Kosten- und Performance-Modellierung von applikationsspezifischen VLSI-Architekturen

Von der Fakultät für Elektrotechnik und Informatik
der Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation

von

Dipl.-Ing. Hartwig Jeschke

geboren am 22.12.1960 in Kellinghusen

2005

1. Referent: Prof. Dr.-Ing. P. Pirsch

2. Referent: Prof. Dr.-Ing. E. Barke

Tag der Promotion: 3. Juni 2005

Vorwort

Diese Arbeit ist während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mikroelektronische Systeme der Universität Hannover entstanden.

Herrn Prof. Dr.-Ing. P. Pirsch danke ich für die Betreuung dieser Arbeit, die Übernahme des 1. Referates und die konstruktive Diskussion zu den Inhalten.

Herrn Prof. Dr.-Ing. E. Barke danke ich für die Übernahme des 2. Referates und für die gleichfalls konstruktive Diskussion zur vorliegenden Arbeit.

Erste Ideen zu dieser Dissertation zum Thema Modellierung entstanden während meiner Mitarbeit im Projekt *Redundante Strukturen für Signalprozessoren,* wo ich bei der Konzeption und der Realisierung eines programmierbaren Videosignalprozessors beteiligt war. Hier ergaben sich vielfältige Fragestellungen, die den Bedarf an neuen und leistungsfähigen Ansätzen für die Modellierung aufzeigten. Mein Dank gilt Herrn Prof. Dr.-Ing. H.-G. Musmann, der als Projektleiter im Projekt *Redundante Strukturen für Signalprozessoren* hervorragende Arbeitsbedingungen ermöglicht hat.

Jede Dissertation profitiert ganz besonders von einem intakten kollegialen Umfeld, das eine notwendige Voraussetzung für ein zielgerichtetes und konzentriertes Arbeiten ist. Hier möchte ich mich bei allen Kollegen bedanken, die positiv zu einem freundschaftlichen und sachgerichteten Arbeitsklima während der Bearbeitung dieser Dissertation beigetragen haben.

Herrn Dr.-Ing. H. Volkers, Herrn Dipl.-Ing. T. Wehberg und Herrn Dr.-Ing. K. Gaedke danke ich für die gute Zusammenarbeit im Projekt *Redundante Strukturen für Signalprozessoren.* Herrn Dr.-Ing. W. Schlink danke ich dafür, daß er mich bei administrativen Arbeiten entlastet hat. Herrn M.S.E.E. Mark Kulaczewski danke ich besonders für die hervorragende Zusammenarbeit rund um den *Neubau Technische Informatik*. Herrn Prof. Dr.-Ing. M. Winzker und Herrn Dipl.-Ing. H.-J. Stolberg danke ich für das Korrekturlesen der Arbeit.

Mein besonderer Dank gilt meiner Frau, meinen Eltern und meinen Töchtern für ihre Geduld und ihre Unterstützung während dieser Arbeit.

Hannover, 4. Juni 2005 Hartwig Jeschke

Inhaltsverzeichnis

Anhänge

Verwendete Abkürzungen und Formelzeichen

In dieser Arbeit wird das Dezimalzeichen als "." dargestellt.

Abkürzung/ Zeichen	Erläuterung
	Allgemeine Abkürzungen und Zeichen
ASIC	(**A***pplication* **S***pecific* **I***ntegrated* **C***ircuit)*
CCIR 601	Standard-Fernseh-Bildformat mit 720·576 Bildpunkten
CIF	Standard-Bildformat mit 352·288 Bildpunkten (**C***ommon* **I***nterface* **F***ormate*)
Core	Kernmodul
Fuzzy Set	Unscharfe Menge
GOPS	*(***G***iga* **O***perations* **P***er* **S***econd)*
IP	(**I***ntellectual* **P***roperty)*
MIPS	*(***M***ega* **I***nstructions* **P***er* **S***econd)*
MOPS	*(***M***ega* **O***perations* **P***er* **S***econd)*
MPEG	*(***M***oving* **P***ictures* **E***xpert* **G***roup)*
QCIF	Standard-Bildformat mit 176 ·144 Bildpunkten (**Q***uarter* **C***ommon* **I***nterface* **F***ormate*)
s	Strukturgröße (gezeichnete Gatterlänge)
v	Strukturverkleinerung
	Kapitel 2
ALU	(**A***rithmetic* **L***ogic* **U***nit*)
CU	(**C***ontrol* **U***nit*)
η	Effizienz als Quotient aus Performance zu Kosten
LM	(**L***ocal* **M***emory*)
PE	(**P***rozessor* **E***lement*)
PM	(**P***rogram* **M***emory*)
PU	(**P***rocessing* **U***nit*)
R/C	Granularität der Verarbeitung (**R***un-time* / **C***ommunication-time*)
VLSI	(**V***ery* **L***arge* **S***cale* **I***ntegration)*
ζ	Effizienz als gewichtete Summe aus Performance und Kosten

