um dessen Wirksamkeit feststellen zu können. Nach Anwendung der Optimierungsmethode werden analog zum Abschnitt 8.4.3 die Effizienzkennzahlen für den Cache mithilfe der Analysesoftware bestimmt.

Die Ergebnisse sind in der Tab. 8.5 eingetragen. Ein Vergleich mit dem nicht optimierten Fall zeigt eine deutliche Verringerung von k_{miss} von 1.26% auf 0.83% für den Datencache bzw. von 4.15% auf 3.22% für den Befehlscache und unterstreicht damit die Wirksamkeit der statischen Blockoptimierung. Auch die Anzahl der zwischen Cache und Hauptspeicher transferierten Datenworte hat sich durch

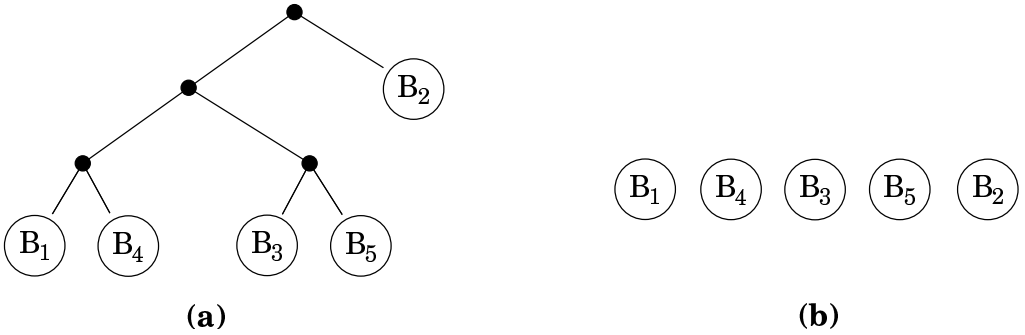


Abbildung 8.8: *Baumstruktur für die Zusammenfassung der Blöcke aus Abb. 8.6.* (a) Durch Zusammenfassen von Blöcken werden Knoten einer Baumstruktur gebildet bis ein vollständiger Baum vorliegt. (b) Aus dem rekursiven „Durchwandern" des Baumes kann die Blockreihenfolge abgelesen werden.

	Datencache		Befehlscache	
	Anzahl	*Anteil*	*Anzahl*	*Anteil*
Gesamtanzahl Zugriffe K_{ges}	$26.1 \cdot 10^6$	100%	$1.54 \cdot 10^6$	100%
Anzahl Cache-Hits K_{hits}, k_{hits}	$25.8 \cdot 10^6$	99.17%	$1.49 \cdot 10^6$	96.78%
Anzahl Cache-Misses K_{miss}, k_{miss}	$215 \cdot 10^3$	0.83%	$49.5 \cdot 10^3$	3.22%
transferierte Speicherworte K_{trans}	$1.72 \cdot 10^6$	100%	$396 \cdot 10^3$	100%
davon verwendet K_{load}, k_{load}	$1.62 \cdot 10^6$	94.21%	$342 \cdot 10^3$	86.41%

Tabelle 8.5: *Zugriffsstatistik für Befehls- und Datencache nach erfolgter statischer Blockoptimierung.* Die Tab. zeigt Effizienzkennzahlen in absoluten und prozentuellen Größen getrennt für Befehls- und Datencache, nachdem die Methode der statischen Blockoptimierung angewendet wurde.

die Optimierung stark reduziert. Analog zu den Gln. (8.9) und (8.10) lassen sich die mittleren Energiemengen je Speicherzugriff mit

$$\frac{E_{ges}}{K_{ges}} = 0.9678 \cdot 0.56\text{nJ} + 0.0322 \cdot 271\text{nJ} = 9.27\text{nJ} \tag{8.16}$$

für den Befehlscache und

$$\frac{E_{ges}}{K_{ges}} = 0.9917 \cdot 0.56\text{nJ} + 0.0083 \cdot 271\text{nJ} = 2.80\text{nJ} \tag{8.17}$$

für den Datencache angeben. Durch Vergleich der Zahlenwerte aus den Gln. (8.9), (8.10), (8.16) und (8.17) erhält man bei statischer Blockoptimierung eine Verringerung des Energiebedarfes um 21% für den Befehlscache und um 29% für den Datencache.

8.5.3 Optimierung anhand der Adressraumdistanz

Durch die statische Blockoptimierung werden Cache-Zeilen effizienter genutzt, da häufig nacheinander angeforderte Speicherstellen im Adressraum nebeneinander zu liegen kommen. Völlig unberücksichtigt bleibt jedoch dabei die zeitliche Verteilung der Zugriffe auf die beteiligten Speicherblöcke, d. h. es wird nicht berücksichtigt, ob ein Block im Laufe der Programmausführung in regelmäßigen Abständen wiederkehrend oder nur für kurze Zeit angefordert wird. Für die Ausnutzung eines Cache-Speichers ist aber die zeitliche Lokalität der Zugriffe mitentscheidend, nachdem Cache-Einträge laufend durch neue ersetzt werden müssen.

Der Einfluss der Verdrängungsstrategie auf die Cache-Effizienz

Ein idealer Cache enthält zu einem gegebenen Zeitpunkt all jene Speicherstellen, die bei weiterer Programmausführung wieder Verwendung finden. Nicht mehr referenzierte Speicherstellen können überschrieben werden. Bei Beschränkung der Speicherkapazität kann der ideale Cache nicht mehr alle Speicherstellen enthalten, sondern nur jene, die bei fortgesetzter Programmausführung eine maximale Anzahl von „cache hits“ hervorrufen. D. h. ein idealer Cache verdrängt bei Neuaufnahme eines Eintrages immer jenen Eintrag, welcher für nachfolgende Speicherzugriffe die geringste Bedeutung besitzt, also die wenigsten „cache hits“ generieren würde.

Die Verdrängungsstrategie hat damit zentralen Einfluss auf die Effizienz eines Caches. Nachdem in einem Cache über künftige Zugriffe kein Wissen vorliegt, wird in der Praxis zumeist eine der folgenden drei Strategien angewendet [14]:

- *„Random“:* Der verdrängte Cache-Eintrag wird durch Zufall ausgewählt (Pseudozufallsgenerator)

- *„Least Recently Used“*: Es wird jener Eintrag überschrieben, dessen letzte Verwendung am weitesten zurück liegt
- *„Round Robin“:* Die Verdrängung erfolgt zyklisch, also der Reihe nach innerhalb eines Sets. Dieses Verfahren wird im Intel XScale eingesetzt.

Alle diese Strategien sind nur Näherungen für den idealen Cache. Durch Kenntnis der Adressraumspur aus dem Simulator sind jedoch zu einem gegebenen Zeitpunkt alle nachfolgenden Speicherzugriffe bekannt, sodass man dieses Wissen gezielt einsetzen kann, um bei Verwendung der fest vorgegebenen Verdrängungsstrategie trotzdem jene Cache-Einträge zu ersetzen, die bei nachfolgenden Speicherzugriffen geringste Bedeutung aufweisen.

Man kann aus der Analyse der Adressraumspur und der daraus hervorgehenden Umgestaltung des Adresslayouts eine Cache-Zuteilung erreichen, die trotz nicht idealer Eigenschaften der gegebenen Verdrängungsstrategie das Idealverhalten eines Caches besser nachbildet, als dies bei den oben beschriebenen drei Strategien der Fall ist.

Definition der Adressraumdistanz

Als Adressraumdistanz wird jene Anzahl δ_{xy} von Speicherwortadressen definiert, die der Mikroprozessor bei Programmausführung nach Zugriff auf den Speicherblock B_x bis zum Zugriff auf den Speicherblock B_y insgesamt vom Hauptspeicher anfordert. Mithilfe der Ausführungsreihenfolge J aus Gl. (8.11) und den Blocklängen l_i lässt sich die Adressraumdistanz aus

$$\delta_{xy} = \sum_{k=\{j_{r+1},\ldots,j_{s-1}\}} l_k, \quad x = j_r,\ y = j_s \tag{8.18}$$

berechnen. Gl. (8.18) gilt für beliebiges Auftreten und beliebige Kombinationen von B_x und B_y innerhalb der Ausführungsreihenfolge J.

Die Adressraumdistanz ist ein Maß dafür, wie viele Speicheradressen nach dem Zugriff auf Block B_x in den Cache geladen werden müssen, bis schließlich auf den Block B_y zugegriffen wird. Ist δ_{xy} größer als die Kapazität K_C des Cache-Speichers, deutet dies darauf hin, dass der Block B_x bei Zugriff auf B_y mit hoher Wahrscheinlichkeit nicht mehr im Cache verweilt. Besonders interessant ist daher die Adressraumdistanz δ_{xx} zwischen zwei hintereinanderfolgenden Zugriffen auf denselben Block B_x, da sie als Wahrscheinlichkeit eines „cache hit“ beim zweiten Zugriff auf B_x interpretiert werden kann.

Für eine Optimierung eines gesamten Programmdurchlaufes sind einzelne Adressraumdistanzen δ_{xx} alleine nicht aussagekräftig, da sie nur Momentansituationen beschreiben. Um eine generellere Aussage über die Cachebarkeit eines

Blocks B_x zu erhalten ist eine Mittelwertbildung

$$\Delta_{xx} = \overline{\delta_{xx}} \tag{8.19}$$

über alle möglichen δ_{xx} einer Ausführungsreihenfolge erforderlich. Für eine einfachere Interpretation der mittleren Adressraumdistanz Δ_{xx} ist der Bezug auf das Fassungsvermögen des Cache-Speichers sinnvoll, woraus man die bezogene, mittlere Adressraumdistanz

$$d_{xx} = \frac{\Delta_{xx}}{K_C} \tag{8.20}$$

erhält. Bei Berechnung der bezogenen, mittleren Adressraumdistanz für alle N Speicherblöcke ergibt sich der Vektor

$$D = (d_{11},\, d_{22},\, \ldots,\, d_{NN})^T. \tag{8.21}$$

Methodik der Optimierung anhand der Adressraumdistanz

Die Steigerung der Cache-Effizienz durch Optimierung anhand der Adressraumdistanz baut auf der statischen Blockoptimierung auf und erweitert diese Methode hinsichtlich der Zuteilung von Speicherblöcken zu den gegebenen Cache-Ressourcen.

Die Optimierung erfolgt in drei Schritten. Im ersten Schritt werden einzelne Speicherblöcke durch statische Blockoptimierung zu größeren Blockgruppen zusammengefasst, wobei das sukzessive Zusammenfassen zu einer Gruppe genau dann beendet wird, wenn die Gesamtlänge der Gruppe ein Vielfaches der Länge einer Cache-Zeile beträgt. Das Ergebnis dieses Schrittes ist eine Vielzahl von Blockgruppen, die sich jeweils lückenlos einer ganzzahligen Anzahl von Cache-Zeilen zuteilen lassen.

Im zweiten Schritt wird die bezogene, mittlere Adressraumdistanz für die einzelnen Blockgruppen gemäß Gl. (8.21) berechnet.

Im dritten Schritt erfolgt die Zuteilung der Blockgruppen zu den Sets des Cache-Speichers anhand der Adressraumdistanz, wobei folgende Strategie verwendet wird:

- Blockgruppen mit $d_{xx} \gg 1$ bzw. $d_{xx} \geq d_{max}$, mit einer noch zu bestimmenden festen Schranke d_{max}, sind kaum cachebar und können ohne Bedenken rasch wieder im Cache überschrieben werden. Alle solche Blockgruppen werden daher demselben Cache-Set zugeteilt.
- Blockgruppen mit $d_{xx} \approx 0$ besitzen hohe zeitliche Lokalität und werden vom Mikroprozessor sehr rasch wieder angefordert. Sie sind sehr gut cachebar,

können aber nach ihrer kurzzeitig sehr häufigen Verwendung rasch wieder im Cache überschrieben werden. Aus diesem Grund werden derartige Blockgruppen ebenfalls einem einzigen Cache-Set zugeteilt.

- Im Bereich $0 < d_{xx} < d_{max}$ wird der Adressraumabstand der Blockgruppen und damit die Wahrscheinlichkeit, dass eine Blockgruppe bei ihrem Zugriff schon im Cache überschrieben wurde, immer größer. Unter Berücksichtigung der Round Robin Verdrängungstrategie im Intel XScale ist jedoch der Zeitpunkt des Überschreibens einer Cache-Zeile in einem Cache-Set über die Anzahl der diesem Cache-Set zugeteilten Speicherblöcke steuerbar. Einem Cache-Set sind daher entweder wenige Blockgruppen mit hohen d_{xx} oder viele Blockgruppen mit niedrigem d_{xx} zuzuordnen, um über alle Cache-Sets eine gleichverteilte Wahrscheinlichkeit für „cache hits“ zu erhalten.

Implementierung in Software

Die Umsetzung der Optimierung anhand der Adressraumdistanz in Software baut zunächst auf der statischen Blockoptimierung auf, um Blockgruppen zu bilden, deren Längen ein Vielfaches der Cache-Zeilenlänge betragen. Nach Berechnung der bezogenen, mittleren Adressraumdistanzen gemäß Gl. (8.21) erfolgt die Zuteilung der Blockgruppen zu den Cache-Sets.

Die Cache-Zuteilung erfolgt schrittweise in mehreren Iterationen wobei in jedem Schritt der Parameter d_{max} variiert wird, bis k_{miss} ein Minimum bzw. k_{hit} ein Maximum ausbildet.

Der Startwert für d_{max} wird aus der Wahrscheinlichkeitsfunktion $h(d_{xx})$ für die bezogene, mittlere Adressraumdistanz d_{xx} gewonnen, die in der Abb. 8.9 für den Befehlscache grafisch dargestellt ist. Die Kennlinie aus dieser Abb. ist so zu interpretieren, dass jeder Punkt $(d_\xi, h(d_\xi))$ die Wahrscheinlichkeit $h(d_\xi)$ dafür angibt, mit der $d_{xx} \leq d_\xi$ ist. Nachdem d_{xx} nur diskrete Werte annimmt, stellt $h(d_{xx})$ eine Treppenfunktion dar. Im Sinne einer Ableitung ergeben die Höhen $\Delta h(d_{xx})$ dieser Treppen eine ebenfalls diskrete Verteilungsfunktion (Wahrscheinlichkeitsdichte) für d_{xx}.

Aus der Kennlinie der Abb. 8.9 ist abzulesen, dass $d_{xx} = 0$ mit einer Wahrscheinlichkeit von etwa 16% auftritt. Zwischen $0 < d_{xx} < 1$ steigt die Kennlinie stark an, während sie für $d_{xx} > 1$ etwa linear verläuft. Für ca. 26% aller Blockgruppen gilt $d_{xx} < 1$. Für die restlichen 74% der Blockgruppen liegt bei $d_{xx} > 1$ in etwa eine Gleichverteilung (linearer Anstieg) vor.

Als Startwert für d_{max} wird mindestens die doppelte Adressraumdistanz gewählt, wie sie im Knick der Kennlinie $h(d_{xx})$ vorliegt, also $d_{max} = 1.5$ (siehe Abb. 8.9). Der Kennlinienknick trennt Blockgruppen mit hoher und geringer räumlicher Lokalität voneinander. Für Blockgruppen mit hoher Lokalität arbeitet der Cache ohnehin effizient, weswegen d_{max} anfänglich mit dem doppelten Wert dieser Grenze festgelegt wird.

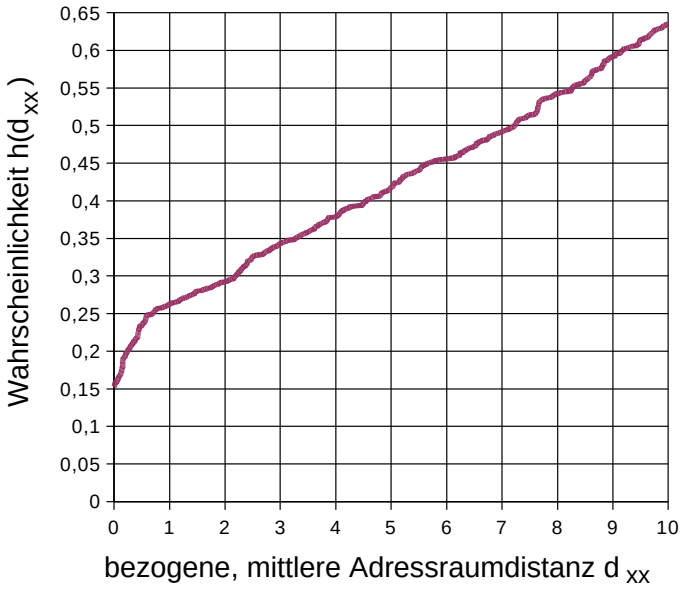


Abbildung 8.9: *Wahrscheinlichkeitsfunktion $h(d_{xx})$ für die bezogene, mittlere Adressraumdistanz d_{xx}.* Die dargestellte Kennlinie gibt an, mit welcher Wahrscheinlichkeit die bezogene, mittlere Adressraumdistanz kleiner oder gleich d_{xx} in einem gewählten Punkt ist.

Mit Kenntnis von d_{max} lässt sich eine Cache-Zuteilung vornehmen. Der Cache des Intel XScale bietet dazu 32 mögliche Sets (Set 0 bis Set 31, Abb. 8.2). Alle Blockgruppen mit $d_{xx} > d_{max}$ werden dem Set 31, jene mit $d_{xx} = 0$ dem Set 0 zugeteilt.

Die restlichen 30 Sets stehen für die Blockgruppen mit $0 < d_{xx} \leq d_{max}$ zur Verfügung. Dazu muss der Bereich $0 < d_{xx} \leq d_{max}$ in 30 Zonen unterteilt werden, wobei der Durchschnittswert $\overline{d_{xx}}$ für die mittlere Adressraumdistanz mal der Anzahl n von Blockgruppen innerhalb dieser Zone für alle Zonen identisch ist:

$$\overline{d_{xx}} \cdot n = \text{const.} \tag{8.22}$$

Unter Berücksichtigung der Verteilungsfunktion $\Delta h(d_{xx})$ ergibt sich damit innerhalb der einzelnen Zonen

$$\sum_{Zone} d_{xx}\Delta h(d_{xx}) = \text{const.} = \frac{1}{30} \sum_{0<d_{xx}\leq d_{max}} d_{xx}\Delta h(d_{xx}), \tag{8.23}$$

wodurch eine Zuteilung der Blockgruppen mit $0 < d_{xx} \leq d_{max}$ zu den Cache-Sets 1 bis 30 einfach möglich ist.

Nach Zuteilung sämtlicher Blockgruppen können die Kennzahlen für die Cache-Effizienz mithilfe der Analysesoftware ausgewertet werden. Anschließend ist das Verfahren durch Variation von d_{max} zu wiederholen, wobei die Effizienzkennzahlen laufend zu berechnen sind.

Für den Befehlscache ergibt sich in Abhängigkeit von d_{max} der in Abb. 8.10 dargestellte Verlauf für k_{hit}. Ein ausgeprägtes Maximum $k_{hit} = 96.85\%$ findet sich bei $d_{max} = 7$.

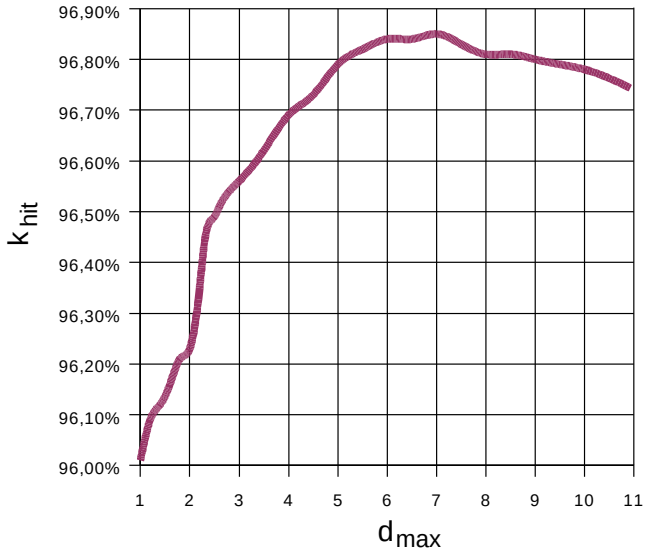


Abbildung 8.10: *Befehlscache-Effizienz k_{hit} in Abhängigkeit vom Optimierungsparameter d_{max}.* Durch Variieren von d_{max} ergibt sich bei $d_{max} = 7$ eine Cache-Effizienz von $k_{hit} = 96.85\%$.

Verbesserung der Cache-Effizienz durch Optimierung anhand der Adressraumdistanz

Nach Anwendung der im vorhergehenden Abschnitt dargestellten Optimierungsmethode ergeben sich die in Tab. 8.6 eingetragenen Effizenzkennzahlen.

Im Vergleich zur statischen Blockoptimierung ergeben sich für k_{miss} weitere Verbesserungen von 0.83% auf 0.81% (relative Verbesserung um 2.4%) im Falle des Datencache und von 3.22% auf 3.15% (relative Verbesserung um 2.2%) im Falle des Befehlscache. Damit konnte die Anzahl der transferierten Speicherworte im Vergleich zur statischen Blockoptimierung weiter gesenkt werden. Die mittleren Energiemengen je Speicherzugriff ergeben

$$\frac{E_{ges}}{K_{ges}} = 0.9685 \cdot 0.56\text{nJ} + 0.0315 \cdot 271\text{nJ} = 9.08\text{nJ} \tag{8.24}$$

für den Befehlscache und

	Datencache		Befehlscache	
	Anzahl	*Anteil*	*Anzahl*	*Anteil*
Gesamtanzahl Zugriffe K_{ges}	$26.1 \cdot 10^6$	100%	$1.54 \cdot 10^6$	100%
Anzahl Cache-Hits K_{hits}, k_{hits}	$25.8 \cdot 10^6$	99.19%	$1.49 \cdot 10^6$	96.85%
Anzahl Cache-Misses K_{miss}, k_{miss}	$211 \cdot 10^3$	0.81%	$48.5 \cdot 10^3$	3.15%
transferierte Speicherworte K_{trans}	$1.69 \cdot 10^6$	100%	$388 \cdot 10^3$	100%
davon verwendet K_{load}, k_{load}	$1.61 \cdot 10^6$	95.2%	$345 \cdot 10^3$	89.08%

Tabelle 8.6: *Zugriffsstatistik für Befehls- und Datencache nach erfolgter Optimierung anhand der Adressraumdistanz.* Die Tab. zeigt Effizienzkennzahlen in absoluten und prozentuellen Größen getrennt für Befehls- und Datencache, nachdem die Methode der Optimierung anhand der Adressraumdistanz angewendet wurde.

$$\frac{E_{ges}}{K_{ges}} = 0.9919 \cdot 0.56\text{nJ} + 0.0081 \cdot 271\text{nJ} = 2.75\text{nJ} \tag{8.25}$$

für den Datencache. Der Energiebedarf verringert sich dadurch um weitere 2% für den Befehlscache und um 1.8% für den Datencache gegenüber der statischen Blockoptimierung.

8.5.4 Elimination von Einzelwortzugriffen

Mit den Ergebnissen aus Tab. 8.1 ergibt sich eine wichtige Konsequez für Speicherzugriffe, die in einer Cache-Zeile ein einziges Speicherwort auch nur ein einziges mal verwenden. Aus den Abbn. 8.3 und 8.4 ist abzulesen, dass derartige Zugriffe vor allem beim Datencache mit der Häufigkeit $p = 0.38 \cdot 0.67 = 0.25$, also in 25% der Fälle auftreten.

Solche Einzelwortzugriffe sind energetisch ungünstig, weil sie den Transfer einer kompletten Cache-Zeile verursachen. Wie in Gl. (8.6) festgestellt, beträgt der Mehraufwand M_{line} für das Laden einer Cache-Zeile 37% gegenüber dem Laden eines einzelnen Datenwortes aus dem Hauptspeicher.

Durch die Analyse der Adressraumspur können Einzelwortzugriffe identifiziert werden. Falls es möglich ist, diese einzelnen Worte zu isolieren (dies ist beispielsweise für einzelne Elemente eines größeren Datenfeldes nicht möglich), können sie in einen anderen Adressbereich verschoben werden, den der Mikroprozessor nicht

durch den Cache abgedeckt. In der virtuellen Speicherverwaltung des Intel XScale sind derartige Speicherbereiche mit Unterstützung des Betriebssystems durch einfaches Setzen eines Flag-Bits im Deskriptor der entsprechenden Speicherseite einzurichten [97]. Der Mehraufwand für das Laden kompletter Cache-Zeilen entfällt somit.

Beim Identifizieren von Einzelwortzugriffen ist darauf zu achten, dass sich bei mehrmaligem Zugriff auf ein einzelnes Datenwort (mindestens ein "cache hit") das Laden der Cache-Zeile bereits rentiert. Es kommen deshalb nur jene Speicherworte für eine Optimierung in Frage, die vom Mikroprozessor tatsächlich nur ein einziges mal angefordert werden.

Im Vergleich zur statischen Blockoptimierung bzw. zur Optimierung anhand der Adressraumdistanz ist eine softwaretechnische Implementierung für die Elimination von Einzelwortzugriffen relativ einfach zu bewerkstelligen, indem die betroffenen Speicherstellen in einem speziell dafür vorgesehenen Speichersegment abgelegt werden. Bei Analyse der Cache-Effizienzkennzahlen finden Zugriffe auf dieses Segment keine Berücksichtigung, sondern werden nur für die Auswertung des Energiebedarfs mitgezählt. In der Tab. 8.7 sind die Ergebnisse der Effizienzanalyse eingetragen.

	Datencache		Befehlscache	
	Anzahl	*Anteil*	*Anzahl*	*Anteil*
Gesamtanzahl Zugriffe K_{ges}	$26.0 \cdot 10^6$	100%	$1.54 \cdot 10^6$	100%
Anzahl Cache-Hits K_{hits}, k_{hits}	$25.8 \cdot 10^6$	99.29%	$1.49 \cdot 10^6$	96.97%
Anzahl Cache-Misses K_{miss}, k_{miss}	$185 \cdot 10^3$	0.71%	$46.5 \cdot 10^3$	3.03%
transferierte Speicherworte K_{trans}	$1.48 \cdot 10^6$	100%	$372 \cdot 10^3$	100%
davon verwendet K_{load}, k_{load}	$1.43 \cdot 10^6$	96.1%	$336 \cdot 10^3$	90.1%
Speicherzugriffe ohne Cache K_{mem}	25696		1538	

Tabelle 8.7: *Zugriffsstatistik für Befehls- und Datencache nach erfolgter Optimierung anhand der Adressdistanz und Elimination von Einzelwortzugriffen.* Die Tab. zeigt Effizienzkennzahlen in absoluten und prozentuellen Größen getrennt für Befehls- und Datencache, nachdem zunächst die Methode der Optimierung anhand der Adressdistanz angewendet und anschließend Einzelwortzugriffe mit der Anzahl K_{mem} eliminiert wurden.

Die Optimierungsergebnisse zeigen im Vergleich mit der Tab. 8.6 eine weitere Reduktion von k_{miss} auf 0.71% für den Datencache (relative Verbesserung um 12%) und auf 3.03% für den Befehlscache (relative Verbesserung um 3.8%). Durch das Eliminieren der Einzelwortzugriffe sinkt die Anzahl der Zugriffe auf den Cache um K_{mem}, da diese nun direkt auf den Hauptspeicher erfolgen. Bei der Berechnung der mittleren Energiemengen müssen diese Zugriffe mit der Energie $K_{mem} \cdot E_s$ berücksichtigt werden. Die benötigten mittleren Energiemengen betragen

$$\begin{aligned} \frac{E_{ges}}{K_{ges}} &= k_{hit}E_{hit} + k_{miss}E_{miss} + \frac{K_{mem}}{K_{ges}}E_s \\ &= 0.9697 \cdot 0.56\mathbf{nJ} + 0.0303 \cdot 271\mathbf{nJ} + \frac{1538}{1.54 \cdot 10^6}198\mathbf{nJ} = 8.95\mathbf{nJ} \end{aligned} \tag{8.26}$$

für den Befehlscache und

$$\frac{E_{ges}}{K_{ges}} = 0.9929 \cdot 0.56\mathbf{nJ} + 0.0071 \cdot 271\mathbf{nJ} + \frac{25696}{26 \cdot 10^6}198\mathbf{nJ} = 2.67\mathbf{nJ} \tag{8.27}$$

für den Datencache. Das entspricht einer prozentuellen Verbesserung gegenüber der Optimierung anhand der Adressraumdistanz um 1.4% für den Befehlscache und 2.9% für den Datencache.

8.6 Ergebnis der energieoptimierten Cache-Zuteilung

Durch Anwendung der im vorangehenden Abschnitt 8.5 dargestellten Methoden lässt sich der Energieaufwand für die während einer Programmabarbeitung durchgeführten Speicherzugriffe deutlich reduzieren. Aus dem Vergleich der experimentellen Ergebnisse für den nicht optimierten Fall in Tab. 8.3 mit den Ergebnissen nach Anwendung der Optimierungsmethoden in Tab. 8.7 ist eine Quantifizierung des gesamten Optimierungserfolges möglich. Die gesamte Energiemenge E_1 für sämtliche Speicherzugriffe beträgt im nicht optimierten Fall

$$\begin{aligned} E_1 \quad = \quad & (K_{hit,I} + K_{hit,D})E_{hit} + \\ & (K_{miss,I} + K_{miss,D})E_{miss} = 122\mathbf{mJ}, \end{aligned} \tag{8.28}$$

wobei die Indizes I und D die jeweiligen Kenngrößen für den Befehlscache („instruction cache“) bzw. Datencache bezeichnen. Nach erfolgter Optimierung ergibt sich die gesamte Energiemenge E_2 zu

$$\begin{aligned} E_2 &= (K_{hit,I} + K_{hit,D})E_{hit} + \\ &\quad (K_{miss,I} + K_{miss,D})E_{miss} + \\ &\quad (K_{mem,I} + K_{mem,D})E_s = 83.4\text{mJ}. \end{aligned} \tag{8.29}$$

Der gesamte prozentuale Optimierungserfolg beträgt damit:

$$\frac{E_1 - E_2}{E_1} = 31.6\%. \tag{8.30}$$

Der geringere Energiebedarf ist auch an der Statistik für Cache-Zugriffe verankert (Abb. 8.11). Im Vergleich zum nicht optimierten Fall (Abb. 8.3) ist der Anteil vollständig ausgenutzter Cache-Zeilen deutlich gestiegen. Mehr als 70% aller Cache-Zeilen werden nun durch den Mikroprozessor in vollem Umfang verwendet. Der Anteil an Einzelwortzugriffen ist auf weniger als 2% abgesunken.

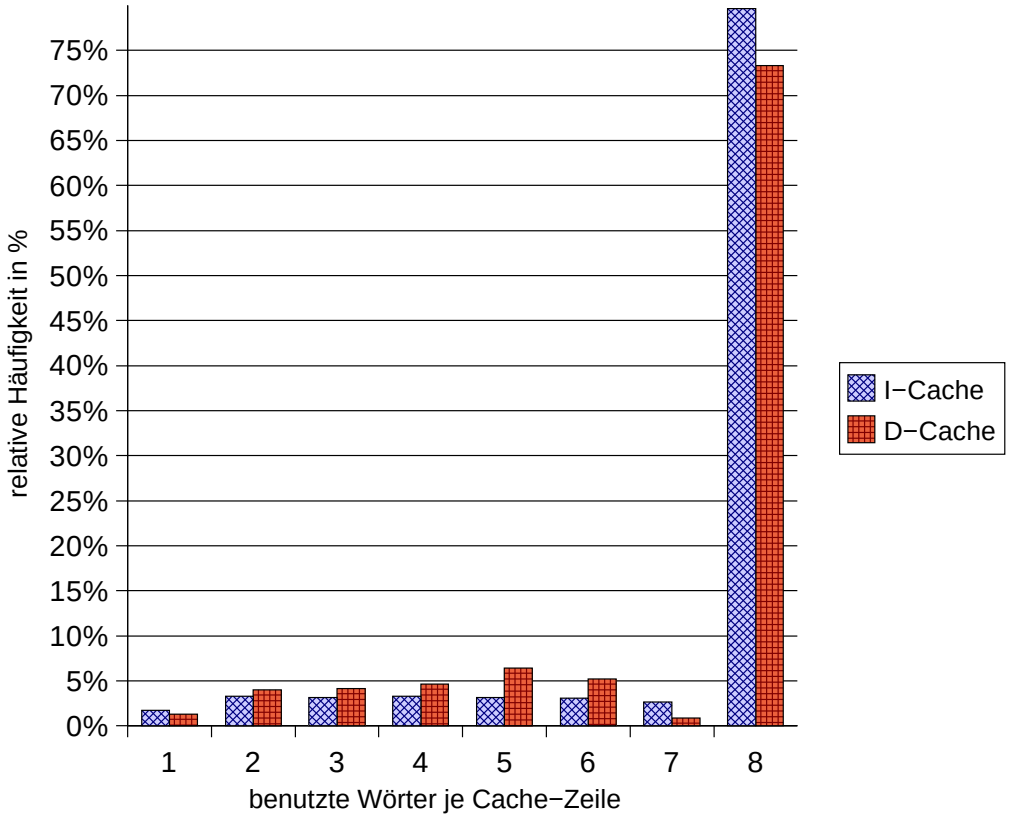


Abbildung 8.11: *Mittlere Ausnutzung von Cache-Einträgen nach Optimierung.* Sowohl für den Befehls- als auch für Datencache werden in über 70% der Fälle alle acht Speicherworte der Cache-Zeile ausgenutzt. Im Vergleich mit der ursprünglichen Abb. 8.3 ist eine nahezu vollständige Elimination von Einzelwortzugriffen und eine deutliche Verringerung der Anzahl unvollständig ausgenutzter Cache-Zeilen erkennbar.

Kapitel 9

Zusammenfassung

Die vorliegende Dissertationsarbeit setzt sich mit dem Entwurf von hochperformanten Low Power Mikroprozessorsystemen auseinander. In der Aufgabenstellung wird ein Mikroprozessorsystem gefordert, das hinsichtlich Leistungsverbrauch, Rechengeschwindigkeit, geometrische Abmessungen und Speicherausstattung bestehende Lösungen deutlich übertrifft.

Die größte Herausforderung besteht in der gleichzeitigen Erfüllung der geforderten Kennzahlen für Leistungsverbrauch und Performance. Um dieser Herausforderung gerecht zu werden, wurden neben einer konsequenten Umsetzung bekannter Methoden des "designs for low power" neue Verfahren und Methoden entwickelt, um den Zielkonflikt zwischen Rechenleistung und Stromaufnahme zu beherrschen.

Nach der Darstellung des Stands der Technik wurde im Kapitel 5 eine Methode vorgestellt, die dem Entwurf von hochperformanten Low Power Mikroprozessorsystemen zugrundezulegen ist. Es handelt sich hierbei um ein ebenenübergreifendes Designparadigma, das den Einsatz von verlustleistungsoptimalen Techniken von der Transistor-Ebene bis hinauf zur Software-Ebene vorsieht.

Unter Berücksichtigung des Designparadigmas aus Kapitel 5 identifiziert der Mittelteil dieser Arbeit mithilfe gezielter Marktrecherchen modernste mikroelektronische Bauelemente, die aufgrund ihrer speziellen Eigenschaften eine hardwaretechnische Umsetzung der gestellten Aufgabe ermöglichen. Das Ergebnis ist ein heterogenes System aus Spezialkomponenten, die durch einen programmierbaren Logikbaustein (FPGA) miteinander verschalten sind. Der FPGA übernimmt neben den Funktionen SDRAM-Controller, Flash-Controller, UART und Interrupt-Controller auch die wichtige Aufgabe des umfassenden Low Power Managements auf der Mikroprozessorplatine. Dadurch werden sämtliche Teilfunktionen des Mikroprozessorsystems nur bei tatsächlichem Bedarf zur Verfügung gestellt, während sie in der restlichen Zeit deaktiviert sind bzw. in einem energiesparenden Zustand verweilen.

Das Kapitel 7 stellt eine vollständige Software-Entwicklungsumgebung für das Mikroprozessorsystem vor. Als Grundlage dienen dabei das Betriebsystem Linux und

die GNU-Entwicklungswerkzeuge. Hauptaufgabe ist einerseits die Anpassung der frei verfügbaren Softwarepakete an die Gegebenheiten des Mikroprozessorsystems und insbesondere die Portierung des Linux-Kernels.

Im Rahmen der Tätigkeiten zur Erstellung einer Entwicklungsumgebung wurde ein neuer Hardware-Simulator für das Intel XScale-basierte Mikroprozessormodul entwickelt. Der Simulator ist nicht nur in der Lage, Programmcode in einem XScale-Prozessormodell auszuführen, sondern bildet auch zusätzlichen Komponenten wie MMU, Cache, Speicherbausteine und UARTs nach. Dadurch wird eine vollständige Simulationsumgebung geschaffen, die eine Softwareentwicklung ohne Vorliegen der realen Hardware erlaubt. Selbst die Ausführung von umfassenden Betriebssystemen ist möglich, wie am Beispiel von Linux in Kapitel 7, Abb. 7.5 dargestellt wurde.

Ein Hardware-Simulator besitzt den enormen Vorteil, dass die Entwicklungsprozesse für Hard- und Software parallel ablaufen können. Darüber hinaus sind innerhalb eines Simulationsmodells sämtliche Vorgänge bei Ausführung eines Programms im Detail beobachtbar. Real nicht messbare Größen können so messbar gemacht werden (z.B. detaillierte Statistiken über Cache-Allokation).

Im letzten Teil der vorliegenden Arbeit werden neue Methoden zur energieoptimalen Cache-Zuteilung vorgestellt. Grundprinzip ist die Reduktion der für Speicherzugriffe des Mikroprozessors benötigten Energiemenge durch bessere Ausnutzung der vorhandenen Cache-Speicher.

Die vorgeschlagenen Verfahren analysieren die aus dem Hardwaresimulator bei Ausführung von Testprogrammen gewonnene Adressraumspur und steigern durch Umordnen des Adresslayouts die räumliche und zeitliche Lokalität der ausgeführten Programmsequenzen. Damit ist es gelungen, bei gleichzeitigem Performancegewinn eine Reduktion der für Speicherzugriffe benötigten Energiemenge um etwa 32% herbeizuführen.

Je nach Anteil des Hauptspeichers am Gesamtleistungsverbrauch kann durch die hier vorgeschlagenen Methoden die Stromaufnahme eines Mikroprozessorsystems deutlich verringert werden.

Kapitel 10

Ausblick

Das im Rahmen dieser Dissertation entwickelte Mikroprozessormodul stellt einen funktionstauglichen Prototypen für ein hochperformantes Low Power Mikroprozessorsystem dar. Damit werden neue, bisher nicht realisierbare Anwendungen, z. B. im Bereich von mobilen, tragbaren Computersystemen möglich.

Um für das neue Mikroprozessormodul eine weitere Reduktion der mittleren Stromaufnahme zu erzielen, ist die Umsetzung von zusätzlichen Maßnahmen möglich. So kann beispielsweise durch Umgestaltung von Programmquellcode die Effizienz von Algorithmen erhöht und der Berechnungs- bzw. der Energieaufwand dadurch reduziert werden.

Für die Weiterentwicklung des vorliegenen Prototypen eines XScale-Mikroprozessormoduls zu einem kommerziell nutzbaren Produkt sind noch gezielte Maßnahmen zur Senkung der Herstellkosten zu ergreifen.

Durch den Einsatz des Betriebssystems Linux wurde ein innovativer Weg beschritten. Laut Auskunft eines Herstellers von mikroelektronischen Komponenten [171] wird zur Zeit in 50% der neu entwickelten Embedded Systeme Linux als Betriebssystem eingesetzt.

10.1 Weitere Möglichkeiten zur Verringerung der Verlustleistung

Das Spektrum der Möglichkeiten zur Verringerung der Verlustleistung erstreckt sich über das gesamte Ebenenmodell aus Abb. 5.1, vom mikroelektronischen Bauelement bis zur Software.

10.1.1 Einsatz neuer Bauelemente

Besonders naheliegend ist der Einsatz von neu in den Markt eingeführten Bauteilen, die besonders für den Einsatz in batteriebetriebenen Geräten konzipiert sind. Der riesige Markt für mobile Telefone und Computersysteme fordert die laufende Entwicklung und Weiterentwicklung solcher Bauteile heraus.

Schlüsseltechnologie und Antriebsmotor für die Entwicklung neuer Low Power Komponenten ist die CMOS-Technologie für kleinste Transistor-Strukturen im Bereich unter 0.13μm. Zur Zeit wird an der prozesstechnischen Realisierung von 90nm-Fertigungslinien gearbeitet [177, 178].

Kleinste Geometrien gewährleisten nicht nur kurze Signallaufzeiten und geringe Gatterkapazitäten sondern ermöglichen durch die hohe Integrationsdichte den verstärkten Einsatz von parallelen Schaltungsstrukturen als Ersatz für höher getaktete serielle Schaltwerke um eine Verbesserung der Verlustleistung herbeizuführen (vgl. Abb. 3.7). Die Verfügbarkeit von Sub-Mikron CMOS-Technologie stellt somit einen zusätzlichen Freiheitsgrad beim Entwurf von Low Power Schaltkreisen dar [180].

Ein mögliches Beispiel für die Nutzung neuer Bauelemente ist die PXA-Prozessorfamilie [179] der Fa. Intel, die spezielle für den Einsatz in PDAs entwickelt wurde. Zusätzlich zu dem hier verwendeten XScale 80200 integriert der PXA-250 neben einem umfassenden Memorycontroller eine Reihe von I/O-Komponenten (UART, LCD, USB) und erreicht damit eine noch höhere Integrationsdichte.

10.1.2 Einsatz eines ASIC

Der im neuen Mikroprozessormodul eingesetzte FPGA ist bei hohen Stückzahlen (vgl. Abb. 6.2) durch einen CBIC oder Full Customized ASIC zu ersetzen. Der durch die Programmierbarkeit eines FPGA verursachte Mehraufwand an Schaltkreislogik könnte dadurch eliminiert werden. Auch die für den ASIC benötigte Siliziumfläche verringert sich drastisch. Man kann davon ausgehen, dass die Verlustleistung eines Full Customized ASIC etwa um 90% geringer ist als ein funktional identischer FPGA [181].

10.1.3 Nutzung des Mini-Datencaches

Der Intel XScale Mikroprozessor besitzt neben Befehls- und Datencache noch einen weiteren Mini-Datencache , der in erster Linie für die Verarbeitung von Datenströmen vorgesehen ist [97]. Dieser Mini-Datencache hat ein Fassungsvermögen von 2kbyte und ist zweifach assoziativ. In den Betrachtungen zur energieoptimalen Cachezuteilung des Kapitel 8 wurde dieser Cache nicht berücksichtigt, da er vom Betriebssystem Linux derzeit nicht unterstützt wird.

Der Mini-Datencache stellt jedenfalls eine zusätzliche - 2kbyte große - Cacheressource dar, die bei entsprechender Nutzung eine weitere Steigerung der Effizienz von Speicherzugriffen hervorruft. Dazu sind die Methoden der energieoptimalen Cachezuteilung zu erweitern und der Mini-Datencache durch Eingriff in das Betriebssystem softwaretechnisch zu aktivieren.

10.1.4 Optimierung von Quellcode

Das größte Potential in der Verringerung des Energiebedarfes für eine konkrete Aufgabe ist auf der Software-Ebene zu finden. In [182] wird ein Verbesserungsfaktor von 10-100 abgeschätzt, also eine Reduktion des Energieaufwandes auf 1-10% gegenüber dem nicht optimierten Fall. Die Verbesserung wird allein durch geschicktes Umstellen der zugrundeliegenden Algorithmen erzielt.

Tatsächlich lehren auch die Erfahrungen aus der Praxis, dass die ersten Versionen der softwaretechnischen Realisierung einer Aufgabe meist sehr ineffizient und umständlich programmiert sind. Erst bei iterativer Verfeinerung und Verbesserung der Algorithmen und Abläufe wird eine zufriedenstellende Effizienz erreicht. Das Verbesserungspotential ist zumeist enorm hoch. So konnte auch die Ausführungszeit der im Rahmen von Kapitel 8 implementierten Optimierungsverfahren gegenüber anfänglichen Versionen von mehreren Stunden auf wenige Sekunden verkürzt werden.