Abkürzung/ Zeichen	Erläuterung
	Kapitel 3
$\mu_A(x)$	Grad der Zugehörigkeit eines modellierten Parameters *X* zu einer unscharfen Menge **A**
μ_f	Grad der Erfüllung der Ziele einer Multikriterienanalyse
$[m_1,m_2,a,b]$	Fuzzy Zahl in Trapezdarstellung
	Kapitel 4
BCF	*(**B**est-**C**ase-**F**lächenmodell)*
BCMinF	*(**B**est-**C**ase-**M**inimum-**F**lächenmodell)*
DieSize	Fläche des Dies eines unverpackten Schaltkreises
$d_{Arithmetik}$	Transistordichte für arithmetisch/ logische Einheiten
$d_{Speicher}$	Transistordichte für Speicher
$d_{Norm,Arithmetik}$	Transistordichte für arithmetisch/ logische Einheiten, normiert auf 0.1µm
$d_{Norm,Speicher}$	Transistordichte für Speicher, normiert auf 0.1µm
$d_{Arithmetik,approx}$	Transistordichte für arithmetisch/ logische Einheiten, approximiert mit einer Exponentialfunktion als Funktion der Strukturgröße ***s.***
$d_{Speicher,approx}$	Transistordichte für Speicher, approximiert mit einer Exponentialfunktion als Funktion der Strukturgröße ***s.***
f_{clk}	Taktrate eines PE's
GVF	*(**G**lobaler **V**erdrahtungs**f**aktor)*
L	Anzahl der genutzten Metallisierungsebenen
MF	*(**M**ittelwert-**F**lächenmodell)*
MIF	*(**M**ittelwert-**I**ntervall-**F**lächenmodell)*
n_{bit}	Größe eines Speichers in bit
$N_{Cycles,DP}$	Anzahl der Taktzyklen pro Datenpfadoperation
$N_{Cycles,S}$	Anzahl der Taktzyklen pro skalarer Operation
N_{IO}	Anzahl der Datenzugriffe pro byte aus dem Videodatenstrom
$N_{OP,AS}$	Anzahl der skalaren Operationen und in skalare Operationen aufgelöste Datenpfadoperationen pro byte aus dem Videodatenstrom
$N_{OP,DP}$	Anzahl der Datanpfadoperationen pro byte aus dem Videodatenstrom
$N_{OP,S}$	Anzahl der skalaren Operationen pro byte aus dem Videodatenstrom
n_{Port}	Anzahl der Ports eines Speichers
R_{IO}	Datenzugriffsrate

Abkürzung/ Zeichen	Erläuterung
$R_{OP,AS}$	Rate der skalaren und als Skalaroperationen interpretierten Datenpfadoperationen
$R_{OP,DP}$	Rate der Datenpfadoperationen
$R_{OP,S}$	Rate der skalaren Operationen
R_S	Datenrate des Videodatenstromes (Source Rate)
$R_{S,Modell}$	Modellierter Gesamtdatendurchsatz
$R_{S,OpAS}$	Modellierter Gesamtdatendurchsatz für skalare Operationen und aufgelöste Datenpfadoperationen
S_M	Größe lokaler Speicher
T_{PE}	Gesamtverarbeitungszeit eines Prozessorelements
$T_{PE,DP}$	Verarbeitungszeit eines Prozessorelements für Datenpfadoperationen
$T_{PE,IO}$	Zugriffszeit eines Prozessorelements für externe Datenzugriffe
$T_{PE,Proc}$	Gesamtverarbeitungszeit eines Prozessorelements für Operationen
$T_{PE,S}$	Verarbeitungszeit eines Prozessorelements für skalare Operationen
Y	Ausbeute bei der Herstellung eines Schaltkreises *(**Y**ield)*
	Kapitel 5
E_{Mittel}	Mittlerer quadratischer, relativer Fehler
$N_{Trans,Arithmetik}$	Gesamtanzahl der Transistoren für Arithmetisch-/ Logische Einheiten
$N_{Trans,Andere}$	Anzahl der Transistoren für eine andere Verarbeitungseinheiten
$N_{Trans,DP}$	Anzahl der Transistoren für einen Datenpfad
$N_{Trans,Skalar}$	Anzahl der Transistoren für eine Skalareinheit
ϱ	Korrelationskoeffizient
$R_{S,Modell}$	Modellierter Datendurchsatz unter Berücksichtigung des Charakters der Verarbeitung (skalare Operationen und zu Datenpfadoperationen verknüpfte Einzeloperationen) und der externen Datenzugriffe.
$R_{S,OpAS,max}$	Modellierter Datendurchsatz, der sich aus dem Vergleich der Operationsraten eines Prozessors (meist in *MOPS* oder *GOPS* spezifiziert) und den Operationsanforderungen einer Applikation ergibt.

Kurzfassung

Zwischen der stark ansteigenden Komplexität integrierter Schaltkreise und der nur langsam wachsenden Produktivität der eingesetzten Entwurfswerkzeuge gibt es eine zunehmende Lücke *(Design Gap)*, die häufig als großes Problem zukünftiger Prozessorentwürfe gesehen wird. Ein Lösungsansatz zur Reduktion des *Design Gaps* liegt in der Verlagerung der Schaltungssynthese zu immer höheren Abstraktionsebenen *(High Level Synthese)*. Eine *High Level Synthese* basiert auf einer detaillierten Architekturspezifikation. Die hier notwendigen Festlegungen beinhalten das Risiko, den für einen Entwurf bestehenden Lösungsraum frühzeitig so einzuschränken, daß die bestmöglichen Schaltkreisrealisierungen nicht erreichbar sind. Im Hinblick auf dieses Problem sind Modelle wünschenswert, die auf Architekturebene eine Erkundung des möglichen Entwurfsraumes und seine zielgerichtete Fokussierung auf bestmögliche Lösungen unterstützen.