Für die Optimierung von Quellcode gibt es keine einheitlichen bzw. allgemein gültigen Methoden. Jeder anwendungsspezifische Fall erfordert gesonderte Betrachtung. Mit der Optimierung von Quellcode durch Transformation der Speicherorganisation beschäftigen sich eine Reihe von wissenschaftlichen Arbeiten. In [183, 184, 185, 186] wird z. B. besonders auf die Thematik von Speicherplatz und Energieeffizienz in Multimediasystemen eingegangen. Gerade in diesem kostensensitiven Bereich der Consumer-Elektronik hat das Einsparen von Speicherplatz und Energie - ähnlich wie in der Automobilindustrie - wegen der hohen Stückzahlen besonders große wirtschaftliche Bedeutung.

10.1.5 Asynchrone Schaltungstechnik

Heutige Mikroprozessoren sind wegen der besseren Beherrschbarkeit und Übersichtlichkeit praktisch ausschließlich in synchroner Schaltungstechnik ausgeführt. Ein wesentlicher Nachteil daraus ist, dass allein für die über die gesamte Chipfläche verteilte Systemtaktversorgung ein erheblicher Teil der gesamten Verlustleistung entsteht. Für den Alpha-Prozessor der Fa. Digital waren das beispielsweise 50% der Prozessorverlustleistung.

Asynchrone Schaltungslogik funktioniert grundsätzlich ohne Taktsignal und vermeidet so diesen Nachteil. Damit lassen sich wesentlich energieeffizientere Schal-

tungen aufbauen, was allerdings zum Preis einer komplexeren Schaltungstechnik mit Handshakesignalen erkauft werden muss.

Allein die Tatsache, dass es derzeit keinen kommerziell erhältlichen, vollständig asynchronen Mikroprozessor gibt, zeigt, dass diese Technik noch in den Kinderschuhen steckt. Der in Abb. 1.2 dargestellte Konflikt zwischen Rechengeschwindigkeit und Verlustleistung macht jedoch die asynchrone Logik wieder für den großtechnischen Einsatz interessanter. Ein typisches Beispiel dafür ist „Amulet“ [187], die asynchrone Variante der ARM-Mikroprozessorarchitektur, deren Wurzeln in das Jahr 1990 zurückreichen, aber erst jetzt in der Version „Amulet3“ ein kommerziell tragbares Konzept aufweist [188].

Asynchrone Mikroprozessoren werden in den nächsten Jahren zunehmend in der Nische der „Super Low Power“ Systeme Fuß fassen können. Im Bereich der Arbeitsplatzrechner und Server wird sich die bewährte synchrone Schaltungstechnik noch lange behaupten.

10.2 Anwendungen

Das neue XScale-Mikroprozessormodul ermöglicht durch seine Kombination aus hoher Rechenleistung, geringer Stromaufnahme und kleiner Bauform neue Anwendungen, die bisher nicht realisierbar waren:

- ***Drahtloses Hochgeschwindigkeitskamerasystem:*** Für die Beobachtung schneller Prozessabläufe in der Fertigungstechnik wird eine Kamera in den Arbeitsraum der zu beobachtenden Maschine eingebracht. Ohne Steckkontakte und Verbindungskabel kann diese Montage sehr einfach und rasch erfolgen. Die Hochgeschwindigkeitskamera zeichnet Bildfolgen auf, die mit hoher Verarbeitungsgeschwindigkeit des XScale-Mikroprozessors komprimiert im Arbeitsspeicher abgelegt werden. Durch die hohe Kompressionsrate (bis etwa 1:200 [173, 174]), die für Bildfolgeaufzeichnungen möglich ist, erhält man ein Vielfaches der Aufnahmedauer herkömmlicher Hochgeschwindigkeitskameras. Die Ansteuerung der Kamera erfolgt sowie die Offline-Übertragung von Bildinformation über eine drahtlose Funkschnittstelle wie z. B. Bluetooth [172].

- ***Mobiler Spracherkenner:*** Aufgrund der beschränkten Eingabemöglichkeiten tragbarer Computersysteme (z. B. numerische Tastatur am Handy, winzige Tastaturen, Handschrifterkennung eines PDA) stellen sprachgesteuerte Benutzerschnittstellen eine schnelle Möglichkeit für die Eingabe von Texten dar (z. B. eMail, SMS). Die Erkennung einzelner gesprochener Kommandos ist bereits Stand der Technik im Automobilbau [175]. Für kontinuierliche Spracherkennung sind zur Erzielung hoher Erkennungsgüten sehr hohe

Rechenleistungen erforderlich, die bisher nur von Arbeitsplatzrechnern aufgebracht werden konnten. Die hohe Verarbeitungsgeschwindigkeit des neuen Mikroprozessormoduls ermöglicht die Erschließung dieses Anwendungsbereiches auch für tragbare Computersysteme.

- ***Steuerungen mit integrierter Bildverarbeitung:*** In der Automatisierungstechnik wurden bislang Bildverarbeitungssysteme getrennt von Steuerungsystemen (SPS) betrieben. Der Grund dafür ist, dass einerseits Steuerungen den Rechengeschwindigkeitsanforderungen der Bildverarbeitung nicht gewachsen sind und andererseits Bildverarbeitungssysteme nicht im Formfaktor der Steuerungen herstellbar waren. Mit der Entwicklung des hochperformanten und gleichzeitig scheckkartengroßen Mikroprozessormoduls ist ein Zusammenführen von Steuerungstechnik und Bildverarbeitung zu kompakten opto-elektronischen Sensor/Aktorsystemen möglich.
- ***Visuelle Prozessüberwachung:*** Die automatisierte visuelle Überwachung von Herstellprozessen und industriellen Anlagen ist eine wichtige Herausforderung an die industrielle Bildverarbeitung, da in diesem Anwendungsbereich enormes Marktpotential vorhanden ist [176]. Gerade im Überwachungsbereich spielen kleine Systemgeometrien, hohe Rechenleistung und einfache Replizierbarkeit eine wesentliche Rolle. Mithilfe des neuen XScale-Mikroprozessormoduls lassen sich integrierte visuelle Sensorsysteme konzipieren, die sich aufgrund ihrer kleinen Abmessungen und der hohen Rechenleistung optimal für den Einsatz in der Überwachungstechnik eignen. So wäre beispielsweise die Prozessraumüberwachung eines Roboter-Arbeitsbereiches für den Personenschutz durch einen kompakten, intelligenten Sensor realisierbar, der durch ein einfaches digitales Ausgangssignal eine Notabschaltung durchführen könnte.

10.3 Die Zukunft von Embedded Linux

Der Einsatz von Linux im Bereich der Embedded Systeme hat in den vergangenen Jahren einen enormen Aufschwung erlebt und wird die Zukunft der Embedded Systeme wesentlich mitgestalten.

Das einstige Betriebssystem für Hacker ist salonfähig geworden

Anfänglich fast ausschließlich als Server-Betriebssystem eingesetzt, hat sich Linux im Laufe der Zeit durch die zunehmende Modularisierung und Portierung auf Prozessorfamilien des Embedded-Bereiches auch in Kleinsystemen etabliert. Die zunehmende Leistungssteigerung der Mikroprozessoren sowie das steigende Fassungsvermögen eingesetzter Speicherbausteine trugen wesentlich zu dieser Entwicklung bei.

Ein weiterer Grund für die Beliebtheit von Linux ist seine weite Verbreitung als kostenloses Betriebssystem für Arbeitsplatz-PCs. Dieser Bekanntheitsgrad ermutigt viele Hersteller dazu, Linux auch in Embedded-Komponenten und -Geräten einzusetzen. Nach Aussage des Elektronikherstellers Kontron [171] werden zur Zeit etwa 50% der Neuentwicklungen im Embedded-Bereich mit Linux durchgeführt.

Schlagkräftigstes Argument für den Einsatz von Linux ist seine völlige Kostenfreiheit. Im Gegensatz zu anderen Betriebssystemen fallen weder Anschaffungskosten während der Entwicklung noch Lizenzgebühren pro verkauftem Gerät an.

Auch das immense Reservoir an frei verfügbarer Software, die unter dem Betriebssystem Linux läuft, ist ein mitentscheidener Faktor. Damit ergeben sich einerseits notwendige Flexibilitäten für Rapid Prototyping und lassen sich andererseits wertvolle Zeit und Entwicklungskosten einsparen.

Ein weiterer wesentlicher Faktor für kurze Entwicklungszeiten ist, dass Linux sowohl am Zielsystem als auch am Entwicklungsrechner gleichermaßen betrieben werden kann. Anwendungen können bereits am Arbeitsplatzrechner entwickelt und getestet werden, bevor die Integration am Zielsystem erfolgt.

Künftige Einsatzgebiete von Embedded Linux

Linux ist ein mächtiges Betriebssystem mit umfangreichen Möglichkeiten. Allein daraus lässt sich ableiten, dass sich der Einsatz von Linux für Kleinstsysteme (etwa: Mikrocontroller mit 8kbyte RAM und 32kbyte Flash) nicht lohnt bzw. gänzlich unmöglich ist. Erst auf Computersystemen mit einer Speicherausstattung von mehr als 2Mbyte Speicher ist der Betrieb von Linux sinnvoll möglich [144].

Linux wird daher bevorzugt in komplexeren Anwendungsfällen beispielsweise in Geräten mit hohem Bedarf an Kommunikationsmöglichkeiten mit verschiedensten Schnittstellen eingesetzt werden.

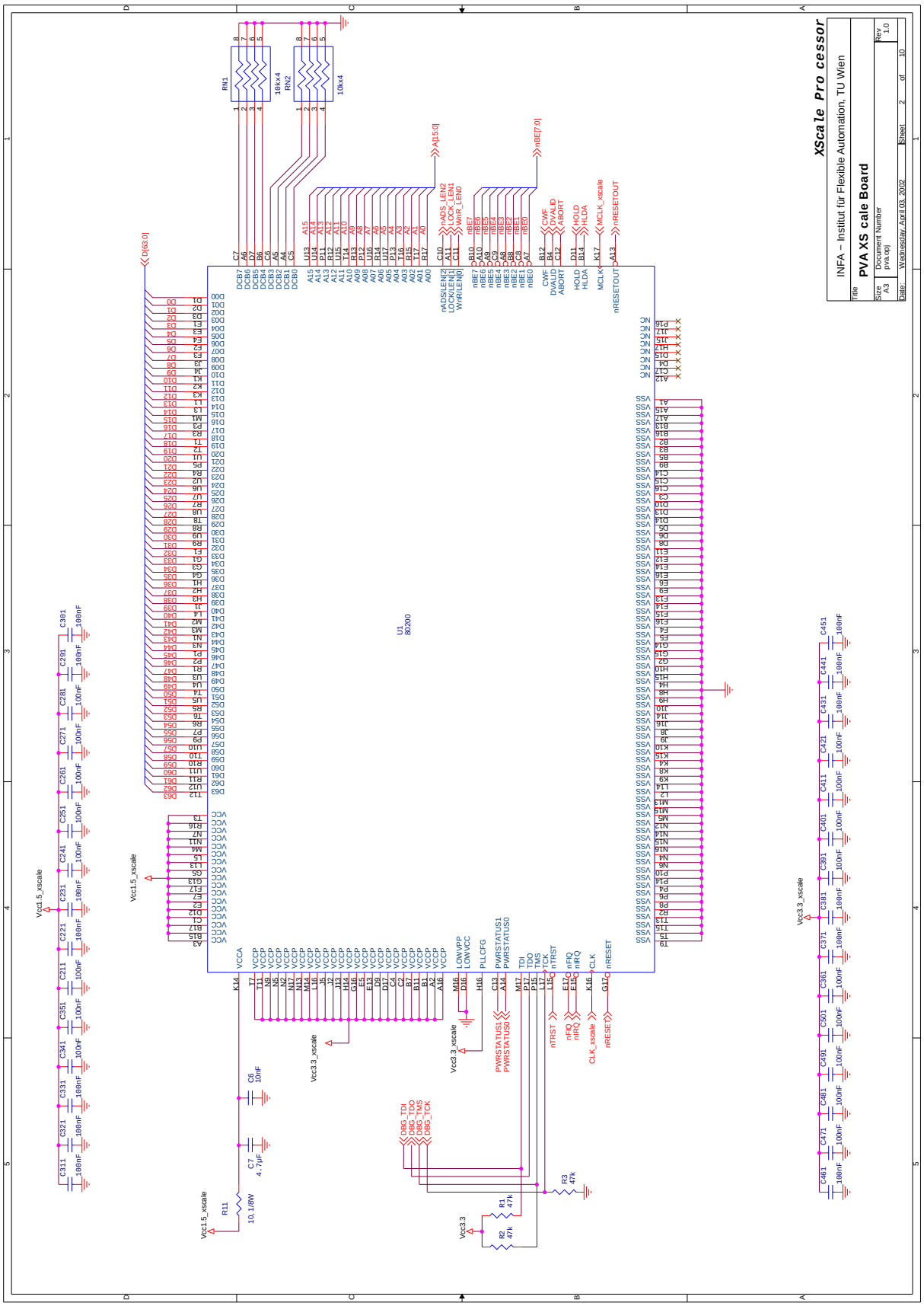
Typische Vertreter solcher Anwendungen sind Multimediasysteme (z. B. integrierte Lenker-Informationssysteme im Automobil), Kommunikationssysteme (z. B. VoIP), internetfähige Embedded Systeme (z. B. „Internet-Appliances“, „Set-Top Boxes“), tragbare Computer (z. B. PDA) und vielseitige Dienstzugangssysteme (z. B. „Service Access Points“ für WLAN und Bluetooth).

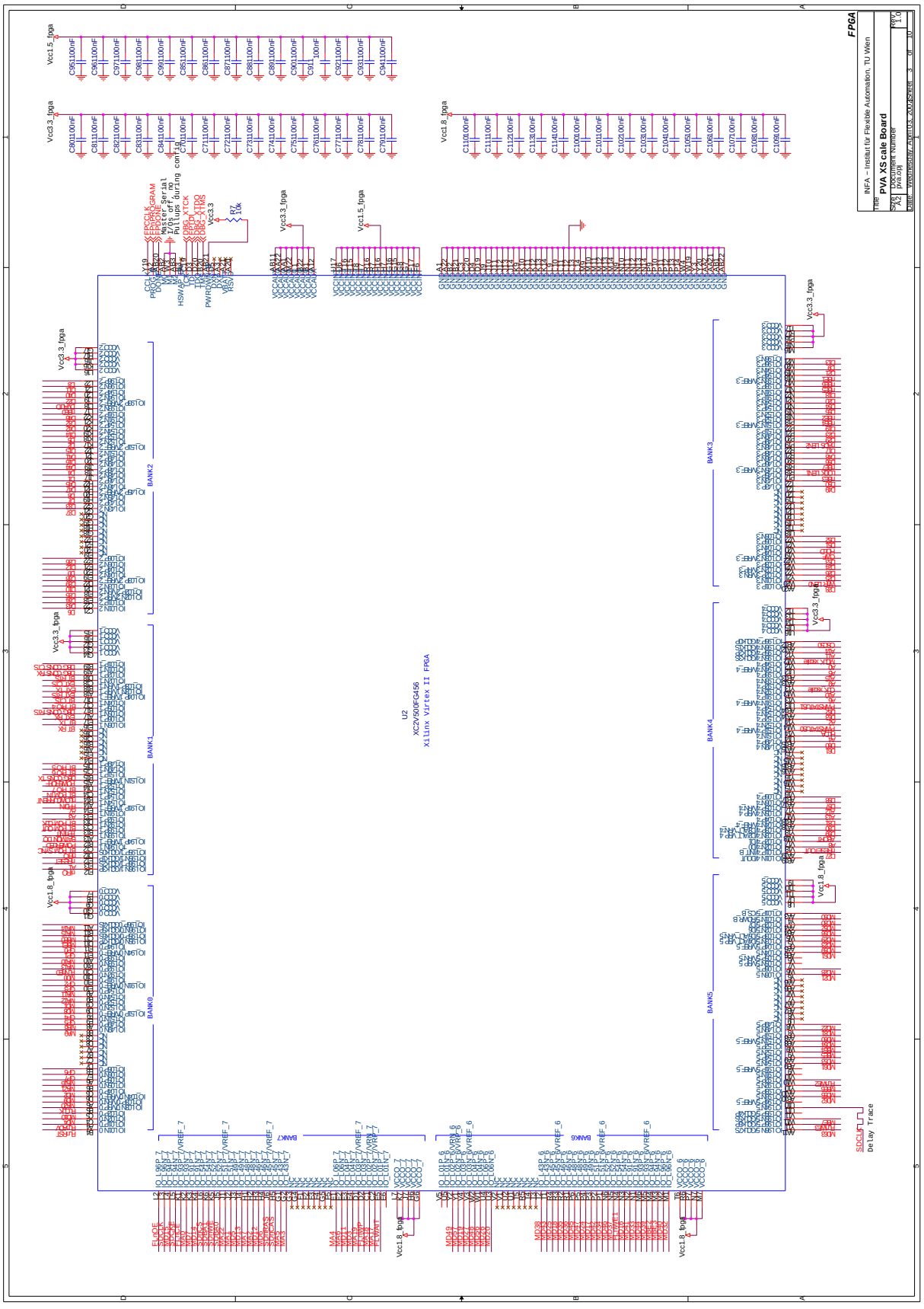
Aber auch im industriellen Umfeld kann Embedded Linux immer besser Fuß fassen. Die Erweiterung von Linux zu einem echtzeitfähigen Betriebsystem (RT-Linux, RTAI, [189, 190]) hat diese Entwicklung entscheidend beschleunigt. So ist mittlerweile Linux sowohl in komplexen Sensorsystemen (z. B. „Smart Cameras“) als auch in SPS-Steuerungen zu finden.