In der Literatur sind erste Ansätze veröffentlicht worden, die auf der Basis analytischer Modelle den Vergleich von alternativen Mikroprozessorarchitekturen und die Erkundung eines Entwurfsraumes unterstützen. Bei Mikroprozessoren werden Kosten und Benchmark-Ergebnisse durch die Auslegung von *Floatingpoint-Einheiten* und *Cache-Speicher* stark beeinflußt. Häufig genügt es daher, analytische Mikroprozessormodelle auf *Floatingpoint-Einheiten* und auf *Cache-Speicher* zu reduzieren. Eine heterogenere Situation ergibt sich für applikationsspezifische integrierte Schaltkreise *(ASICs)*. Im Gegensatz zu anerkannten Mikroprozessor-Benchmarks gibt es vielfältigere Anforderungen an Verarbeitungseinheiten und an die Datenführung. Entwurfsziele, Architekturansätze und VLSI-Technologien können abhängig vom Anwendungsgebiet stark variieren. Für *ASICs* werden daher neue Modelle mit stärkeren Freiheitsgraden für die individuelle Beschreibung von Applikationen, Architekturen und VLSI-Technologien benötigt.

In dieser Arbeit wird ein neuer analytischer Modellansatz für die Bewertung, den Vergleich und die Optimierung von Architekturen applikationsspezifischer Schaltkreise entwickelt. Basis ist ein dreistufiger Ansatz. Auf der ersten Ebene werden Algorithmen und Architekturen weitgehend technologieunabhängig beschrieben und im Hinblick auf Kosten (Schaltkreisgröße) und Performance in Beziehung gesetzt. Auf der zweiten Ebene folgt eine Abbildung auf eine Zieltechnologie, für die konkrete Kosten- und Performance-Werte modelliert werden. Auf der dritten Ebene werden die Zielsetzungen für einen Entwurf mit den vorangehenden Ergebnissen in Beziehung gesetzt *(Fuzzy Multikriterienanalyse)*. So können verschiedene Lösungen mit einem einheitlichen Bewertungskriterium (Grad der Zielerfüllung) in Beziehung gesetzt werden. Im Hinblick auf Unschärfen können Modellparameter als *Fuzzy Zahlen* beschrieben werden. Ein wesentliches Ergebnis ist die Verifikation des entwickelten Ansatzes an Hand realisierter *ASICs* für die Videosignalverarbeitung. Die Verifikation erfolgt aus verschiedenen Blickwinkeln, auf unterschiedlich abstrakten Modellebenen und mit Variationen in der Unschärfe der Daten. Es zeigt sich, daß stark vereinfachte Modelle im besten Fall für den Vergleich bestehender Lösungen einsetzbar sind. In der Konzeptphase der Entwicklung neuer Lösungen sind detailliertere Modelle gefordert, die geringere Fehler aufweisen müssen. Der Einsatz von *Fuzzy Zahlen* bewährt sich insofern, daß einerseits mögliche Lösungsmengen in eine Modellierung eingehen können. Andererseits bleibt die Chance auf konkrete und damit verwertbare Modellierungsergebnisse. Die Anwendung von Optimierungsverfahren zeigt, daß das entwickelte Modell auch hier zu sinnvollen Ergebnissen beiträgt.

Schlagworte zum Inhalt: Performance-Modellierung, ASIC, VLSI, Fuzzy Modellierung

Abstract

The productivity of design tools does not increase as much as the complexity of digital circuit designs increases (*Design Gap*). This *Design Gap* is frequently considered as the main problem for the design of future processors. One measure against the *Design Gap* is *High Level Synthesis*, which aims at higher levels of abstraction. A disadvantage of *High Level Synthesis* comes from the need for detailed architectural specifications, which results in the risk of losing possible, good solution sets at early architectural design steps. Because of this problem, new modeling approaches are needed, which support an sufficient design space exploration at early architectural design steps with respect to subsequent realizations of digital integrated circuits.

Recently, some new models have been published for cost and performance modeling of *microprocessors*. They focus on known *benchmark suites* as well as on *floating point units* and *cache memories*. For the design of *Application Specific Integrated Circuits (ASICs)* a more heterogeneous situation has to be considered, e. g. with specific applications, multiple design goals, multiple architectural approaches, and various different technologies for their realization. Therefore, modeling of *ASIC architectures* demands for new *cost* and *performance models*, aiming at higher flexibilities.

In this thesis a new analytical modeling approach is introduced for the evaluation, the comparison, and the optimization of *ASIC* architectures. This model is subdivided into three levels. At the first level algorithms and architectures are described independently from specific VLSI-technologies. At the second level, previous results are mapped onto a specific technology. At the third level the previous results are analyzed by a *Fuzzy Multicriteria Approach*, which results in a measure on the fulfillment of design goals. With respect to uncertainties, modeling parameters can be described as *Fuzzy Numbers*.

The most important result of this thesis is based on the verification of the proposed approach, which is based on the analysis of publishes realizations in the field of programmable video signal processors. The verification process compares known processors to their related modeling results and considers multiple sights with different levels of abstraction and variations in the degree of uncertainty of data.