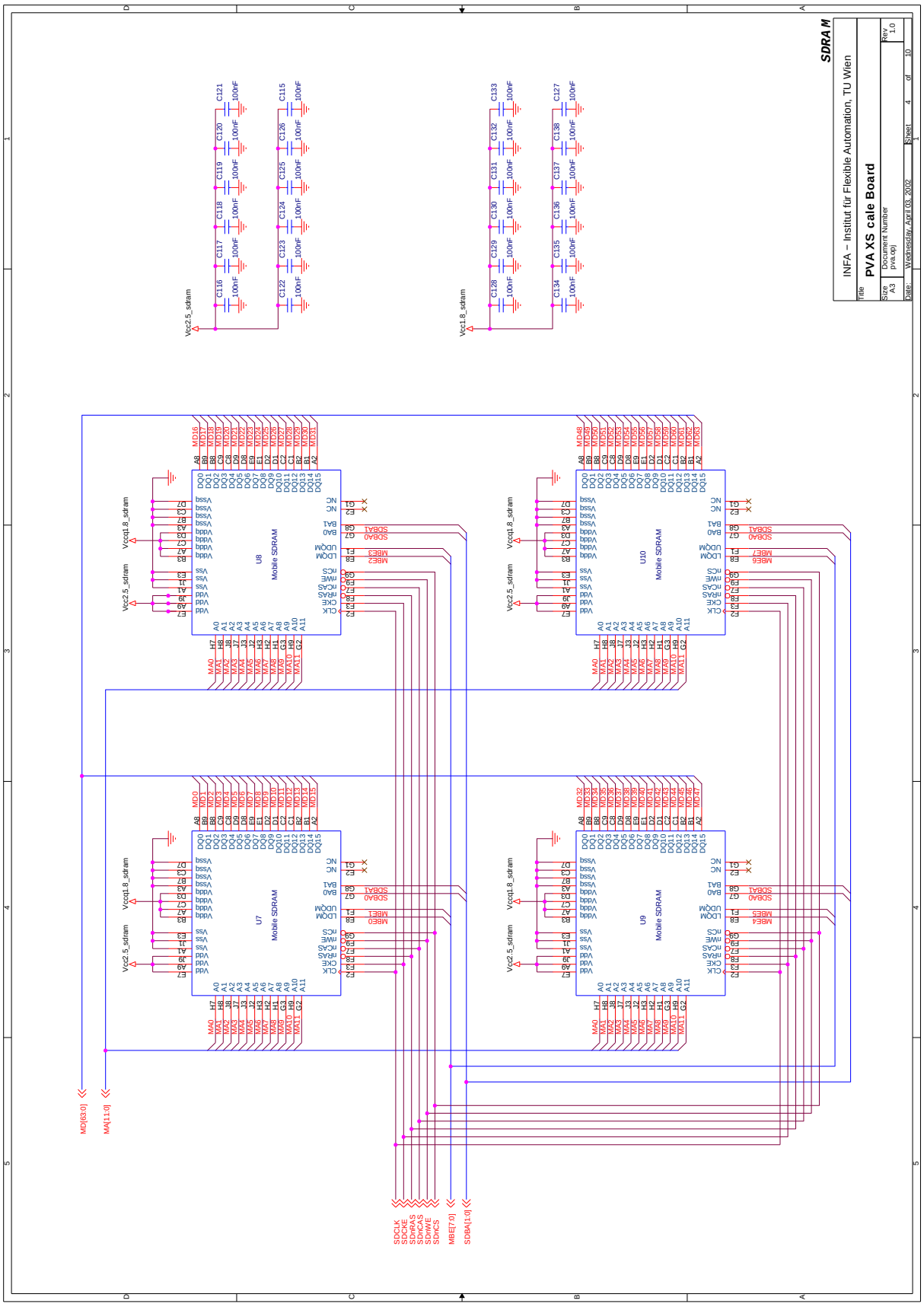
Anhang A

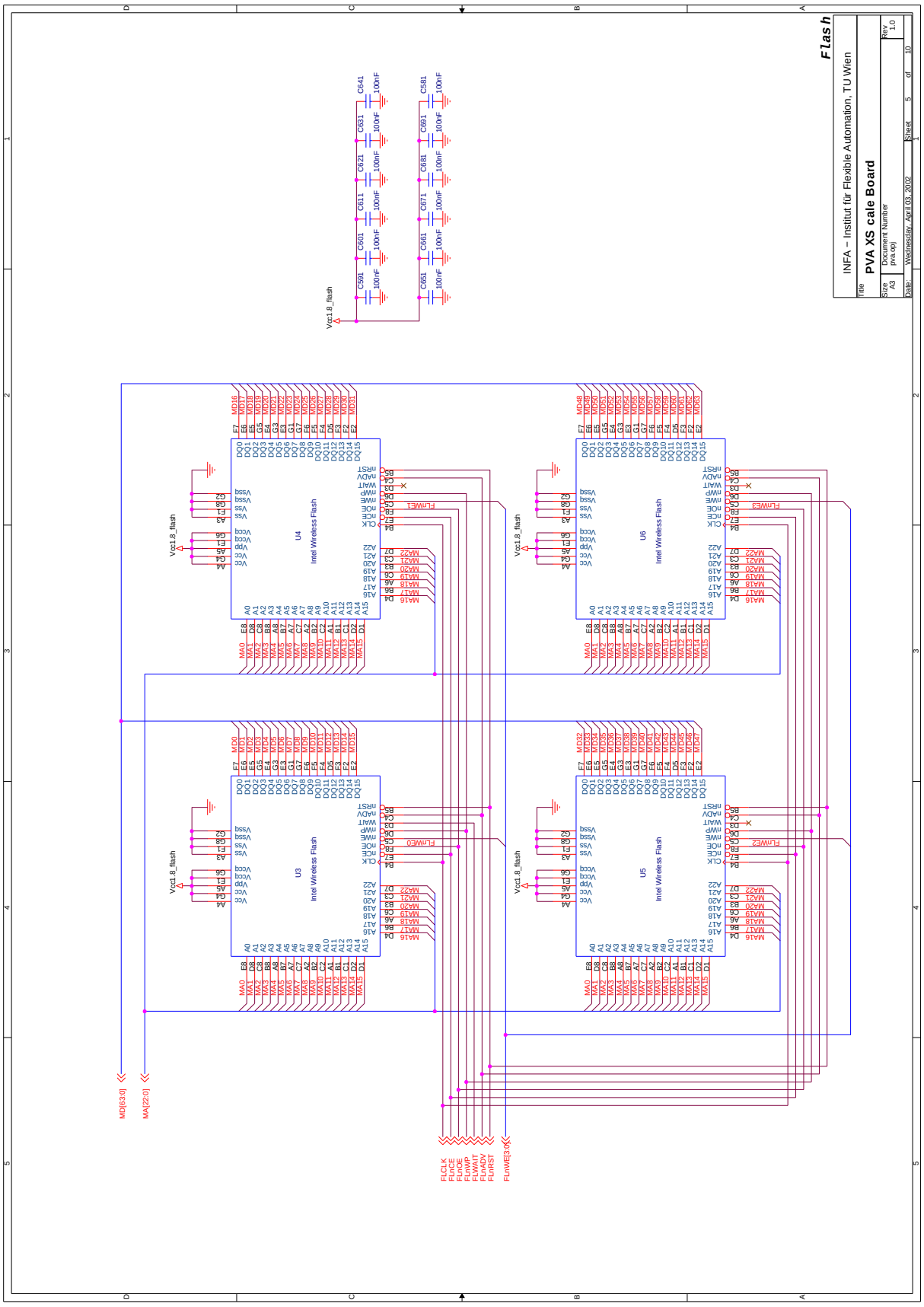
Schaltpläne

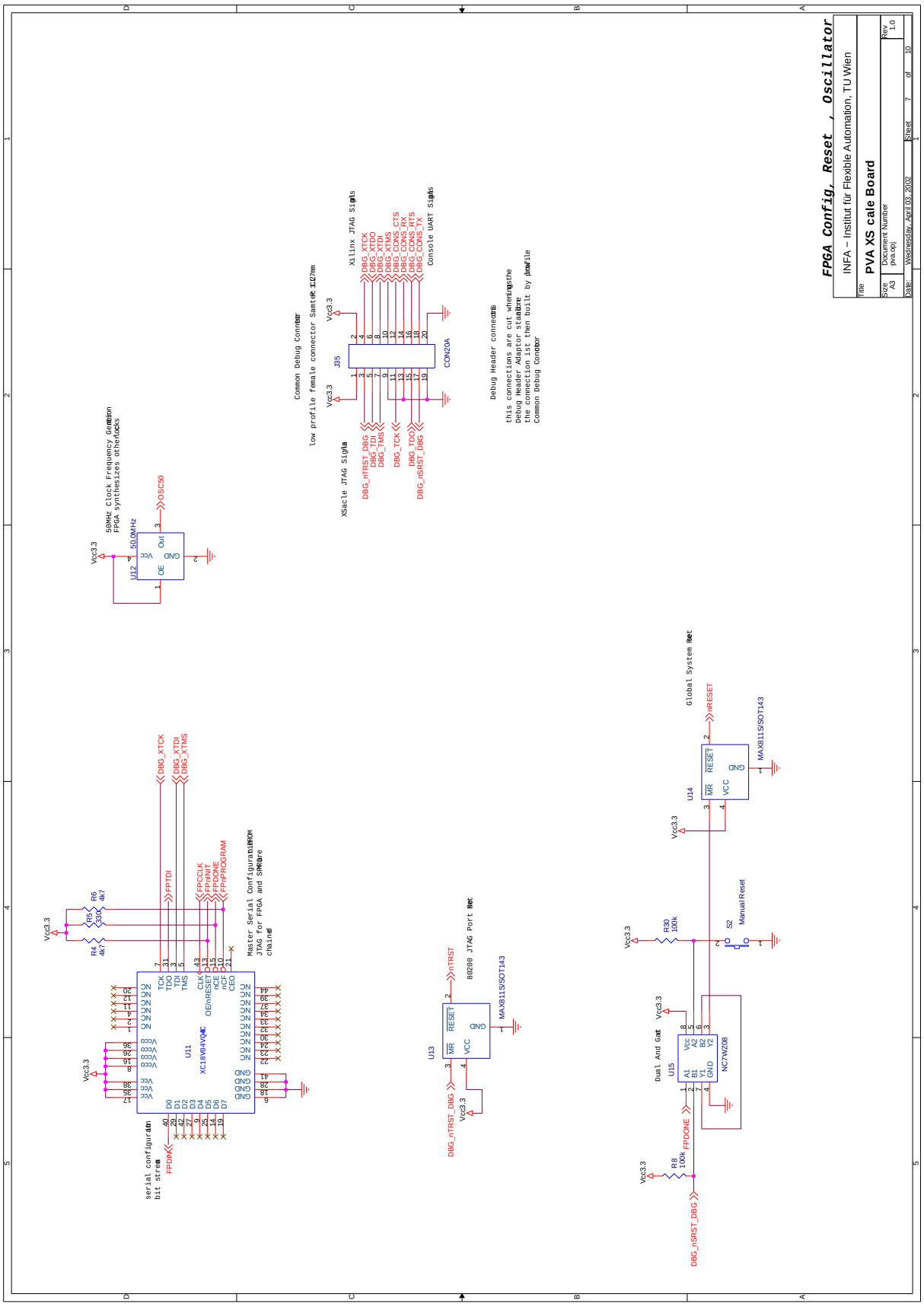
Die Schaltpläne für das neue Mikroprozessormodul wurden mit dem Softwarepaket OrCAD-Capture erstellt. Der gesamte Schaltplan ist zur besseren Übersichtlichkeit auf folgende Teilpläne aufgeteilt:

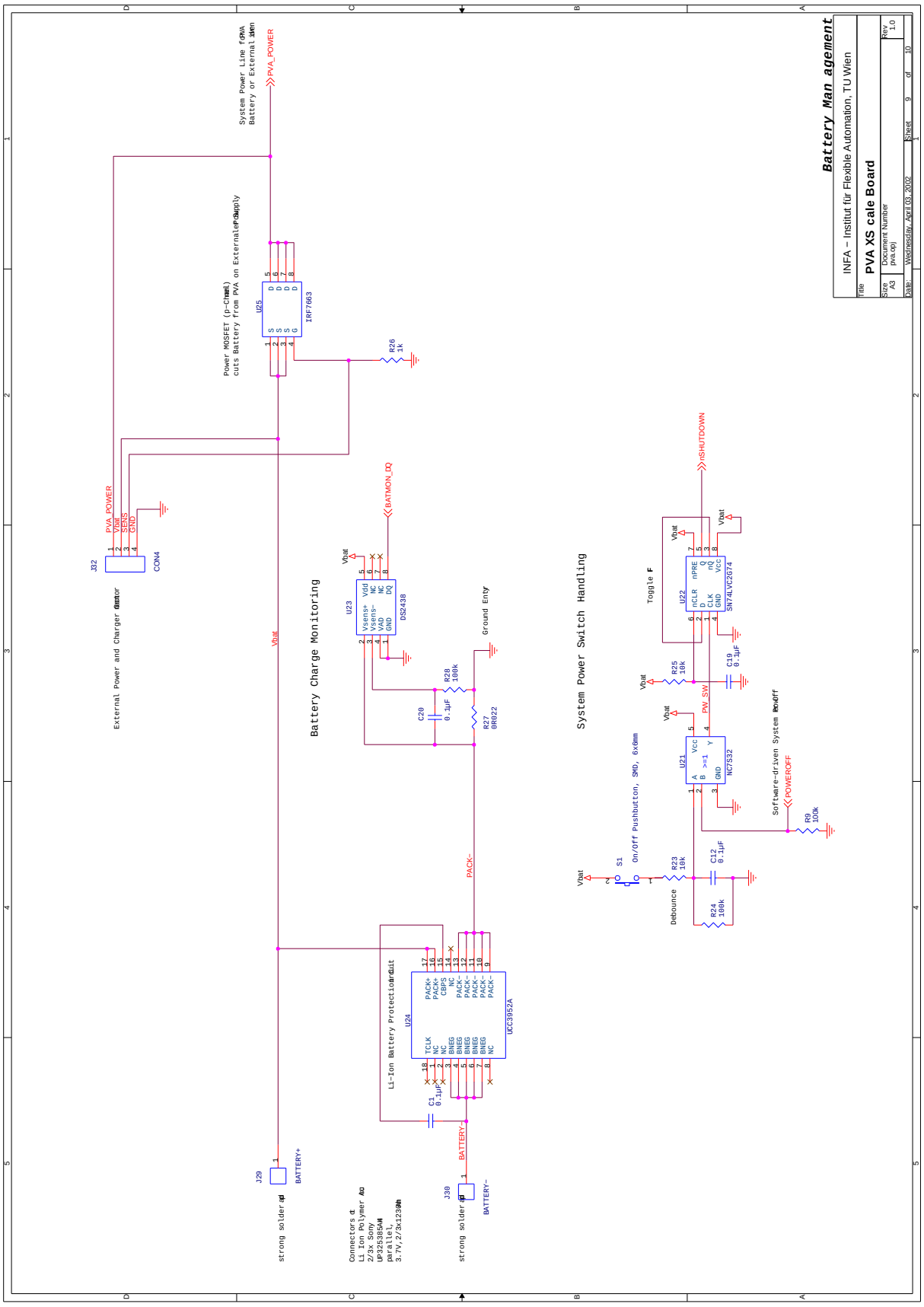

XScale Pro cessor
INFA – Institut für Flexible Automation, TU Wien
PVA XS cale Board
Title
Size A3
Document Number pva.opj
Rev 1.0
Date: Wednesday, April 03, 2002
Sheet 2 of 10
RN1
10kx4
RN2
10kx4
U1
80200
R11 10,1/8W
C7 4,7µF
C6 10nF
R1 47k
R2 47k
R3 47k
Vcc1.5_xscale
Vcc3.3_xscale
Vcc3.3

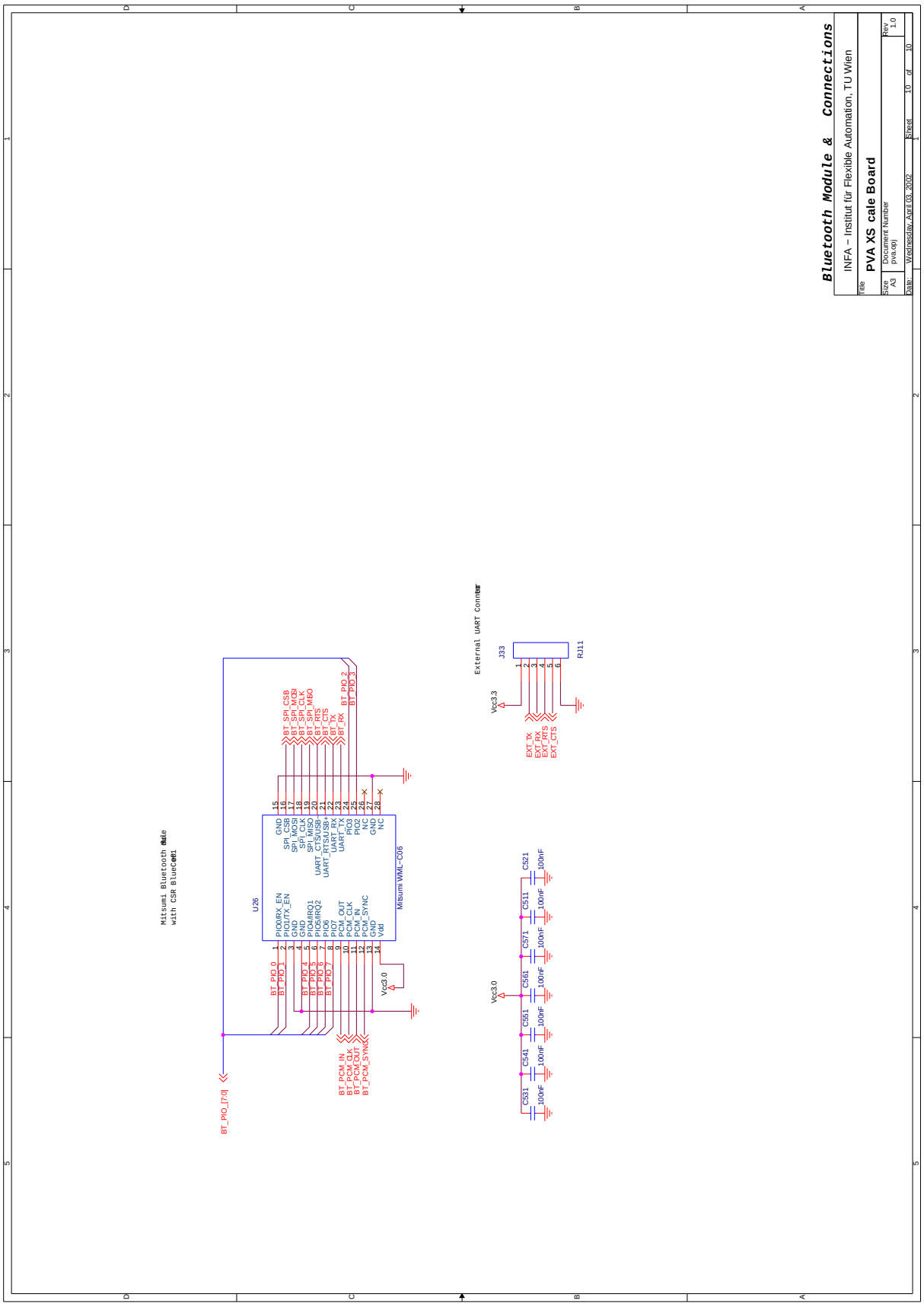

FPGA
INFA – Institut für Flexible Automation, TU Wien
PVA XS cale Board
U2
XC2V500FG456
Xilinx Virtex II FPGA
BANK0
BANK1
BANK2
BANK3
BANK4
BANK5
BANK6
BANK7
Vcc1.5_fpga
Vcc3.3_fpga
Vcc1.8_fpga
Vcc3.3
SDCLK
Delay Trace

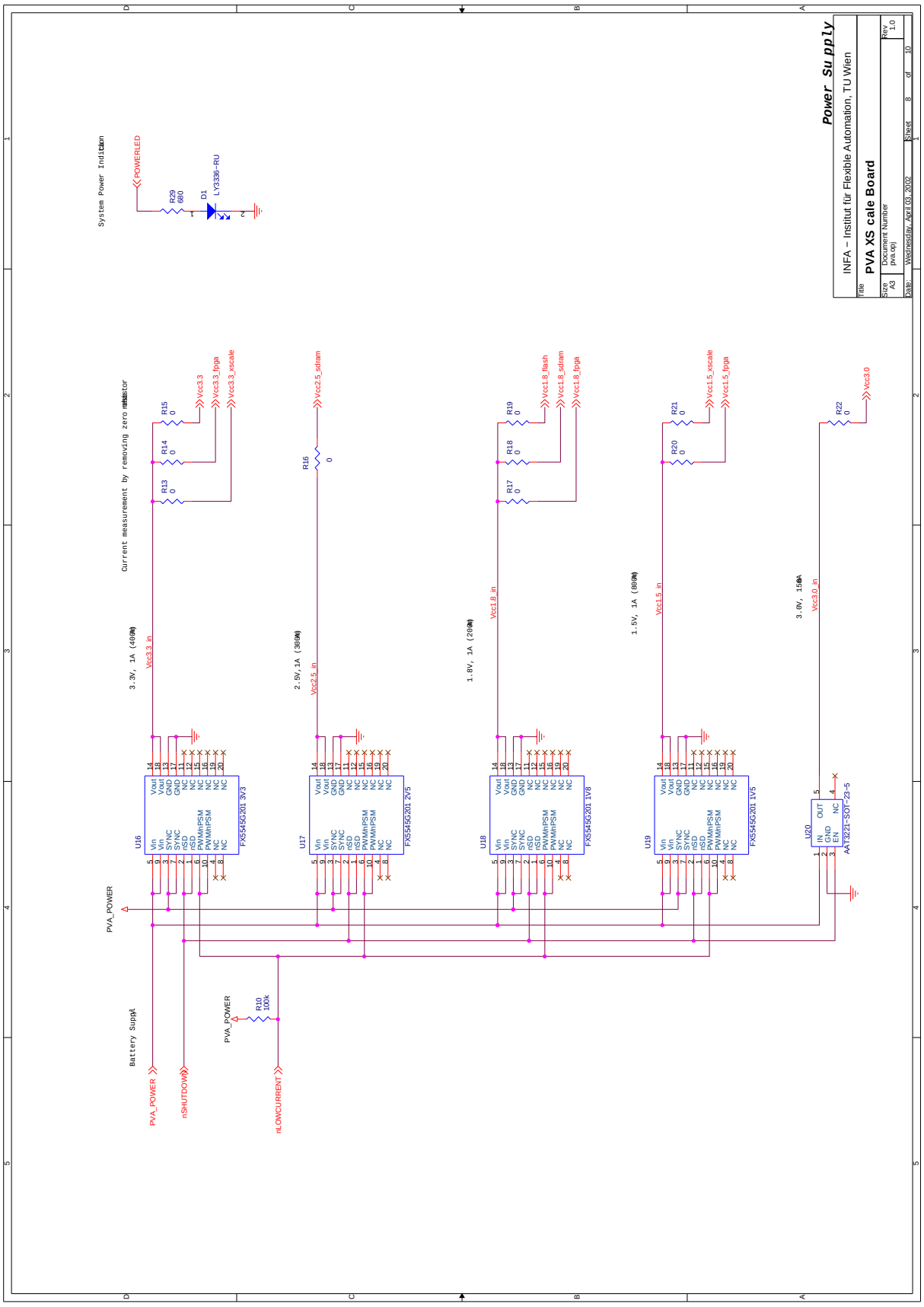

SDRAM
INFA – Institut für Flexible Automation, TU Wien
Title
PVA XS cale Board
Size A3
Document Number pva.obj
Rev 1.0
Date: Wednesday, April 03, 2002
Sheet 4 of 10
Vcc2.5_sdram
Vcc1.8_sdram
C116 100nF
C117 100nF
C118 100nF
C119 100nF
C120 100nF
C121 100nF
C122 100nF
C123 100nF
C124 100nF
C125 100nF
C126 100nF
C115 100nF
C128 100nF
C129 100nF
C130 100nF
C131 100nF
C132 100nF
C133 100nF
C134 100nF
C135 100nF
C136 100nF
C137 100nF
C138 100nF
C127 100nF
U8
U10
U7
U9
Mobile SDRAM
MD[63:0]
MA[11:0]
SDCLK
SDCKE
SDRAS
SDCAS
SDWE
SDCS
MBE[7:0]
SDBA[1:0]

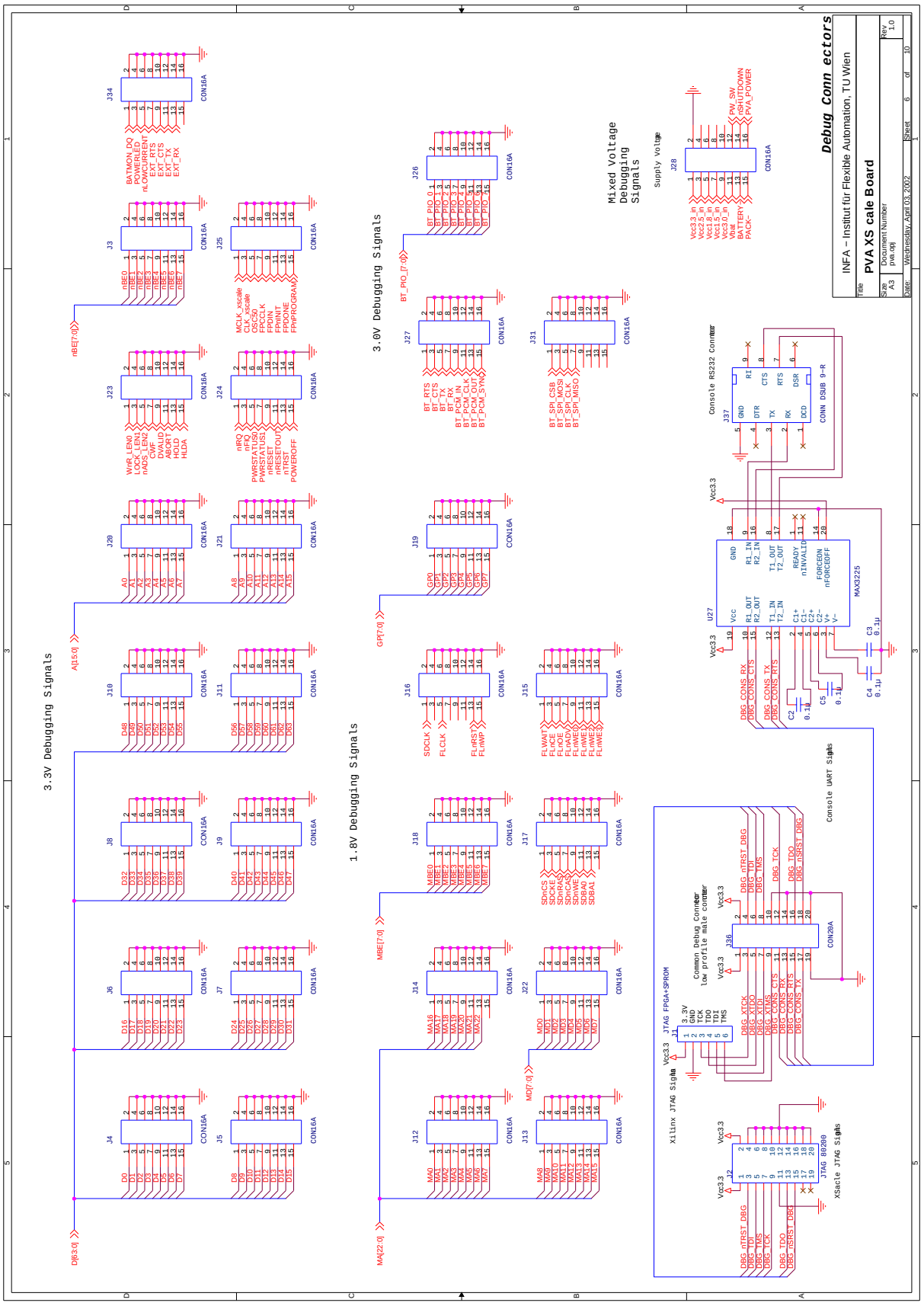

Flash
INFA – Institut für Flexible Automation, TU Wien
Title
PVA XS cale Board
Size A3
Document Number pva.dsn
Rev 1.0
Date: Wednesday, April 03, 2002
Sheet 5 of 10
Vcc1.8_flash
C641 100nF
C631 100nF
C621 100nF
C611 100nF
C601 100nF
C591 100nF
C581 100nF
C691 100nF
C681 100nF
C671 100nF
C661 100nF
C651 100nF
U4
U3
U6
U5
Intel Wireless Flash
MD[63:0]
MA[22:0]
FLCLK
FLnCE
FLnOE
FLnWP
FLnWAIT
FLnADV
FLnRST
FLnWE[3:0]


FPGA Config, Reset , Oscillator
INFA – Institut für Flexible Automation, TU Wien
PVA XS cale Board
Size A3
Document Number pva.obj
Rev 1.0
Date: Wednesday, April 03, 2002
Sheet 7 of 10
50MHz Clock Frequency Generation
FPGA synthesizes other clocks
U12
50.0MHz
OSC50
Vcc3.3
Common Debug Connector
low profile female connector Samtec IDC/mm
Xilinx JTAG Signals
XSacle JTAG Signals
J35
CON20A
DBG_XTCK
DBG_XTDO
DBG_XTDI
DBG_XTMS
DBG_nTRST_DBG
DBG_TDI
DBG_TMS
DBG_TCK
DBG_TDO
DBG_nSRST_DBG
DBG_CONS_CTS
DBG_CONS_RX
DBG_CONS_RTS
DBG_CONS_TX
Console UART Signals
Debug Header connects
this connections are cut when using the
Debug Header Adapter Cable. Then
the connection is then built by the
Common Debug Connector
Master Serial Configuration PROM
JTAG for FPGA and SRAM
chained
U11
XC18V04VQ44C
R6 4k7
R5 330
R4 4k7
FPTDI
FPCCLK
FPINIT
FPDONE
FPGPROGRAM
serial configuration bit stream
FPDIN
U13
MAX811S/SOT143
80200 JTAG Port Reset
nTRST
DBG_nTRST_DBG
U14
RESET
Global System Reset
nRESET
R30 100k
S2
ManualReset
U15
Dual And Gate
NC7WZ08
R8 100k
FPDONE
DBG_nSRST_DBG


Battery Management
INFA – Institut für Flexible Automation, TU Wien
PVA XS cale Board
Battery Charge Monitoring
System Power Switch Handling
External Power and Charger Connector
Li-Ion Battery Protection Circuit
Power MOSFET (p-Channel)
cuts Battery from PVA on External Power
Toggle F
Debounce
Software-driven System On/off
PVA_POWER
BATTERY+
BATTERY-
BATMON_DQ
nSHUTDOWN
PACK+
PACK-
Vbat
Ground Entry
On/Off Pushbutton, SMD, 6x6mm
DS2438
UCC3952A
SN74LVC2G74
NC7S32
IRF7663
CON4


Bluetooth Module & Connections
INFA – Institut für Flexible Automation, TU Wien
PVA XS cale Board
Mitsumi Bluetooth Module
with CSR BlueCore01
U26
Mitsumi WML-C06
External UART Connector
J33
RJ11
Vcc3.3
Vcc3.0
BT_PIO_[7:0]
BT_PCM_IN
BT_PCM_OUT
BT_PCM_SYNC
EXT_TX
EXT_RX
EXT_RTS
EXT_CTS
Size A3
Document Number pva.opj
Rev 1.0
Date: Wednesday, April 03, 2002
Sheet 10 of 10


Power Supply
INFA – Institut für Flexible Automation, TU Wien
Title
PVA XS cale Board
Size
A3
Document Number
pva.dsn
Rev
1.0
Date:
Wednesday, April 03, 2002
Sheet
8
of
10
System Power Indication
POWERLED
R29
680
D1
LY336-RU
Current measurement by removing zero resistor
3.3V, 1A (400mA)
2.5V,1A (380mA)
1.8V, 1A (200mA)
1.5V, 1A (380mA)
3.0V, 150mA
Vcc3.3_in
Vcc2.5_in
Vcc1.8_in
Vcc1.5_in
Vcc3.0_in
R15
R14
R13
R16
R19
R18
R17
R21
R20
R22
Vcc3.3
Vcc3.3_fpga
Vcc3.3_xscale
Vcc2.5_sdram
Vcc1.8_flash
Vcc1.8_sdram
Vcc1.8_fpga
Vcc1.5_xscale
Vcc1.5_fpga
Vcc3.0
U16
U17
U18
U19
U20
FXS5456201 3V3
FXS5456201 2V5
FXS5456201 1V8
FXS5456201 1V5
AAT3221-SOT-23-5
Battery Supply
PVA_POWER
nSHUTDOWN
nLOWCURRENT
R10
100k


3.3V Debugging Signals
1.8V Debugging Signals
3.0V Debugging Signals
Mixed Voltage Debugging Signals
Debug Conn ectors
INFA – Institut für Flexible Automation, TU Wien
PVA XS cale Board
Console RS232 Connector
Console UART Signals
Common Debug Connector Low profile male connector
Xilinx JTAG Signals
XScale JTAG Signals
JTAG FPGA+SPROM
Supply Voltage
CONN DSUB 9-R
MAX3225
CON20A
JTAG 80200

Anhang B

Leiterplattenlayout

Ausgehend vom Schaltplan wird mithilfe des Softwarepaketes OrCAD-Layout die Gestaltung des PCB und die Entflechtung der Signalleitungen vorgenommen. Im vorliegenden Fall ergibt sich durch die hohe Anzahl von Anschlüssen der BGA-Bausteine (z. B. FPGA: 456 Anschlüsse) sowie der kleinen geometrischen Scheckkarten-Bauform eine enorm hohe Dichte an Signalleitungen. Dies läßt sich nur durch eine entprechend gewählte Anzahl von Signalebenen in einem Multilayer-PCB realisieren. Für das PCB des neuen Mikroprozessormoduls werden neben den zwei Bauteilebenen (Vorder- und Rückseite) noch weitere acht innere Signalebenen eingeführt, von denen vier für die unterschiedlichen Versorgungsspannungen zur Verfügung stehen.

Design for Test

Das PCB wurde so gestaltet, dass sich im Scheckkarten-großen Zentrum der Platine der eigentliche Kern der Schaltung befindet und sich sämtliche, nur für Test- und Inbetriebnahmezwecke vorgesehenen Schnittstellen am Rand der Karte angeordnet werden. Bei erfolgreicher Inbetriebnahme lässt sich das PCB-Layout sehr einfach durch Weglöschen aller nach außen in den Rand der Karte führenden Signalleitungen auf den Schaltungskern verkleinern. Im Prinzip kann diese Verkleinerung auch durch mechanisches Abschneiden des Randes herbeigeführt werden.

Aspekte der Leiterbahn-Entflechtung

Für die systematische Entflechtung der Signalleitungen wurden für die mehrfach vorkommenden Bauteile (SDRAM, Flash) regelmäßige Strukturen entworfen, die sich durch einfaches Kopieren vervielfachen lassen. Vertikale bzw. horizontale Signalleitungen werden dabei in unterschiedlichen Signalebenen geführt.

Die Entflechtung der 456 Anschlüsse am FPGA ist eine Herausforderung für sich. Allein das Herausführen aller Leitungen benötigt zumindest drei Signalebenen

(Abb. 77). Das Auskreuzen von Leitungen bedarf weiterer Ebenen. Um den Aufwand in Grenzen zu halten bzw. ein erfolgreiches Entflechten überhaupt erst zu ermöglichen, wurde die Belegung der Anschlüsse am FPGA so vorgenommen, dass möglichst wenig Kreuzungen entstehen. Das führt gezwungenermaßen zu einer unaufgeräumten, wahllos erscheinenden Darstellung der Anschlüsse im Schaltplan. Eine weitere Randbedingung für die Entflechtung des FPGA sind die unterschiedlichen Spannungpegel (1.8V und 3.3V) an den acht Signalgruppen des FPGA. Die Signalgruppen wurden daher so aufgeteilt, dass die 1.8V-Gruppen den Speicherbausteinen und die 3.3V-Gruppen dem Mikroprozessor zugewandt sind. Die mittlere Länge der Signalleitungen lässt sich so erheblich verkürzen.

Einteilung in Signalebenen

Die Wahl der notwendigen Anzahl von Signalebenen ist zu Beginn der Entflechtungstätigkeiten sehr schwer abzuschätzen und kann nur aus der Erfahrung ermittelt werden. Grundsätzlich ist aus Gründen der Fertigungskosten eine möglichst geringe Anzahl von Ebenen anzustreben. Andererseits wird durch zusätzliche Ebenen die Entflechtung erleichert und die mittlere Singalleitungslänge reduziert, was sich positiv auf die Signalintegrität auswirkt.

Für das PCB des neuen Mikroprozessormoduls wurde die Einteilung der Signalebenen folgendermaßen durchgeführt:

- Ebene 0 (Abb. B.1): Diese Bauteilseite kann wegen der Aufbringung der Bausteine nur beschränkt zur Führung von Leiterbahnen herangezogen werden. Trotzdem wird diese Ebene intensiv für die Entflechtung benutzt.
- Ebene 1 (Abb. B.2): Die erste Innenlage ist fertigungstechnisch relativ einfach über kleine Mikrovias von Ebene 0 aus erreichbar und dient zur Führung von Signalleitungen, die überwiegend parallel zur langen Kante des Boards verlaufen.
- Ebene 2 (Abb. B.3): Ebenso wie Ebene 1 dient diese Signalebene zur Entflechtung der Leitungen. In erster Linie werden die Leiterbahnen auf dieser Ebene überwiegend entlang der kurzen Kante des Boards geführt.
- Ebene 3 (Abb. B.4): Wie Ebenen 1 und 2 werden hier Signale geführt, wobei die Hauptrichtung vertikal ist.
- Ebene 4 (Abb. B.5): Hier werden jene Leitungen geführt, die in den oberen Ebenen nicht mehr entflechtet werden können, dazu gehören z. B. einige Leiterbahnen, die zu den Testschnittstellen führen.
- Ebene 5 (Abb. B.6) bildet in der Mitte des PCB die Signalmasse in Form einer durchgehenden leitfähigen Ebene, die im Bereich der Durchkontaktierungen Aussparungen aufweist.

- Ebene 6 (Abb. B.7) und 7 (Abb. B.8) sind als Versorgungsspannungsebenen konzipiert.
- Ebene 8 (Abb. B.9) dient sowohl als Versorgungsspannungsebene als auch als Signalführungsebene für jene Leiterbahnen, die auf der Bauteilrückseite (Ebene 9) keinen Platz mehr finden.
- Ebene 9 (Abb. B.10): Diese Bauteilrückseite trägt die Bausteine der Spannungsversorgung, die Stützkondensatoren sowie den Großteil der diskreten Bauelemente und dient zur Führung von Signalleitungen.

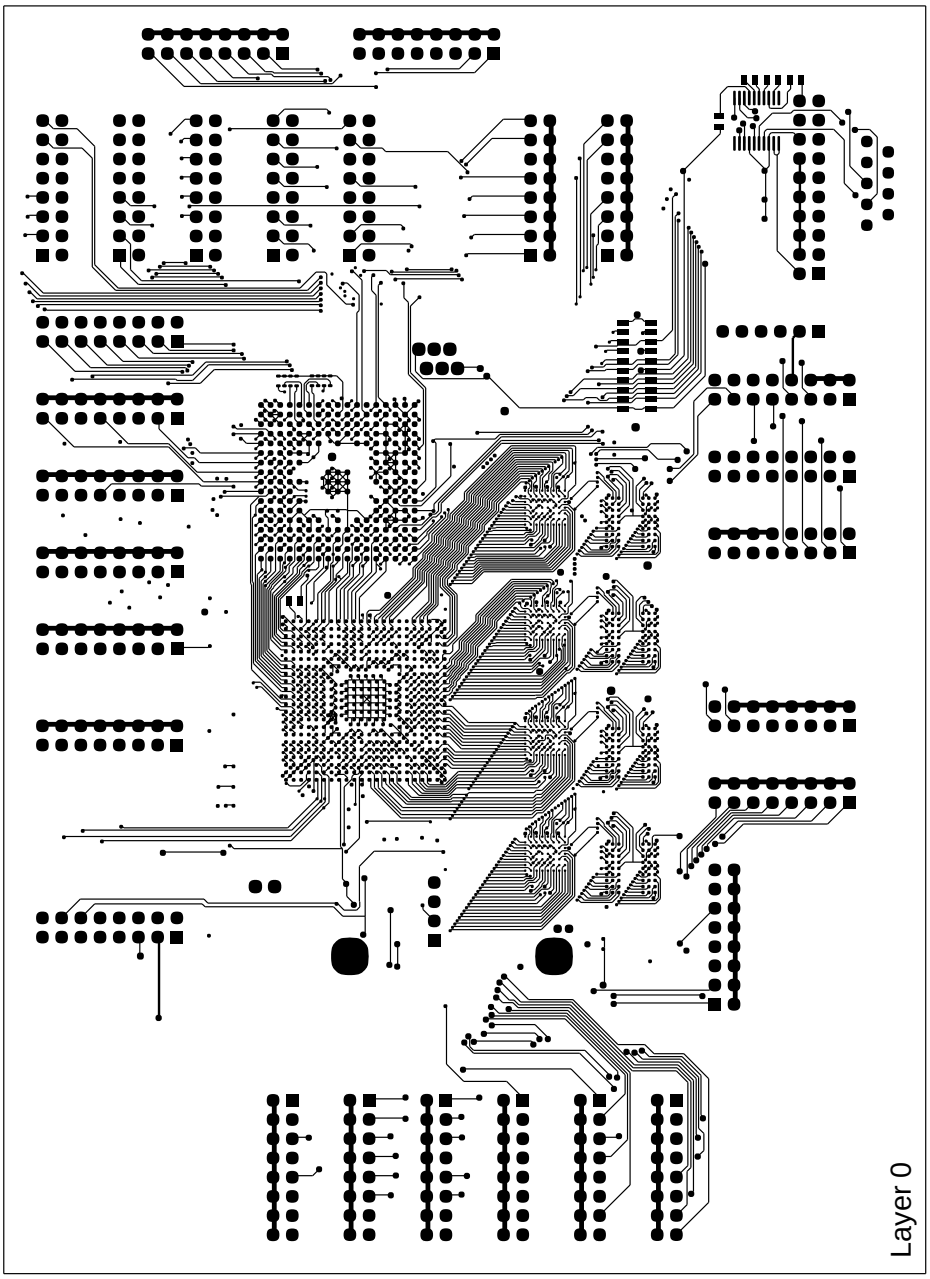


Abbildung B.1: *Leiterplattenlayout, Ebene 0.* Die Bauteilseite trägt die hochintegrierten Bausteine des Mikroprozessors, des Speichers und des FPGA und dient zur Entflechtung von Signalleitungen.

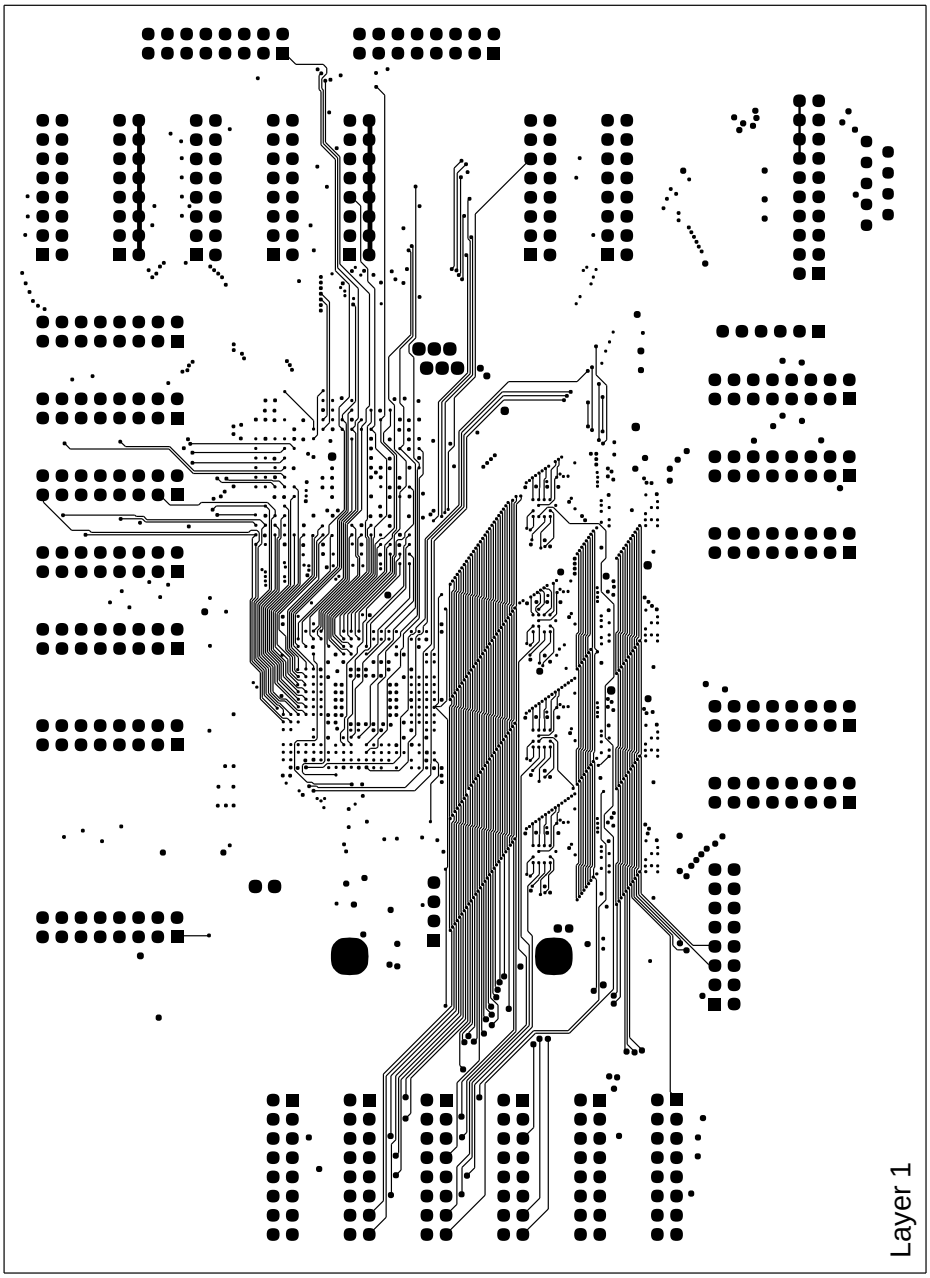


Abbildung B.2: *Leiterplattenlayout, Ebene 1.* Diese Ebene wird in erster Linie für die Führung von Leiterbahnen verwendet, die parallel zur längeren Kante des Boards verlaufen.

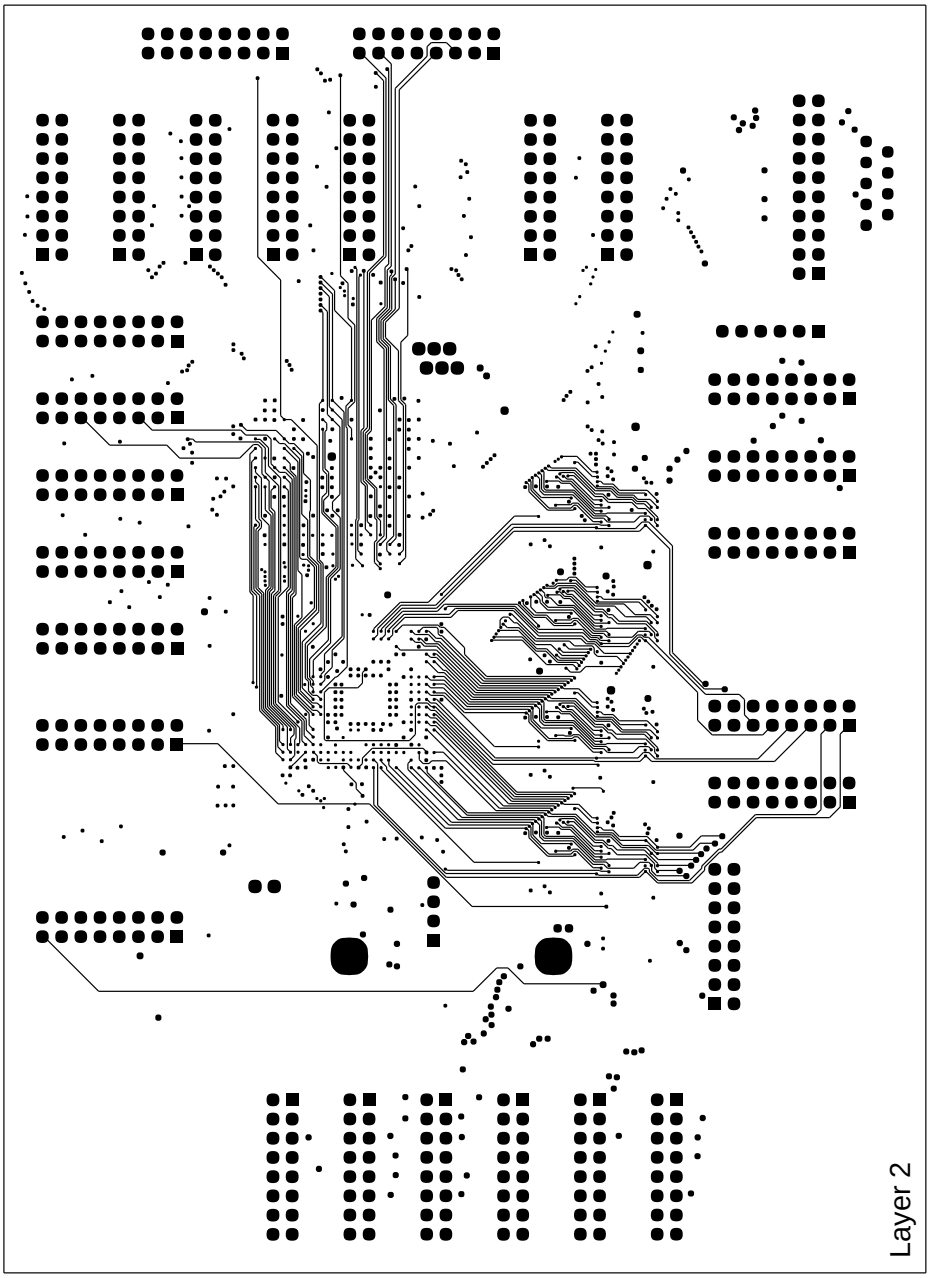


Abbildung B.3: *Leiterplattenlayout, Ebene 2.* Diese Ebene wird in erster Linie für die Führung von Leiterbahnen verwendet, die parallel zur kürzeren Kante des Boards verlaufen.