In the case of highly simplified models, are only sufficient for modeling and comparison of existing architectures. At early concept phases of new processor designs more complex models are needed with much smaller error margins. They have been developed in this thesis.

The application of *Fuzzy Numbers* has been analyzed, too. They allow to consider possible solution sets, at a chance for sufficient modeling results. The application of optimization approaches show sufficient results for the new modeling approach, too.

Keywords: performance modeling, cost modeling, ASIC, VLSI, Fuzzy Multiobjective Decision Making, Fuzzy Arithmetic

1 Einleitung

Mit Einführung des ersten monolithisch integrierten Mikroprozessors durch die Firma Intel im Jahr 1971 (2300 Transistoren) begann eine revolutionäre Entwicklung, die sich eng an einem von Moore vorhergesagten exponentiellen Wachstum orientiert [63]. In Kürze wird erwartet, daß erste Mikroprozessoren die Grenze von einer Milliarde Transistoren auf einem monolithischen Schaltkreis überschreiten [9].

Der Entwurf von neuen Mikroprozessoren als auch von applikationsspezifischen *VLSI-Schaltkreisen (ASICs)* ist ein iterativer Prozeß [18]. Beginnend mit der Analyse eines Anwendungsprofiles, erfolgt eine erste Spezifikation einer Schaltkreisarchitektur. Diese wird dann in mehreren Schritten zu niedrigeren Abstraktionsebenen umgesetzt, bis die Layout-Daten eines Schaltkreises spezifiziert sind. Auf allen Abstraktionsebenen eines Entwurfes kann es passieren, daß die Verifikation der jeweiligen Ergebnisse zeigt, daß vorangehende Schritte in die falsche Richtung geführt haben. Als Konsequenz droht dann die Wiederholung zeitaufwendiger und teurer Entwurfsschritte.

Die Iterationen führen zusammen mit der technologisch bedingten Komplexitätssteigerung dazu, daß die Produktivität der Schaltkreisentwicklung nicht schritthaltend mitwächst [32]. Es ergibt sich eine wachsende Lücke *(Design Gap)* zwischen der Komplexität der Schaltkreise und der Produktivität der Schaltkreisentwicklung (Bild 1. 1).

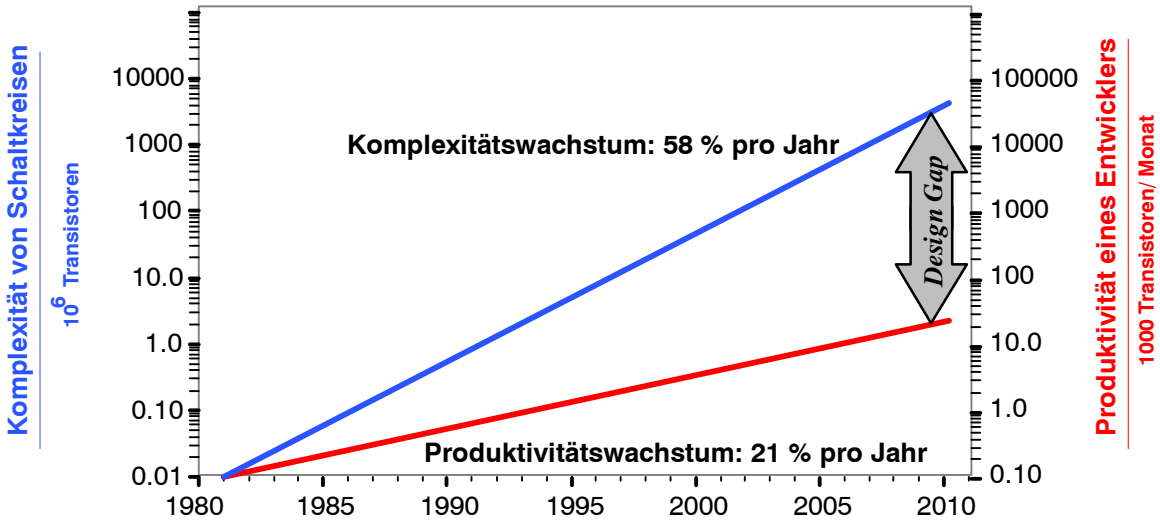


Bild 1. 1 . ***Design Gap:*** Wachsende Lücke zwischen der Komplexität realisierter VLSI Schaltkreise und der Produktivität der Schaltkreisentwicklung (Quelle: ITRS Roadmap *Design*, 1999 [32]).

In der Literatur wird das *Design Gap* häufig als zentrales Problem für den Entwurf moderner Prozessoren gesehen [8],[9],[32],[33]. Dem *Design-Gap* kann mit einzelnen Architekturansätzen begegnet werden. Bei Mikroprozessoren sind homogene Multiprozessorarchitekturen

denkbar, die monolithisch auf einem Schaltkreis realisiert werden [9]. Identische Prozessorkerne werden mehrfach eingesetzt.

Im Vergleich zu Mikroprozessoren sind applikationsspezifischen *VLSI-Schaltkreise (ASICs)* meist heterogener strukturiert. Einzelne Schaltkreismodule werden oft besonders an die Anforderungen einer Applikation angepaßt. Im Hinblick auf die Reduzierung des Entwurfsaufwandes für *ASICs* wird häufig die Wiederverwendung bereits entwickelter und verifizierter Schaltkreismodule (*IP-Cores*) vorgeschlagen (*IP-Reuse*) [32],[33]. Einmal entwickelte *Cores* werden entweder als synthetisierbare, funktionale Beschreibungen oder als fertige Layout-Module in neuen Schaltkreisentwürfen wieder verwendet.