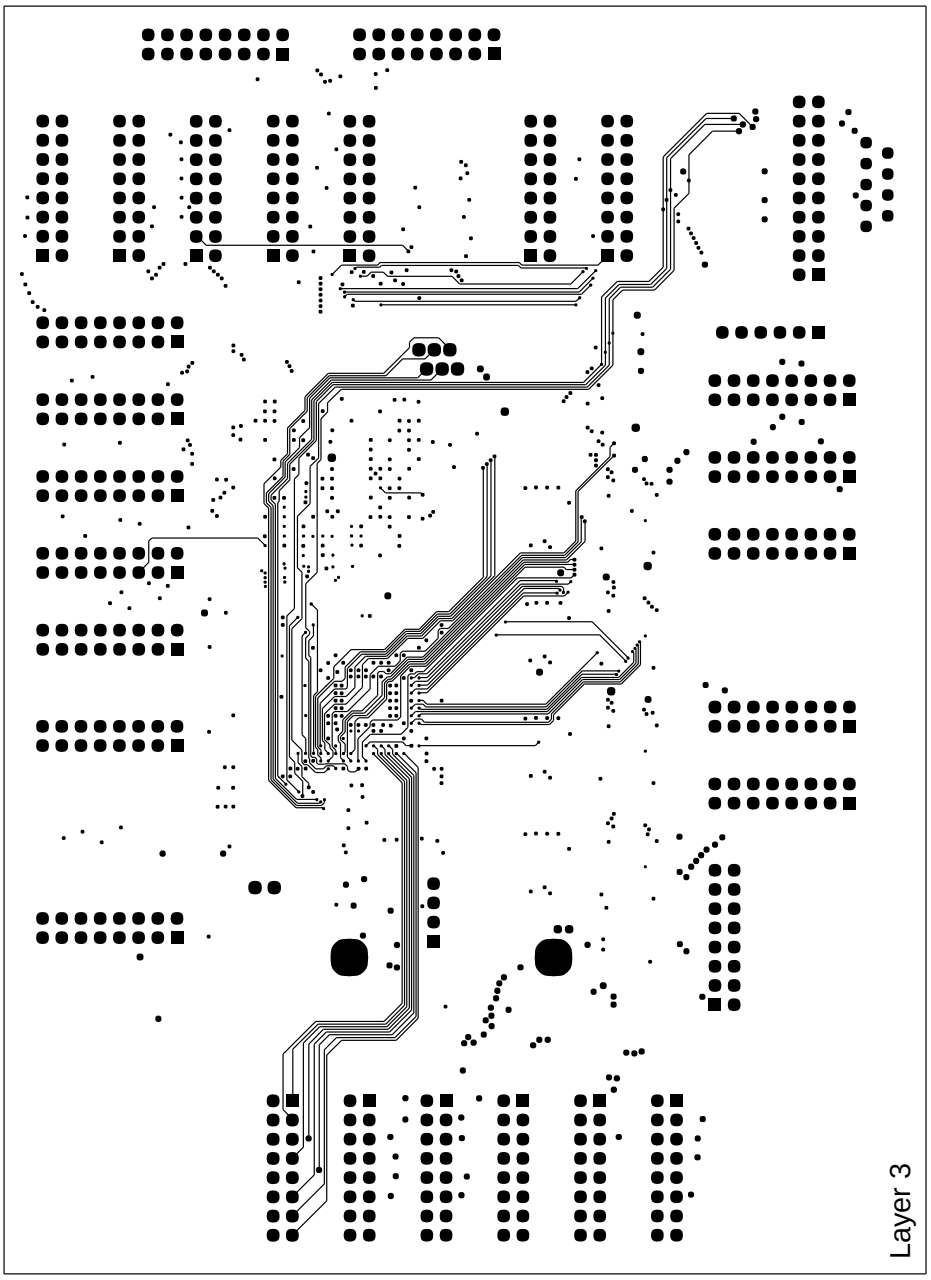


Abbildung B.4: *Leiterplattenlayout, Ebene 3.* Diese Ebene dient zur Führung von Leiterbahnen, die am FPGA angeschlossen sind und nicht in darüberliegenden Ebenen Platz finden.

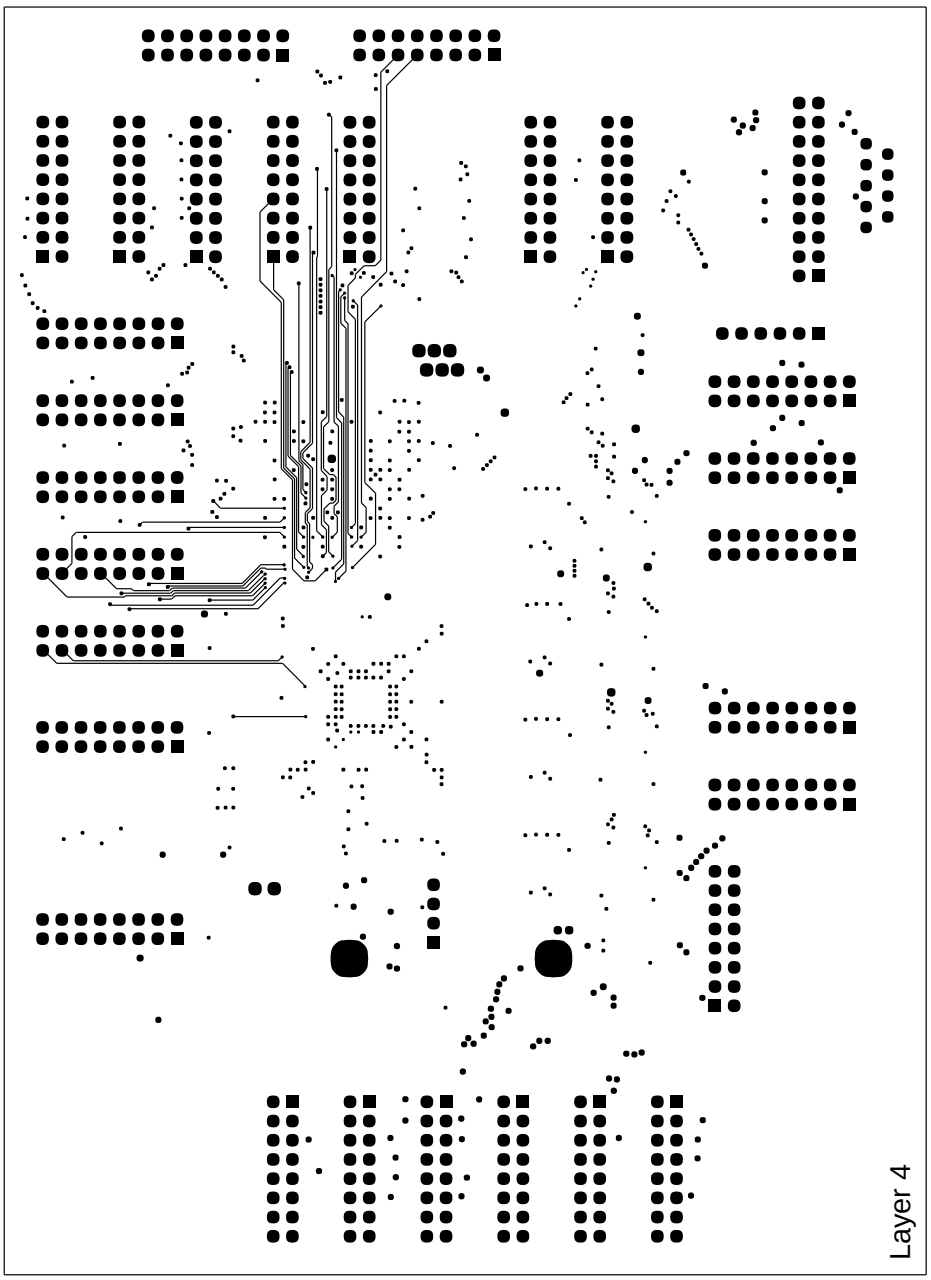


Abbildung B.5: *Leiterplattenlayout, Ebene 4.* In dieser Ebene werden vorwiegend Signale zwischen Mikroprozessor und Testschnittstellen geführt.

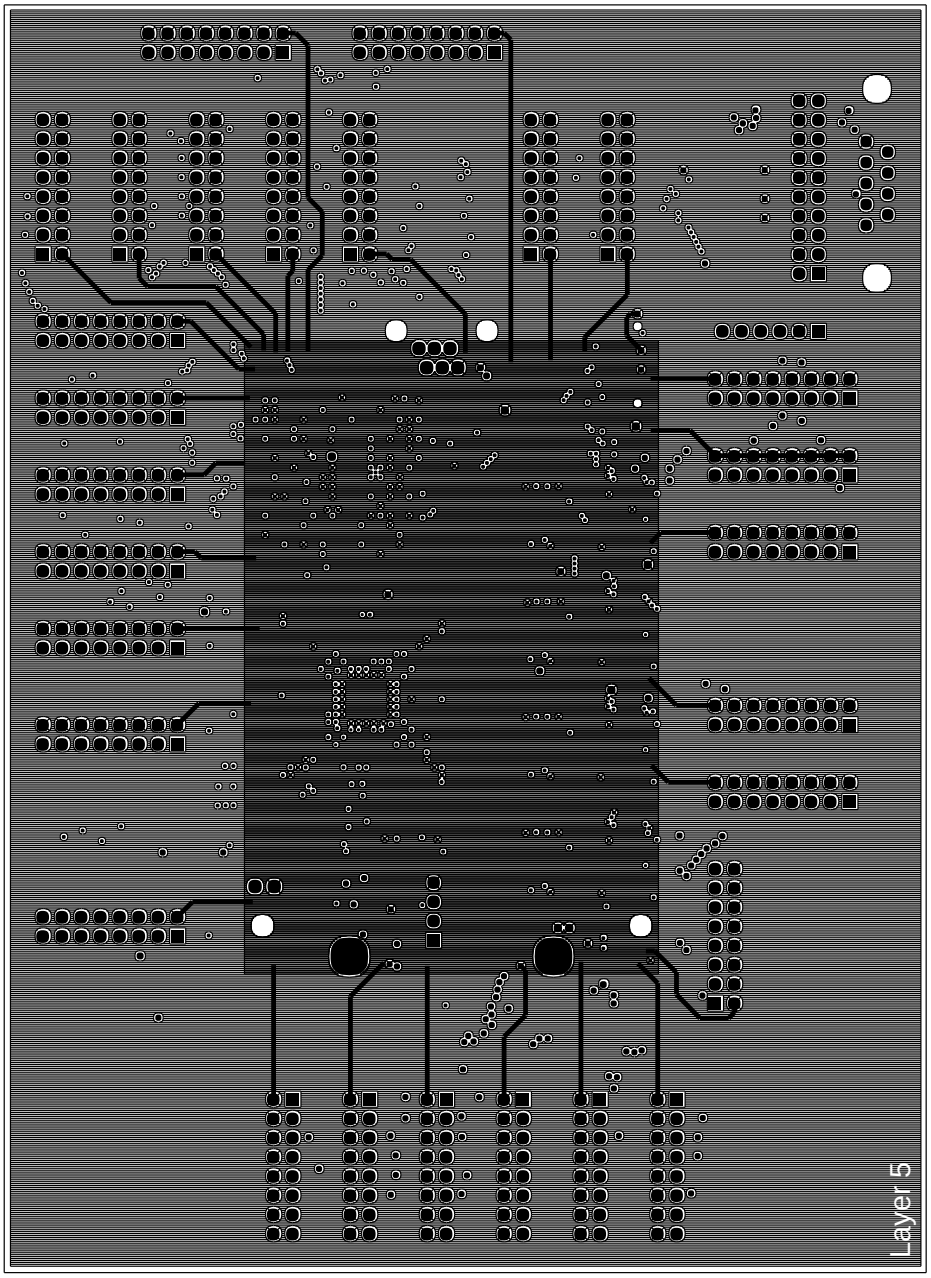


Abbildung B.6: *Leiterplattenlayout, Ebene 5.* Diese Ebene bildet die Signal-Masse.

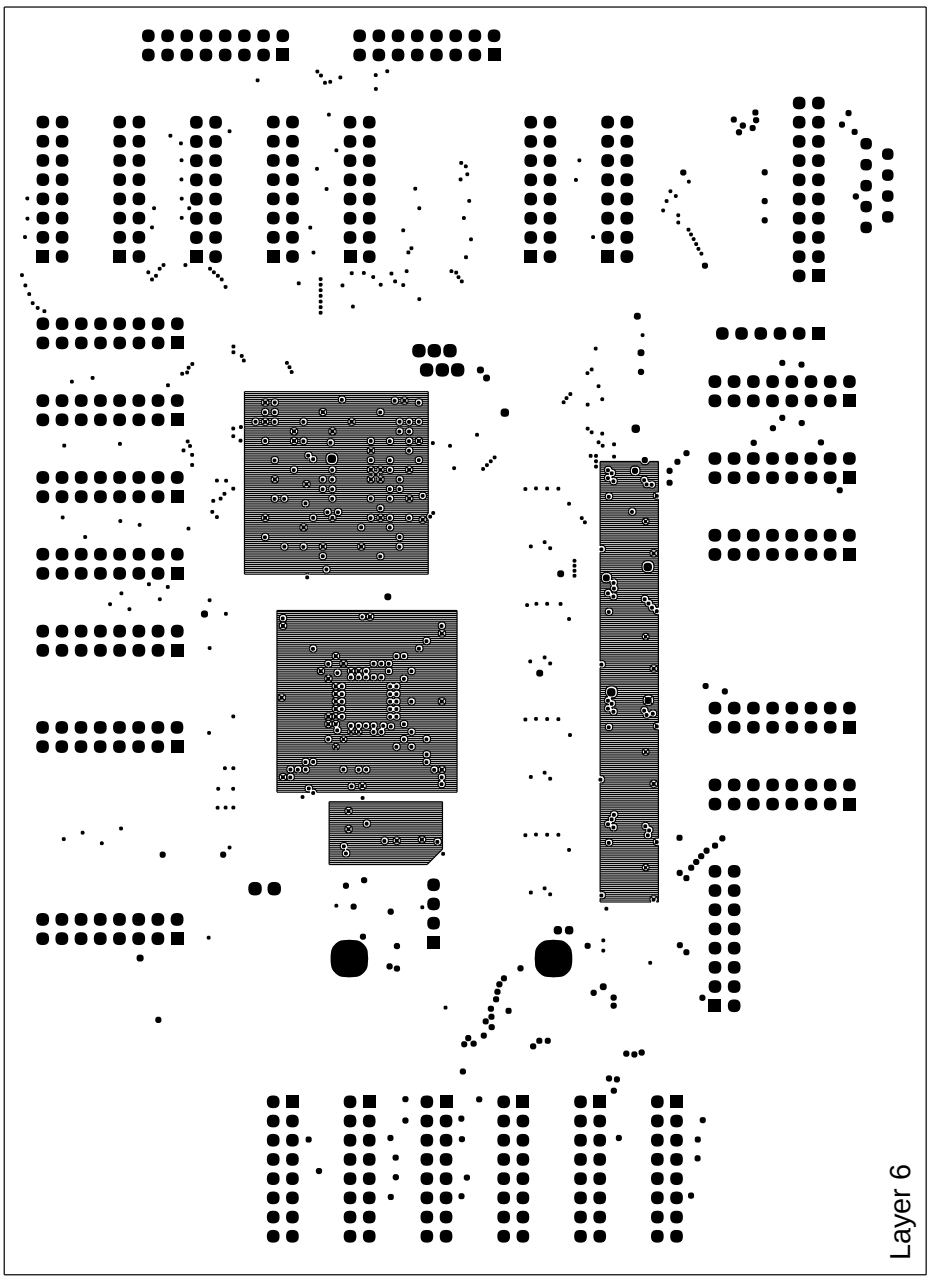


Abbildung B.7: *Leiterplattenlayout, Ebene 6.* Diese Ebene dient als Versorgungsspannungsebene.

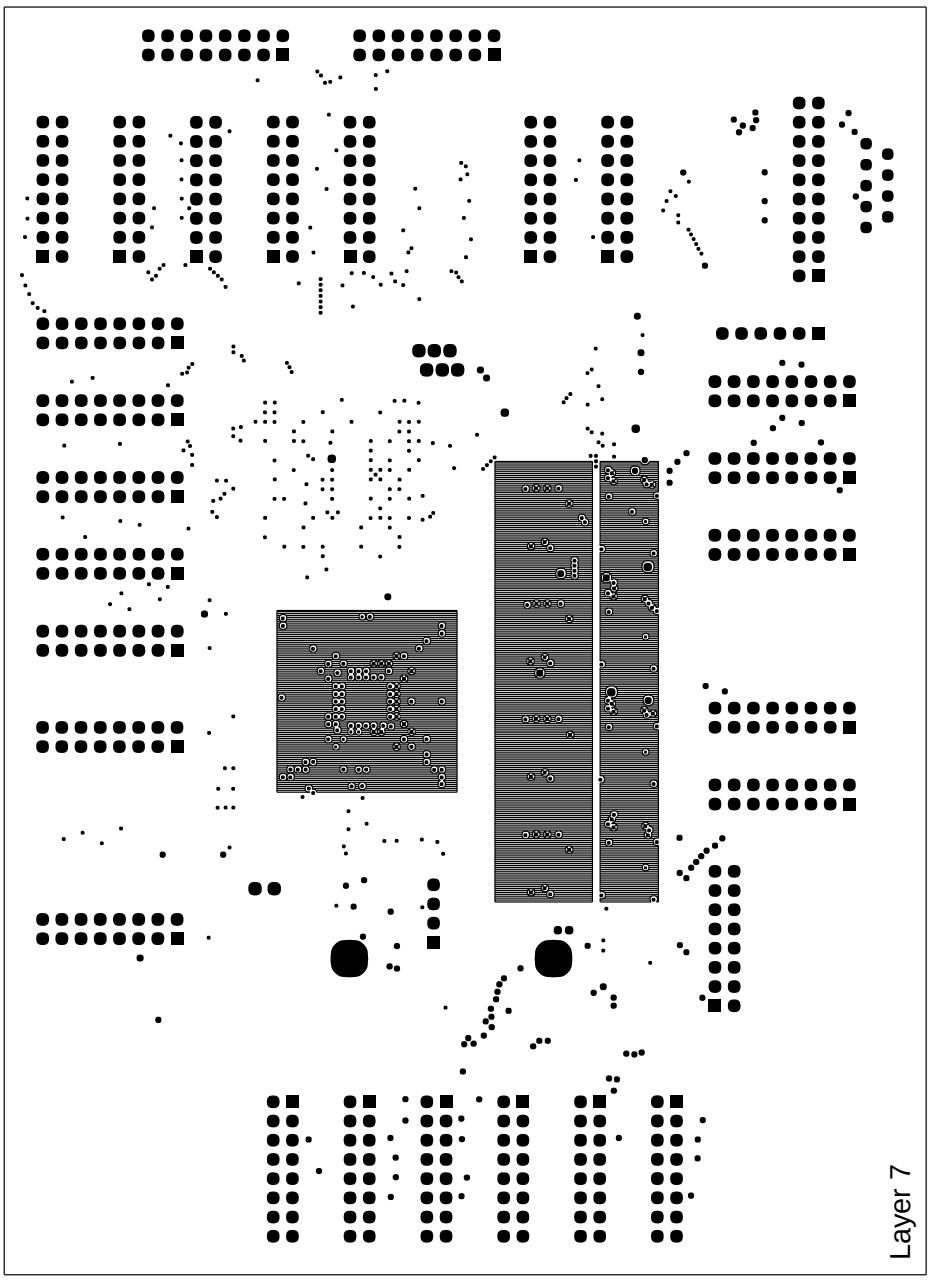


Abbildung B.8: *Leiterplattenlayout, Ebene 7.* Diese Ebene dient als Versorgungsspannungsebene.

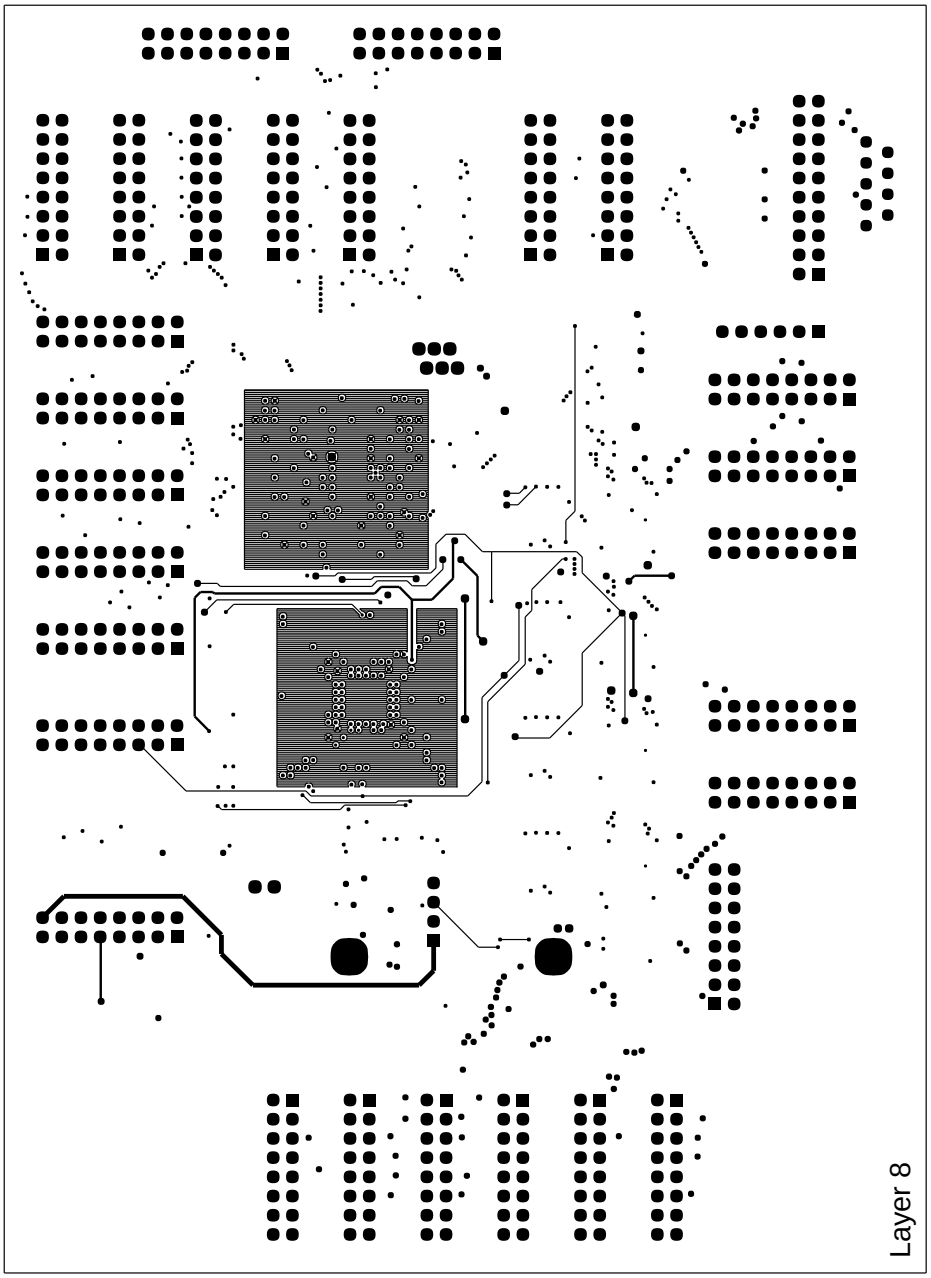


Abbildung B.9: *Leiterplattenlayout, Ebene 8.* In dieser Ebene werden neben Versorgungsspannungen auch einige Leitungen geführt, die an der Bauteilrückseite (Ebene 9) keinen Platz finden.

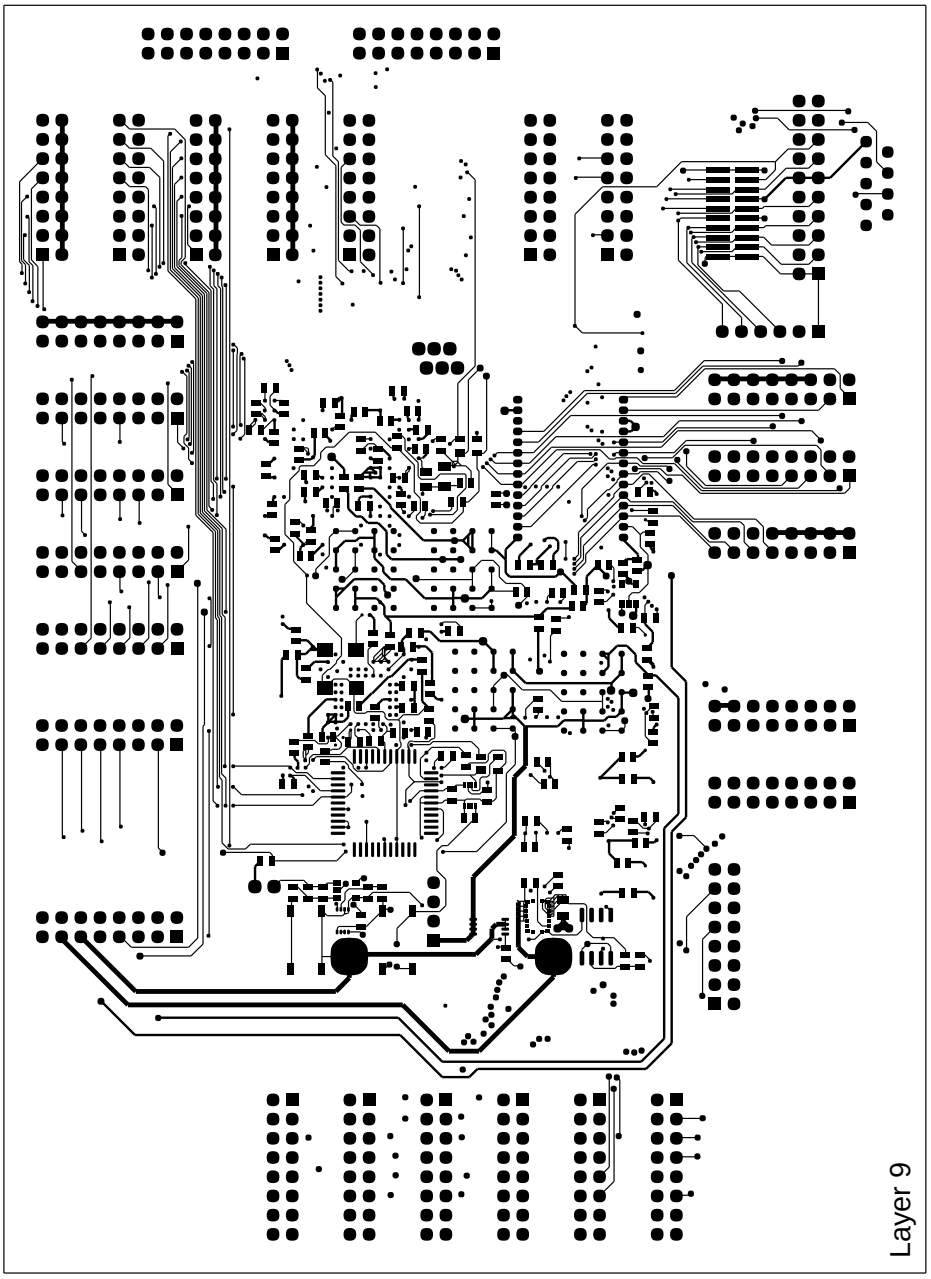


Abbildung B.10: *Leiterplattenlayout, Ebene 9.* Diese Ebene bildet die Bauteilrückseite, an der die Bausteine der Spannungsversorgung, die Stützkondensatoren sowie den Großteil der diskreten Bauelemente angebracht sind.

Anhang C

Adressraum des Mikroprozessormoduls

Da der Intel XScale 80200 Prozessor nicht zwischen Speicher- und I/O-Zugriffen unterscheidet, sind sämtliche Register der Ein-/Ausgabemodule vom Logikbaustein direkt in den Adressraum des Prozessors einzublenden. Dies erfolgt durch entsprechende Adressdekodierung. Der Flash Speicher muss so in den Adressraum eingeblendet werden, dass der Reset-Vektor des Intel XScale 80200 (Adresse 0) im Flash Speicher zu liegen kommt. Der SDRAM Speicher wird willkürlich ab Adresse 0x8000 0000 platziert. Die Abb. C.1 zeigt eine grafische Veranschaulichung des Adressraumes

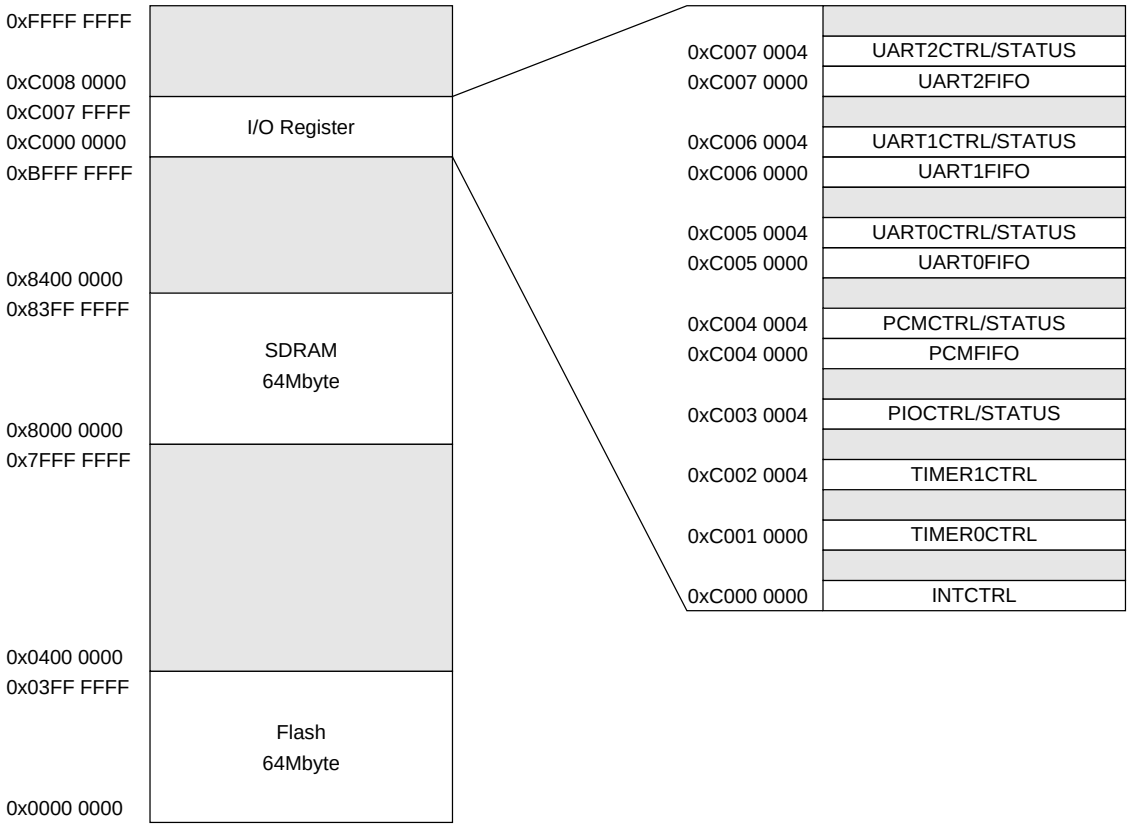


Abbildung C.1: *Adressraum des Mikroprozessormoduls*. Beginnend an der XScale Reset-Adresse 0 liegt der Falsh-Speicher, aus dem der Bootloader abgearbeitet wird. Die Register der Ein-/Ausgabemodule (UART, Timer, Interrupt Controller) sind ab der Adresse 0xC000 0000 in den Adressraum des Prozessors eingeblendet.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] *„IEEE Standard Glossary of Software Engineering Terminology“*, IEEE Standard 610.12, 1990

[2] Bernd Jähne, *„Digitale Bildverarbeitung*“, Zweite Auflage, Springer Verlag 1991

[3] Milan Sonka, Vaclav Hlavac, Roger Boyle: *„Image Processing, analysis, and Machine Vision“*, Second Edition, Brooks/Cole Publishing Company, 1999

[4] J. R. Parker, *„Algorithms for Image Processing and Computer Vision“*, John Wiley & Sons, 1997

[5] Yuichiro Anzai, *„Pattern Recognition and Machine Learning“*, Academic Press, 1992

[6] *„Verlustleistung von CPUs“*, PC Games Hardware Magazin, Februar 2002, Seite 155

[7] *„Wasserkühlung für Server“*, c't Magazin, Heft Nr. 10, Seite 34-35, Mai 2002

[8] Fred Pollack, Intel Corporation, *„New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies“*, 32nd Annual International Symposium on Microarchitecture, 16.-18. November, Haifa, 1999

[9] Srilatha Manne, Trevor Mudge, Dirk Grunwald, *„Cool Chips Tutorial - An Industrial Perspective on Low Power Processor Design“*, Proceedings of the 32 nd Annual International Symposium on Microarchitecture, 15. November 1999, Haifa, Israel.

[10] Alexander Glaser, *„Abbrandrechnungen für ein System zur Eliminierung von Waffenplutonium“*, Diplomarbeit am Institut für Kernphysik, TU Darmstadt, Februar 1998

[11] Technische Daten für das Kernkraftwerk Biblis/Deutschland, *„http://www.rhs-hannover.de/was/projekte/atomausstieg/Technik.htm“*, Stand November 2003

[12] Dionyz Pogany, Viktor Dubec, Sergey Bychikhin, Erich Gornik, *„Single-shot nanosecond thermal imaging of semiconductor devices using absorption measurements“*, IEEE Transactions on Device and Materials Reliability, Vol. 3, No. 3, September 2003, Seite 85-88

[13] Dave Jaggar, *„ARM Architecture Reference Manual“*, Prentice Hall 2000, Document Number ARM DDI 0100E

[14] D. A. Patterson, J. L. Hennessy, *„Computer Organization and Design - The Hardware / Software Interface“*, Second Edition, Morgan Kaufmann Publishers 1998

[15] J. L. Hennessy, D. A. Patterson, *„Computer Architecture - A Quantitative Approach“*, First Edition, Morgan Kaufmann Publishers 1996

[16] Ricardo Gonzalez, Mark Horowitz, *„Energy Dissipation in Gerneral Purpose Microprocessors“*, IEEE Journal of Solid State Circuits, Vol. 31, No. 9, Sept. 1996, Seite 1277-1284

[17] Reinhard Weicker, *„Dhrystone: A Synthetic Systems Programming Benchmark“*, Communications of the ACM, Vol. 27, No. 10, Oktober 1984, Seite 1013-1030

[18] Richard York, *„Benchmarking in context: Dhrystone“*, White Paper, ARM Ltd., März 2002

[19] Reinhard Weicker, *„An Overview of Common Benchmarks“*, IEEE Computer, Vol. 23, No. 12, Dezember 1990, Seite 65-75

[20] SPEC Benchmarks Homepage, *„http://www.specbench.org“*, April 2002

[21] Markus Levy, *„Motorola unveils PowerPC-performance story“*, EDN Europe, August 2002, Seite 12

[22] Embedded Microprocessor Benchmark Consortium Homepage, *„http://www.eembc.org“*, August 2002

[23] PFU Systems Homapage, *„http://www.pfusystems.com/products/overview.html“*, August 2002

[24] PFU Systems, *„RazorBlade System-On-Module“*, Datenblatt, Dokument Nr. PSI-RB7-700-FS-010, August 2002

[25] Boser Technology Hompage, *„http://www.boser.com.tw/product/cpucard/hs1600.htm“*, September 2002

[26] Boser Technology, *„HS-1600 Transmeta Crusoe Mini Board with VGA, Audio & LAN“*, Datenblatt, Dokument 15F-6 Nr. 77

[27] Keith & Koep GmbH Homepage, *„http://www.keith-koep.com“*, September 2002

[28] Keith & Koep GmbH, *„Trizeps Board“*, Datenblatt und User Manual, Version 2.0, 6. Dezember 2001

[29] Compaq Homepage, *„http://athome.compaq.com/showroom/static/iPaq/3835.asp“*, September 2002

[30] Trevor Mudge, *„Power: A First Class Architectural Design Constraint“*, IEEE Magazin Computer, April 2001, Seite 52-58

[31] Friedrich Paschke, *„Modellbildung“*, Skriptum zur Vorlesung, Technische Universität Wien, Oktober 1994

[32] Ulrich Tietze, Christoph Schenk, „Halbleiter Schaltungstechnik“, 10. Auflage, Springer-Verlag, Heidelberg, 1993

[33] Andreas Stiller, *„Prozessorgeflüster: Von Irritationen und Schlafstörungen“*, c't Magazin, Heft Nr. 19, Seite 25, September 2002

[34] Dake Liu, Christer Svensson, *„Trading Speed for Low Power by Choice of Supply and Threshold Voltages“*, IEEE Journal of Solid-State Circuits, Vol. 28, No. 1, Jänner 1993, Seite 10-17

[35] Pankaj Pant, Vivek K. De, Abhijit Chatterjee, *„Simultaneous Power Supply, Threshold Voltage and Transistor Size Optimization for Low Power Operation of CMOS Circuits“*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 6, No. 4, Dezember 1998, Seite 538-545

[36] Hyeongseok Yu, Jun-Dong Cho, *„Low-power design and architecture - In conserving energy, every bit helps“*, IEEE Potentials, August/September 2001

[37] Jerry Frenkil, *„A multi-level approach to low-power IC design“*, IEEE Spectrum, Februar 1998, Seite 54-60

[38] Jerry Frenkil, *„Tools and Methodologies for Low Power Design“*, Proceedings of the 34th Design Automation Conference, 1997, Seite 76-81

[39] Trevor Pering, Thomas Burd, Robert Brodersen, *„Voltage Scheduling in the lpARM Microprocessor System“*, Proceedings of the International Symposium on Low Power Electronics and Design, Rapallo, Italien, Juli 2000, Seite 96-101

[40] G. Schrom, S. Selberherr, *„Ultra-Low-Power CMOS Technologies“*, Invited Paper, IEEE Proceedings of the Semiconductor Conference, 1996, Vol. 1, Seite 237-246

[41] G. Schrom, *"Ultra-Low-Power CMOS Technology"*, Dissertation am Institut für Mikroelektronik, Technische Universität Wien, 1998

[42] Jim Turley, *„ARM Grabs Embedded Speed Lead“*, Microprocessor Report, Vol. 10, No. 2, Seite 1-6, Februar 1996

[43] Jim Turley, *„StrongARM Punches Up ARM Performance“*, Microprocessor Report, Vol. 9, No. 15, Seite 1-6, November 1995

[44] Jim Turley, *„SA-1100 Puts PDA on a Chip“*, Microprocessor Report, Vol. 11, No. 12, Seite 1-4, September 1997

[45] Tom R. Halfhill, *„Intel Flexes StrongARM With New Chips“*, Microdesign Ressources, Vol. 13, Seite 1, April 1999

[46] Jim Turley, *„StrongARM Speed to Triple“*, Microprocessor Report, Vo. 13, No. 6, Seite 1-4, Mai 1999

[47] L. T. Clark, E. J. Hoffman, J. Miller, M. Biyani, Y. Liao, S. Strazdus, M. Morrow, K. E. Velarde, M. A. Yarch, *„An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications“*, IEEE Journal of Solid State Circuits, Vol. 36, No. 11, Seite 1599-1608, November 2001

[48] Johnson Kin, Munish Gupta, William H. Mangione-Smith, *„Filtering Memory References to Increase Energy Efficiency“*, IEEE Transactions On Computers, Vol. 49, No. 1, Jänner 2000

[49] Tohru Ishihara, Kunihiro Asada, *„A System Level Memory Power Optimization Technique Using Multiple Supply and Threshold Voltages“*, Proceedings of the ASP-DAC 2001, Design Automation Conference, IEEE, Februar 2001

[50] Changwoo Jung, Jihong Kim, *„Instruction Cache Organisation for Embedded Lop-Power Processors“*, IEEE Electronic Letters, Vol. 37, No. 9, Seite 554-555, 26. April 2001

[51] T. Stouraitis, V. Paliouras, *„Considering the Alternatives in Low-Power Design“*, IEEE Circuits and Devices Magazine, Vol. 17, No. 4, Seite 23-29, Juli 2001

[52] Karl Pettis, Robert C. Hansen, *„Profile Guided Code Positioning“*, Proceedings of the ACM SIGPLAN 90 Conference on Programming Language Design and Implementation, Seite 16-27, New York, Juni 1990

[53] Abraham Mendlson, Sholmit S. Pinter, Ruth Shtokhamer, *„Compile Time Instruction Cache Optimizations“*, Computer Architecture News, Seite 44-51, März 1994.

[54] Trishul M. Chilimbi, Mark D. Hill, James R. Larus, *„Cache-Conscious Structure Layout“*, Proceedings of the ACM SIGPLAN 99 Conference on Programming Language Design and Implementation, Seite 1-12, Mai 1999

[55] S. Bartolini, C. A. Prete, *„A Software Strategy to Improve Cache Performance“*, IEEE Computer Society, Technical Commitee on Computer Architecture TCCA Newsletter, Jänner 2001

[56] Richard Eier, *„Digitale Systeme“*, Skriptum zur Vorlesung, Technische Universität Wien, September 1992

[57] Ahmed M. Shams, Tarek K. Darwish, Magdy A. Bayoumi, *„Performance Analysis of Low-Power 1-Bit CMOS Full Adder Cells“*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 10, No. 1, Seite 20-29, Februar 2002

[58] Keshab K. Parhi, *„Approaches to Low-Power Implementations of DSP Systems“*, IEEE Transactions on Circuits ans Systems - I: Fundamental Theory and Applications, Vol. 48, No. 10, Seite 1214-1224, Oktober 2001

[59] Amr M. Fahim, Mohamed I. Elmasry, *„Low-Power High-Performance Arithmetic Circuits and Architectures“*, IEEE Journal of Solid-State Circuits, Vol. 37, No. 1, Seite 90-94, Jänner 2002

[60] Xue-mei Zhao, Yi-zheng Ye, Ming-yan Yu, Xiao-ming Li, *„A Fast Low Power Embedded Cache Memory Design“*, IEEE Proceedings of the 4th International Conference on ASIC, 2001, Seite 566-569

[61] Robert W. Brodersen, W. R. Davis, Dennis Yee, Ning Zhang, *„Wireless Systems On a Chip Design“*, Proceedings of the International Symposium on VLSI Technology, Systems, and Applications, 2001, Seite 45-48

[62] R. Veljanovski, J. Singh, M. Faulkner, *„ASIC and DSP Implementation of Channel Filter for 3G Wireless TDD System“*, IEEE Proceedings of the 14th Annual International Conference on ASIC/SOC, 2001, Seite 47-51

[63] Luca Benini, Giovanni De Micheli, Enrico Macii, Massimo Poncino, Stefan Quer, *„Power Optimization of Core-Based Systems by Address Bus Encoding“*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 6, No. 4, Seite 544-562, Dezember 1998

[64] Matthew Farrens, Arvin Park, Gary Tyson, *„Modifying VM Hardware to Reduce Address Pin Requirements“*, IEEE Proceedings of the 25th Annual International Symposium on Microarchitecture MICRO 1992, Seite 210-213

[65] Anantha P. Chandrakasan, Robert W. Brodersen, *„Minimizing Power Consumption in Digital CMOS Circuits“*, Proceedings of the IEEE, Vol. 83, No. 4, April 1995

[66] A. J. Strojwas, M. Quarantelli, J. Borel, C. Guardiani, G. Nicollini, G. Crisenza, B. Franzini, J. Wiart, *„Manufacturability of Low Power CMOS Technology Solutions“*, Proceedings of the international Symposium on Low Power Electronics and Design, 1996, Seite 225-232

[67] Daniel P. Foty, Edward J. Nowak, *„MOSFET Technology for Low-Voltage/Low-Power Applications“*, IEEE Micro, Juni 1994, Seite 68-76

[68] M. Stockinger, *"Optimization of Ultra-Low-Power CMOS Transistors"*, Dissertation am Institut für Microelektronik, Technische Universität Wien, 2000.