Auf Ebene der Entwurfswerkzeuge wird als wichtigste Maßnahme zur Verbesserung der Produktivität die Verlagerung von Syntheseschritten in höhere Abstraktionsebenen vorgeschlagen [33],[78]. Es müssen insbesondere bei applikationsspezifische *VLSI-Prozessoren* zahlreiche, unterschiedliche Zielsetzungen für einen Architekturentwurf berücksichtigt werden:

- Geringe Herstellungskosten bei hohen Stückzahlen, z. B. Video-Decoder für DVD-Player.
- Geringer Stromverbrauch bei batteriebetriebenen Geräten, z. B. Mobiltelefone, PDAs, elektronische Schließanlagen.
- Hohe Rechenleistungen für rechenintensive Echtzeitanwendungen, z. B. Videosignalverarbeitung.
- Flexibilität in der Verarbeitung verschiedener Algorithmen zu unterschiedlichen Zeitpunkten, z. B. programmierbare Videosignalprozessoren.

Die Berücksichtigung der unterschiedlichen Entwurfsziele der verschiedenen Applikationen führt in der Regel zu jeweils stark an die Applikation angepaßten VLSI-Architekturen. Bisher ist kein High-Level-Synthese-Ansatz bekannt, der für größere Applikationsklassen die direkte Abbildung komplexer Applikationen auf VLSI-Architekturen ermöglicht. In der Praxis beginnen Architekturentwicklungen daher häufig auf der Ebene einer vereinfachten Systemmodellierung. Meist beschränken sich die verfügbaren Ansätze auf Teilaspekte der Entwicklung von Mikroprozessoren [7], [13], [18].

1 .1 Systemmodellierung mit analytischen Ansätzen

Bei dem Entwurf neuer *VLSI-Architekturen* beginnt eine Systemmodellierung häufig mit analytischen Modellberechnungen. Als Beispiel bieten Halbleiterhersteller zu ihren Prozeßdaten passende Berechnungsprogramme [91] oder Tabellenkalkulationen für die Bestimmung der zu erwartenden Schaltkreisgröße eines Entwurfes an, mit denen aus den Transistoranzahlen für Speicher, Arithmetik und Logik eine Schaltkreisfläche abgeschätzt werden kann. Daneben gibt es eine Vielzahl von weiteren analytischen Modellen zur Untersuchung der Performance schaltungstechnischer Einzelmaßnahmen wie *Pipelining* oder *Parallelverarbeitung* [38],

[39],[45]. In Abhängigkeit von Architekturparametern, wie Speichergrößen, Pipeline-Tiefe oder der Anzahl paralleler Einheiten können Aussagen zu Schaltkreisgrößen oder der Leistungsfähigkeit einer VLSI-Implementierung getroffen werden. Neuere Ansätze zur Modellierung von Mikroprozessoren [16], [17],[18],[57] berücksichtigen das Zusammenspiel komplexer Funktionseinheiten, wie Floatingpoint-Einheiten und Cache-Speicher. Diese Modelle zeigen, daß es möglich und sinnvoll ist, für komplexe Prozessoren Aussagen zu Gesamtkosten und -Performance einer VLSI-Implementierung über Modellbeziehungen zu gewinnen. Die Übertragbarkeit der Mikroprozessormodelle auf applikationsspezifische Schaltkreise ist insofern fraglich, weil sich der Charakter der Verarbeitung zwischen Mikroprozessoren (große Bedeutung der Floatingpoint-Verarbeitung) und *ASICs* (Floatingpoint-Verarbeitung oft nicht benötigt) stark unterscheidet. Weiterhin werden Mikroprozessoren meist mit dem Ziel höchster Taktraten entworfen, während bei *ASICs* vielfältige, abweichende Ziele bestehen können. Die direkte Übertragung bekannter Mikroprozessormodelle auf *ASICs* erscheint daher fraglich.

1 .2 Unschärfe in der Konzeptphase

Der Einsatz analytischer Ansätze für die Systemmodellierung ist vor allem in der einfachen Handhabbarkeit begründet. Generell sind aber ihre Genauigkeit und eine meist fehlende - aber indirekt vorausgesetzte - statistische Grundlage für die in ein Modell eingehenden Parameter sehr kritisch zu bewerten.

In der Regel beginnt die Konzeptphase eines neuen Schaltkreises mit der Analyse bekannter, ähnlicher Lösungen. Ein erstes Problem ist der einheitliche Vergleich von Lösungen, die in unterschiedlichen Technologien realisiert wurden.

Zusätzlich werden nicht in jedem Fall alle Realisierungskennwerte veröffentlicht. Häufig werden auch Daten angegeben, die erst mit einer kommenden Technologie erreicht werden sollen. Der Versuch, aus realisierten Lösungen Kennwerte für die analytische Modellierung einer eigenen, neuen VLSI-Architektur zu extrahieren, führt dazu, daß man zu vielen Daten Annahmen treffen muß.