[69] H. J. M. Veendrick, *„Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits“*, IEEE Journal Solid State Circuits, Vol. SC-19, August 1984, Seite 468-473

[70] The International Technology Roadmap For Semiconductors 2001, *„http://public.itrs.net/“*, Stand August 2002

[71] Pankaj Pant, Rabindra K. Roy, Abhijit Chatterjee, *„Dual-Threshold Voltage Assignment with Transistor Sizing for Low Power CMOS Circuits“*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 9, No. 2, April 2001, Seite 390-394

[72] Gerfried Zeichen, Karl Fürst, *„Automatisierte Industrieprozesse“*, Springer Verlag, Wien, September 2000

[73] Stuart F. Oberman, Hesham Al-Twaijry, Michael J. Flynn, *„The SNAP Project: Design of Floating Point Arithmetic Units“*, Proceedings of the 13th IEEE Symposium on Computer Arithmetic 1997, Seite 156-165

[74] Alexander Weinmann, *„Regelungen. Analyse und technischer Entwurf. Band 2: Multivariable, digitale und nichtlineare Regelungen; optimale und robuste Systeme“*, 3. Auflage, Springer Verlag 1995

[75] Frederick S. Hillier, Gerald J. Liebermann, *„Operations Research“*, 4. Auflage, Oldenbourg Verlag München - Wien, 1988.

[76] Philip E. Gill, Walter Murray, Margaret H. Wright, *„Practical Optimization“*, Academic Press, 7. Auflage, 1988.

[77] Josef Kallrath, *„Gemischt-ganzzahlige Optimierung“*, 1. Auflage, Vieweg Verlag Braunschweig, Wiesbaden, 2002.

[78] Intel Corporation, *„Intel PXA255 Processor, Electrical, Mechanical, and Thermal Specification“*, Document No. 278780, März 2003

[79] *„http://developer.intel.com/design/pca/applicationsprocessors/1110_brf.htm“*, Stand 8. Mai 2002

[80] *„http://www.analog.com/technology/dsp/Sharc/index.html“*, Stand 8. Mai 2002

[81] *„http://www.analog.com/technology/dsp/TigerSHARC/index.html“*, Stand 8. Mai 2002

[82] *„http://developer.intel.com/design/intelxscale/“*, Stand 8. Mai 2002

[83] *„http://www.transmeta.com/technology/index.html“*, Stand 8. Mai 2002

[84] *„http://developer.intel.com/design/mobile/pentiumiii/“*, Stand 8. Mai 2002

[85] *„http://dspvillage.ti.com/docs/catalog/dspplatform/dspplatform.jhtml?familyID=132“*, Stand 8. Mai 2002

[86] *„http://dspvillage.ti.com/docs/catalog/dspplatform/dspplatform.jhtml?familyID=114“*, Stand 8. Mai 2002

[87] *„http://www.necel.com/microprocessors/PD30700-250.cfm“*, Stand 8. Mai 2002

[88] *„http://semiconductor.hitachi.com/superh.htm“*, Stand 8. Mai 2002

[89] *„http://www.carmeldsp.com/about/documentation.html“*, Stand 8. Mai 2002

[90] *„http://www.zfmicro.com/zfx86.html“*, Stand 8. Mai 2002

[91] *„http://www.amd.com/epd/processors/4.32bitcont/14.lan5xxfam/24.lansc520/“*, Stand 8. Mai 2002

[92] *„http://www.national.com/appinfo/solutions/0,2062,396,00.html“*, Stand 8. Mai 2002

[93] *„http://e-www.motorola.com/webapp/sps/site/taxonomy.jsp?nodeId=01M0yls8rH3“*, Stand 8. Mai 2002

[94] *„http://e-www.motorola.com/webapp/sps/site/taxonomy.jsp?nodeId=01M98594“*, Stand 8. Mai 2002

[95] *„http://www.maverickaudio.com/sub.cfm?CategoryID=3&CirrusProductNumber=EP7211“*, Stand 8. Mai 2002

[96] Steve Leibson, *„XScale (StrongARM-2) Muscles In“*, Microprocessor Report, Vol. 14, Seite 1-5, September 2000

[97] Intel Corporation, *„Intel XScale Core Developer's Manual“*, Document No. 273473-001, Dezember 2000

[98] Intel Corporation, *„Intel 80200 Processor based on Intel XScale Microarchitectur, Developer's Manual“*, Document No. 273411-002, November 2000

[99] Infineon Technologies, *„128Mbit Synchronous Low-Power DRAM in Chipsize Packages, Datasheet“*, Rev. 04/01, April 2001

[100] Infineon Technologies, *„256Mbit Synchronous DRAM Datasheet“*, August 2000

[101] Samsung Electronics, *„Mobile SDRAM 8Mx16 PASR & TCSR, Preliminary Datasheet“*, Rev. 0.6, Oktober 2001

[102] Samsung Electronics, *„256Mbit SDRAM 4M x 16bit x 4Banks Synchronous DRAM LVTTL, K4S561632B Datasheet“*, Rev. 0.2, Mai 2000

[103] Elpida Memory Inc., *„256 Mbit SDRAM EDS2516APSA (16Mwords x 16bits) Datasheet“*, Document No. E0228E20, Ver. 2.0, März 2002

[104] Hitachi Ltd., *„HM5225645F-B60 256M LVTTL Interface SDRAM, 100MHz, Datasheet“*, Document No. ADE-203-1014D(Z), Rev. 2.0, November 2001

[105] Toshiba Semiconductor, *„TC59SM816/08/04BFT/BFTL-70,-75,-80 Datasheet“*, Document No. 000707EBA2, Juni 2001

[106] IBM Microdrive Homepage, *„http://www.storage.ibm.com/hdd/micro/datasheet.htm“*, Stand 10. Mai 2002

[107] CompactFlash Association Homepage, *„http://www.compactflash.org/“*, Stand 10. Mai 2002

[108] *„http://www.m-sys.com/content/products/Nandvsnor.asp?PID=16“*, Stand 10. Mai 2002

[109] Intel Corporation, *„1.8V Intel Wireless Flash Memory (W18), Preliminary Datasheet“*, Document No. 290701-003, Juni 2001

[110] Intel Corporation, *„3V Intel StrataFlash Memory, Preliminary Datasheet“*, Document No. 290667-009, August 2001

[111] AMD Inc., *„Am29BDS643D 64Mbit CMOS 1.8V Burst Mode Flash Memory, Preliminary Datasheet“*, Publication No. 23709, Rev. A, Dezember 2000

[112] Fujitsu Semiconductor, *„Flash Memory CMOS 32Mbit Page Dual Operation MBM29PDS322“*, Document No. DS05-20889-2E, Dezember 2001

[113] SHARP Electronics, *„Preliminary Datasheet 64Mbit Flash Memory Model No. LH28F640BFE-PTTL90“*, 29. August 2001

[114] Samsung Electronics, *„K9K1G08U0M 128M x 8bit NAND Flash Memory Datasheet“*, Rev. 0.2, 23. Juli 2001

[115] Toshiba Semiconductor, *„TH58100FT 1Gbit (128Mx8) CMOS NAND EEPROM Datasheet“*, 5. März 2001

[116] Toshiba Semiconductor, *„TC58FVT641 64Mbit (8Mx8) CMOS Flash Memory Datasheet“*, 6. September 2001

[117] Intel Corporation, *„Intel 1.8 Volt Wireless Flash Memory (W18) Product Brief“*, Document No. 298256-004, 2002

[118] Michael John, Sebastian Smith, *„Application-Specific Integrated Circuits“*, Addison Wesley Verlag, Juni 1997, Auszüge aus dem Buch siehe: *„http://www-ee.eng.hawaii.edu/~msmith/ASICs/HTML/ASICs.htm“*

[119] Andreas Steininger, *„VLSI-Entwurf“*, Vorlesungsskriptum, Institut für Technische Informatik, Technische Universität Wien, Jänner 2002

[120] Dave Larson, AMI Semiconductor, *„Cost Reductions to Look For by Converting FPGAs to ASICs“*, EPN - Electronic Product News, Vol. 30, No. 10, Oktober 2001

[121] H. Blume, H. Hübert, H. T. Feldkämper, T. G. Noll, *„Model-based Exploration of the Design Space for Heterogeneous Systems on Chip“*, Proceedings of the Workshop „Heterogeneous reconfigurable Systems on Chip“, Hamburg, April 2002

[122] Christian Siemers, *„Kostenfaktor - Konfigurierbare Chips versus optimierte Spezial-ICs“*, c't Magazin, Heft Nr. 10, Seite 43, Mai 2002

[123] Xilinx Inc. Homepage, *„http://www.xilinx.com/“*, Stand 21. Mai 2002

[124] Xilinx Inc., *„Xilinx Virtex-II 1.5V Field Programmable Gate Arrays Datasheet“*, Document No. DS031-1, Rev. 1.6, Juli 2001

[125] Xilinx Inc., *„Xilinx Virtex Power Estimator Worksheet“*, *„http://www.xilinx.com/support/techsup/powerest/index.htm“*, Stand 21. Mai 2002

[126] IEEE Standard 1149.1a, *„Standard Test Access Port and Boundary-Scan Architecture“*, IEEE 1993

[127] Intel Corporation, *„Intel 80200 Processor based on Intel XScale Microarchitectur, Datasheet“*, Document No. 273414-002, Oktober 2000

[128] Vishay Siliconix, *„High Frequency 600-mA Synchronous Buck/Boost Converter, Datasheet“*, Document No. S-60752, Rev. B, 5. April 1999

[129] Abhay Maheshwari, Soon-Shin Chee, *„Board Routability Guidelines with Xilinx Fine-Pitch BGA Packages“*, Xilinx Application Note: Virtex Series, XAPP157, Version 1.1, 24. April 2002

[130] GNU Hurd Projekt Homepage, *„http://www.debian.org/ports/hurd/“*, Stand 12. Mai 2002

[131] *„http://erika.sssup.it/“*, Stand 12. Mai 2002

[132] *„http://www.redhat.com/products/ecos/“*, Stand 12. Mai 2002

[133] *„http://os.inf.tu-dresden.de/fiasco/“*, Stand 12. Mai 2002

[134] *„http://www.freedos.org“*, Stand 12. Mai 2002

[135] *„http://www.linux.org“*, Stand 12. Mai 2002

[136] *„http://www.netbsd.org“*, Stand 12. Mai 2002

[137] *„http://www.oberon.ethz.ch/native/“*, Stand 12. Mai 2002

[138] *„http://www.poweros.de“*, Stand 12. Mai 2002

[139] *„http://www.qnx.com“*, Stand 12. Mai 2002

[140] *„http://www.rtems.com/index.html“*, Stand 12. Mai 2002

[141] *„http://shark.sssup.it“*, Stand 12. Mai 2002

[142] *„http://www.v2os.cx/“*, Stand 12. Mai 2002

[143] Robert Schwebel, *„Embedded Linux - Handbuch für Entwickler“*, 1. Auflage, mitp-Verlag, Bonn, 2001

[144] John Lombardo, *„Embedded Linux“*, 1. Auflage, New Riders Publishing, Juni 2001

[145] The Linux Kernel Archives, *„http://www.kernel.org/“*

[146] Dietmar Dietrich, *„Bussysteme und Rechnerkommunikation“*, Skriptum zur Vorlesung, Technische Universität Wien, März 1995

[147] MontaVista Software, Homepage, *„http://www.mvista.com/“*, Stand 10. November 2002

[148] Lineo Inc., Homepage, *„http://www.lineo.com/“*, Stand 10. November 2002

[149] Lynux Works Inc., BlueCat Linux Homepage, *„http://www.lynuxworks.com/products/bluecat/bluecat.php3“*, Stand 10. November 2002

[150] RedHat Linux 8.0 Product Homepage, *„http://www.redhat.com/software/linux/technical/“*, Stand 17. November 2002

[151] GNU C-Bibliothek Homepage, *„http://www.gnu.org/software/libc/libc.html“*, Stand 17. November 2002

[152] ucLibc Homepage, *„http://www.uclibc.org/“*, Stand 17. November 2002

[153] RedHat Inc., Newlib Homepage, *„http://sources.redhat.com/newlib/“*, Stand 17. November 2002

[154] dietlibc Homepage, *„http://www.fefe.de/dietlibc/“*, Stand 17. November 2002

[155] GNU General Public License, Deutsche Übersetzung, *„http://www.gnu.de/gpl-ger.html“*, Stand 17. November 2002

[156] GNU Lesser General Public License, Englische Fassung, *„http://www.gnu.org/copyleft/lesser.txt“*, Stand 17. November 2002

[157] Binutils Homepage, *„http://sources.redhat.com/binutils/“*, Stand 17. November 2002

[158] GNU Compiler Collection Homepage, *„http://www.gnu.org/software/gcc/gcc.html“*, Stand 17. November 2002

[159] Robert Kroiß, *„Aufbau einer Embedded Linux Plattform für die Automatisierungstechnik mit dem Intel XScale 80200 Prozessor“*, Diplomarbeit am Institut für Automatisierungs- und Regelungstechnik, Technische Universität Wien, September 2002

[160] SID Homepage, *„http://sources.redhat.com/sid/“*, Stand 20. November 2002

[161] RedHat Inc., *„SID Simulator Users Guide“*, Version vom 16. Oktober 2001

[162] RedHat Inc., *„SID Simulator Component Developer s Guide“*, Version vom 16. Oktober 2001

[163] ARM Ltd., *„ARM Architecture Reference Manual“*, Dokument Nr. ARM DDI 0100E, Juni 2000

[164] Michael Beck, Harald Böhme, Mirko Dziadzka, Ulrich Kunitz, Robert Magnus, Claus Schröter, Dirk Verworner, *„Linux Kernelprogrammierung - Algorithmen und Strukturen der Version 2.4“*, 6. Auflage, Addison-Wesley Verlag, 2001

[165] ARM Linux Homepage, Russel King, *„http://www.arm.linux.org.uk/“*, Stand 20. November 2002

[166] Alessandro Rubini, Jonathan Corbet, *„Linux Device Drivers“*, zweite Auflage, O'Reilly & Associates Inc, Juni 2001

[167] Gerhard Khüny, *„Portierung von Embedded Linux auf ein integriertes Sensorsystem“*, Diplomarbeit am Institut für Automatisierungs- und Regelungstechnik, Technische Universität Wien, September 2002

[168] Patrick Hicks, Matthew Walnock, Robert Michael Owens, *„Analysis of Power Consumption in Memory Hierarchies“*, IEEE Proceedings of the International Symposium on Low Power Design, Seite 239-242, Monterey, Kalifornien, August 1997

[169] Silicon Graphics Inc., *„http://www.sgi.com/tech/stl/“*, Standard Template Library Programmer's Guide, Stand November 2002

[170] Intel XScale IQ80310 Evaluation Toolkit Homepage, *„http://www.intel.com/design/iio/prodbref/iq80310.htm“*, Stand November 2002

[171] Kontron Embedded Computer AG Homepage, *„http://www.kontron.de“*, Stand Dezember 2002

[172] Robert Jelinek, *„Einsatz von Bluetooth-Technologie in der integrierten Sensorik“*, Diplomarbeit am Institut für Automatisierungs- und Regelungstechnik, Technische Universität Wien, September 2002

[173] Heiner Küsters, *„Bilddatenkomprimierung mit JPEG und MPEG: Stand- und Bewegtbildkomprimierung“*, Franzis Verlag, Poing 1995

[174] Dzung Tien Hoang, Jeffrey S. Vitter, *„Efficient algorithms for MPEG video compression“*, Wiley series in telecommunications and signal processing, Wiley & Sons Verlag, New York, 2002.

[175] ScanSoft Inc., Homepage, *„http://www.lhsl.com/automotive/vocon/“*, Stand Dezember 2002

[176] Paolo Remagnino, Graeme Jones, Nikos Paragios, Carlo Regazzoni, *"Video-based Surveillance Systems - Computer Vision and Distributed Processing"*, Kluwer Academic Publishers, November 2001

[177] T. Devoivre, M. Lunenborg, C. Julien, J. Carrere, P. Ferreira, W. Toren, A. VandeGoor, P. Gayet, T. Berger, O. Hinsinger, P. Vannier, Y. Trouiller, Y. Rody, P. Goirand, R. Palla, I. Thomas, R. Guyader, D. Roy, *„Validated 90nm CMOS technology platform with low-k copper interconnects for advanced system-on-chip (SoC)“* IEEE Proceedings of the International Workshop on Memory Technology, Design and Testing (MTDT), 2002.

[178] 90nm CMOS Technologie Joint Venture: STMicroelectronics, Philips und Motorola, *„http://eu.st.com/stonline/press/news/year2003/c1273h.htm“*, Stand Februar 2003

[179] Intel Corporation, *„Intel PXA255 Processor Developer's Manual“*, Document No. 278693, März 2003

[180] Anantha P. Chandrakasan, Samuel Sheng, Robert W. Brodersen, *„Low Power CMOS Digital Design“*, IEEE Journal of Solid-State Circuits, Vol. 27, Nr. 4 , April 1992, Seiten 473 -484

[181] Dave Larson, *„Cost Reduction to Look For by Converting FPGAs to ASICs“*, EPN - Electronic Product News, Vol. 30, Nr. 10, Seite 53, Oktober 2001

[182] Tiberiu Seceleanu, *„Low Power Design“*, Unterlagen zur Lehrveranstaltung *„System On Chip Design“* an der Universität Turku, Institut für Informationstechnologie, Finnland, 2002.

[183] Eddy De Greef, Francky Catthoor, Hugo De Man, *„Program Transformation Strategies for Memory Size and Power Reduction of Pseudoregular Multimedia Subsystems“*, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 8, Nr. 6, Seiten 719-733, Oktober 1998

[184] C. Kulkarni, F. Catthoor, H. De Man, *„Code Transformation for Low Power Caching in Embedded Multimedia Processors“*, Proceedings of the International Symposium on Parallel Processing IPPS, Orlando, April 1998

[185] Eddy De Greef, Francky Catthoor, Hugo De Man, *„Memory Size Reduction through Storage Order Optimization for Embedded Parallel Multimedia Applications“*, Proceedings of the International Symposium on Parallel Processing IPPS, Seiten 84-98, Genf, April 1997

[186] Preeti Ranjan Panda, Nikil D. Dutt, Alexandru Nicolau, *„Data Memory Organization and Optimizations in Application-Specific Systems“*, IEEE Design & Test of Computers, Seiten 56-68, Mai-Juni 2001

[187] Stephen Furber, James Garside, Peter Riocreux, Steven Temple, Paul Day, Jianwei Liu, Nigel Paver, *„AMULET2e: An Asynchronous Embedded Controller“*, Invited Paper, Proceedings of the IEEE, Vol. 87, Nr. 2, Februar 1999

[188] Stephen Furber, *„Takt? Nein, danke!“*, World of Embedded ARM, Sonderheft Elektronik, Seiten 14-19, Oktober 2003

[189] Realtime Linux Homepage, *„http://www.rtlinux.org“*, Stand November 2002

[190] RTAI Homepage, *„http://www.aero.polimi.it/~rtai/index.html“*, Stand November 2002

Lebenslauf

Daten zur Person

Name	Thomas Berndorfer
Geburtsdatum	27.7.1973, Grieskirchen, Oberösterreich
Akademischer Grad	Dipl.-Ing.
Familienstand	verheiratet, zwei Kinder
Ehefrau	Ulrike Berndorfer, Sonderkindergärtnerin
Wohnort	1230 Wien, Konrad Grefe Gasse 22
liebstes Hobby	Bergsteigen

Ausbildung

9/1979 - 7/1987	Volksschule, Hauptschule
10/1987 - 5/1992	HTL Elektrotechnik, Linz, Paul Hahn Str. Abschluss mit ausgezeichnetem Erfolg
10/1992 - 6/1997	Studium Elektrotechnik, Studienzweig Industrielle Elektronik und Regelungstechnik, TU Wien Abschluss in Mindeststudienzeit mit ausgezeichnetem Erfolg, Würdigungspreis des Bundesministeriums für Wissenschaft 1997
10/1997 - 3/2004	Doktoratsstudium Elektrotechnik

Berufliche Laufbahn

1991-1995	Softwareentwickler bei Fa. KEBA, Linz im Bereich industrieller Steuerungen
1996-2003	Softwareentwickler bei Fa. Siemens AG Österreich als freier Mitarbeiter im Bereich Telekommunikation und Navigationssysteme im Automobilbau
1997-2003	Forschungsassistent am Institut für Automatisierungs- und Regelungstechnik, Schwerpunkt Bildverarbeitung und Embedded Systeme
1998/1999	Präsenzdienst, Kraftfahrer und Lehrer an der Fernmeldetruppenschule Wien
2003-	Entwickler für Machine Vision Systeme bei Fa. FESTO