Neben dem Vergleich bestehender Lösungen gibt es auch für einen konkreten Systementwurf vielfältige Unwägbarkeiten auf allen Ebenen eines VLSI-Entwurfes, die dazu führen können, daß Lösungen die aus einem analytischen Modell vorgegebenen Entwurfsziele nicht erreichen. Als Gründe können sogar organisatorische Probleme angeführt werden. Es kommt vor, daß die von einer Technologie ermöglichten Transistordichten im Einzelfall nicht erreicht werden, weil im Projektplan zu wenig Zeit oder zu wenig Entwicklungskapazität für die Layout-Ebene vorgesehen wurden.

Auf Grund der Unschärfe der eingehender Daten kann eine Architekturentwicklung zu einer falschen oder nicht optimalen Lösung führen. Im ungünstigsten Fall wird dieses erst spät im Entwurfsprozeß festgestellt.

Der generelle Einsatz statistischer Methoden zur Bewältigung der Unschärfe scheidet aus, da es oft nur geringe Stichprobenmengen zu bekannten vergleichbaren Lösungen gibt. Es wird dann kaum gelingen, eine vertrauenswürdige Statistik zu definieren.

Es gibt bei fehlender statistischer Grundlage zur Bestimmung der ***erwarteten Modelldaten*** die Alternative, ***mögliche Lösungsmengen*** der Modellparameter zu spezifizieren [100]. Im Idealfall schließen ***mögliche Lösungsmengen*** die ***erwarteten Lösungsmengen*** ein und sind gleichzeitig einfacher spezifizierbar.

Ein etablierter Ansatz zur Berücksichtigung von ***möglichen Lösungsmengen*** ist die klassische Intervallarithmetik [65], die z. B. für eine erste Abschätzung zum Entwurfsraum vor einer Schaltungssynthese eingesetzt werden kann [80]. Hier werden Lösungsmengen innerhalb minimaler und maximaler Grenzen (x_{min}, x_{max} in Bild 1. 2) charakterisiert.

Ein generelles Problem beim Einsatz der Intervallarithmetik liegt darin, daß die Intervallbreiten zunehmen, je mehr Einzelergebnisse zusammengefaßt werden. Als Ergebnis kann dann ein *möglicher Lösungsraum* so groß werden, daß keine verwertbaren Aussagen für einen Architekturentwurf getroffen werden können.

Eine interessante Lösungsstrategie, mit der eine abgestufte Zugehörigkeit von Modellparametern zu *möglichen Lösungsmengen* spezifizierbar ist, stellt die *Fuzzy Set Theorie* [100] dar. Ihre Einsetzbarkeit im Hinblick auf die Modellierung von VLSI-Architekturen wurde bisher nur ansatzweise untersucht, [46],[39], [42].

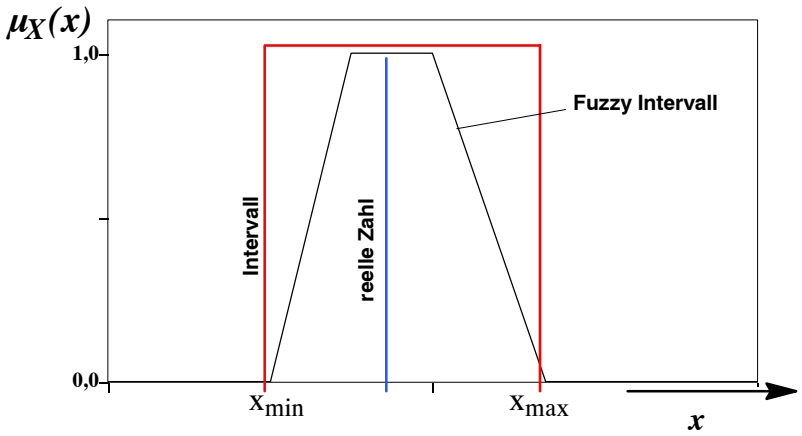


Bild 1. 2 . Unterschiedliche Repräsentation eines Modellparameters ***x*** als mögliche Lösungsmenge ***X***. $\mu_X(x)$: Zugehörigkeitsgrad von ***x*** zur Menge ***X***.

Bild 1. 2 zeigt die alternativen Repräsentationen von Parametern über den Grad der Zugehörigkeit (μ) zur *möglichen Lösungsmenge* eines Modells. Der Wert eines Modellparameters gehört sicher zu einer Lösungsmenge für den Zugehörigkeitsgrad μ=1. Für μ=0 gehört der Wert eines Modellparameters mit Sicherheit nicht zur Lösungsmenge. Dazwischen sind beliebige Abstufungen zwischen μ=0 und μ=1 möglich. Bei einer reellen Zahlendarstellung ist erkennbar, daß man indirekt voraussetzt, daß die verwendete Zahl immer sicher zur Lösungsmenge gehört (μ=1). Bei klassischer Intervallarithmetik gehören die Parameterwerte entwe-

der zur Lösungsmenge (μ=1), oder sie gehören sicher nicht dazu (μ=0). Als Alternative zwischen der reellen Zahlendarstellung und der klassischen Intervalldarstellung, können *Fuzzy-Intervalle* betrachtet werden. In Bild 1. 2 werden *Fuzzy-Intervalle* mit einer Trapezform beschrieben. Andere Darstellungsformen sind alternativ möglich.

1 .3 Ziele der Arbeit

In dieser Arbeit soll auf Basis bestehender analytischer Modellansätze und ihrer Erweiterung um Methoden der *Fuzzy Set Theorie* eine neue analytische Lösung für die Modellierung von Kosten und Performance applikationsspezifischer *VLSI*-Schaltkreise *(ASICs)* entwickelt werden. Das Ziel ist eine Lösung zu Vergleich, Bewertung und Optimierung applikationsspezifischer *VLSI*-Architekturen in frühen Entwicklungsphasen.

Im Hinblick auf das generelle Ziel dieser Arbeit muß zunächst eine Grundlage erarbeitet werden, mit deren Hilfe analytische Modelle einheitlich darstellbar sind. Im nächsten Schritt soll abgeleitet werden, welche Elemente der *Fuzzy Set Theorie* im Rahmen dieser Arbeit sinnvoll einsetzbar sind und wie sie interpretierbar sind.

Auf Basis der berücksichtigten analytischen Modelle und Methoden der *Fuzzy Set Theorie* soll ein dreistufiger, neuer Modellansatz entwickelt werden. In der ersten Ebene sollen im Hinblick auf den einheitlichen Vergleich unterschiedlicher Lösungen Algorithmen und Architekturen weitgehend technologieunabhängig beschrieben und in Beziehung gesetzt werden. In der zweiten Ebene soll die Abbildung auf eine festzulegende Zieltechnologie erfolgen. Die Abbildung auf eine Zieltechnologie soll möglichst auf einem herstellerunabhängigen Modell für die Schaltkreisgröße basieren. In der dritten Ebene folgt eine *Fuzzy-Multikriterienanalyse,* mit der ein Grad der Übereinstimmung von Modellierungsergebnissen mit Entwurfszielen analysiert wird.

Nach Einführung des neuen Modellansatzes liegt der Schwerpunkt dieser Arbeit in der Verifikation. Auf Grund besonders hoher Rechenleistungsanforderungen von Videocodierungsverfahren soll sich die Verifikation an programmierbaren Videosignalprozessoren orientieren. Hier sind Kosten- und Performance-Kriterien, Effizienzmaße und Ergebnisse aus der *Fuzzy-Multikriterienanalyse* mit bekannten, in der Literatur veröffentlichten Realisierungsdaten zu vergleichen.

Im Hinblick auf den Einsatz des Modells in frühen Konzeptphasen einer Architekturentwicklung soll untersucht werden, ob bei Einsatz unscharf charakterisierter Modellparameter brauchbare Ergebnisse erzielbar sind. Eine weitere zu berücksichtigende Fragestellung ist, ob in frühen Konzeptphasen initiale Lösungsräume durch Optimierung sinnvoll einzugrenzen sind.

1 .4 Aufbau der Arbeit

In Kapitel 2 wird zunächst der in der Literatur veröffentlichte Stand der Technik zu analytischen Architekturmodellen für Mikroprozessoren und *ASICs* diskutiert. Aus dieser Diskussion werden analytische Grundfunktionen für die nachfolgenden Untersuchungen abgeleitet.

Im Hinblick auf die Beschreibung unscharfer Modellparameter wird in Kapitel 3 die Erweiterung analytischer Modelle um *Fuzzy-Arithmetik* eingeführt. Bekannte Definitionen für die Effizienz einer VLSI-Architektur werden mit einer *Fuzzy-Multikriterienanalyse* verallgemeinert und erweitert.

In Kapitel 4 wird basierend auf den Voruntersuchungen der vorangehenden Abschnitte dieser Arbeit ein neuer Ansatz für die Modellierung von Performance (Datendurchsatz für eine Applikation) und Kosten (Schaltkreisgröße) applikationsspezifischer Schaltkreise entwickelt. Der Ansatz ist in zwei Ebenen unterteilt. In der ersten Ebene werden Algorithmen und Architekturen weitgehend technologieunabhängig charakterisiert und in Beziehung gesetzt. Die zweite Ebene unterstützt die Abbildung der vorangehenden Zwischenergebnisse auf eine Zieltechnologie. Für die Modellierung der Schaltkreisgröße wird ein neues, herstellerunabhängiges Flächenmodell eingeführt.

In Kapitel 5 wird der entwickelte Ansatz auf bekannte und veröffentlichte Lösungen aus dem Gebiet der Videosignalprozessoren angewendet und verifiziert. Es werden bekannte Referenzdaten mit Modellierungsergebnissen verglichen. Beginnend mit einer Untersuchung zum Datendurchsatz für Videocodierungsverfahren folgt die Modellierung von Schaltkreisflächen. Aus diesen Kosten- und Performance-Kriterien sollen ein direkt berechnetes Effizienzmaß (Quotient aus Datendurchsatz und Schaltkreisfläche) und die Zielerfüllung einer *Fuzzy-Multikriterienanalyse* gebildet und miteinander verglichen werden. Abschließend wird untersucht, ob der entwickelte Ansatz auch sinnvoll für die Optimierung von Architekturkonzepten eingesetzt werden kann.

In Kapitel 6 folgt eine abschließende Diskussion des entwickelten Modells und wie es sich insgesamt für einen Anwender darstellt.

In Kapitel 7 werden die Ergebnisse dieser Arbeit zusammengefaßt.

2 Stand der analytischen VLSI-Architekturmodellierung

Dieser Abschnitt soll allgemeine Begriffe aus der Architekturmodellierung einführen und Argumente dafür darstellen, daß man für die Modellierung applikationsspezifischer VLSI-Schaltkreise *(ASICs)* neue Ansätze braucht, die sich von der Modellierung von Mikroprozessoren unterscheiden (Kapitel 2 .1). Es wird ein allgemein gültiges generisches Architekturmodell eingeführt (Kapitel 2 .2), das der Ordnung und einheitlichen Betrachtung von VLSI-Architekturen dient [38]. In Kapitel 2 .3 werden am Beispiel von Parallelverarbeitung und Pipelining geschwindigkeitssteigernde Architekturmaßnahmen diskutiert. Im Abschnitt 2 .4 werden dann grundlegende analytische Modellfunktionen und Effizienzbetrachtungen diskutiert. In 2 .5 werden die Grenzen der eingeführten Modelle und der Bedarf, analytische Modelle so zu erweitern, daß unscharfe oder unsichere Modellierungsdaten mit berücksichtigt werden können, aufgezeigt.

2 .1 Mikroprozessormodelle

Die Architekturmodellierung hat eine langjährige Tradition für Prozessoren, die in Zentralrechnern oder in Arbeitsplatzrechnern eingesetzt werden. Neben Modellen, die sich mit der Leistungsfähigkeit von Architekturen auseinandersetzen *(Performance-Modellierung)* [45] gibt es einige Modelle, die auch die Kosten einer VLSI-Implementierung mit berücksichtigen [7],[20],[13][18],[81].

Als erstes Performance-Kriterium wird oft die Taktrate betrachtet [7],[20]. Alternativ werden auch Techniken angewendet, bei denen *SPEC-Benchmarkprofile* ermittelt werden. Aus dem Zusammenspiel zwischen dem Benchmarkprofil und einer Prozessorarchitektur wird auf die erreichbare Latenzzeit einer Architektur geschlossen [18]. Eine noch größere Modellierungs-Genauigkeit läßt sich dann erzielen, wenn die Latenzzeit mit Hilfe von Simulationen einer Architektur gewonnen wird [13].

Kostenmodelle für die Abschätzung werden oft aus einer Vielzahl von Parametern einzelner Halbleiterprozesse gewonnen [7],[20],[81],[18].

Generell kann man an den veröffentlichten Modellen erkennen, daß mit steigendem Aufwand einerseits die Genauigkeit der Ergebnisse wächst. Andererseits werden die Modelle komplexer und spezialisierter. Als Folge verringert sich die Anwendbarkeit einzelner Modellansätze für größere Applikationsgebiete oder für unterschiedliche Halbleiterprozesse.

Im Vergleich zwischen *ASIC*-Entwicklungen und Mikroprozessoren lassen sich Unterschiede feststellen, die gegen eine direkte Anwendung der bekannten Mikroprozessormodelle für die Entwicklung von *ASICs* sprechen. Die wichtigsten Argumente werden im Folgenden tabellarisch dargestellt.

	Mikroprozessor	**ASIC**
Entwurfsansatz	*Full Custom*	*Semi Custom*
Skalierung der Flächen bei Strukturverkleinerung um v	$v^{1.2}$ für Logik [18],[57] $v^{1.5}$ für Speicher [18],[57]	v^2 für Logik und Speicher [74][33]
Zusätzliche Metallisierungsebenen	Nutzung für Power- und Takt-Routing [9]	Nutzung für ein kompaktes Layout [12]
Entwurfsziel für Taktrate	möglichst hoch	applikationsabhängig
Performance-Kriterien	Taktrate[7],[20] Latenzzeit [13], [18]	Datendurchsatz [38],[74]
Besondere Entwurfsziele	hohe Performance akzeptable Verlustleistung	Geringe Verlustleistung bei Mobil-Applikationen

Tabelle 2.1 . Für die Architekturmodellierung relevante Unterschiede zwischen Mikroprozessoren und *ASICs*.

Während Mikroprozessoren überwiegend auf *Full Custom* Entwürfen basieren, werden *ASICs* häufig als *Semi Custom* Schaltkreise entwickelt, z. B. mit Standardzellen oder als *Gate Arrays*. Bei *Full Custom* Entwürfen haben die Schaltkreisentwickler Zugriff auf das Layout der einzelnen Transistoren und können so besonders gut kleine, in hohen Stückzahlen günstig herzustellende VLSI-Schaltkreise entwerfen. Wenn die Schaltkreisgröße nicht ganz so wichtig ist, bieten sich *Semi Custom* Entwürfe an, bei denen der Schaltungsentwurf recht gut mit Synthesewerkzeugen und mit Programmen zur automatischen Zellplazierung beherrscht werden kann. Der *Semi Custom* Ansatz besitzt durch die Automatisierung eine wesentlich höhere Produktivität und hat deshalb vor allem für *ASICs* an Bedeutung gewonnen.

Auf Grund der technologischen Weiterentwicklung mit kontinuierlich sinkenden Strukturgrößen müssen Modelle zur Abschätzung der Schaltkreisgrößen die Skalierungseigenschaften berücksichtigen [18]. In der Literatur werden unterschiedliche Skalierungen für *ASICs* und für Mikroprozessoren angegeben [74],[18]. Eine um den Faktor v sinkende Strukturgröße führt bei *ASICs* maximal zu einer quadratischen Flächenreduktion um v^2. Bei Mikroprozessoren liegen in der Praxis die Flächenskalierungsfaktoren zwischen $v^{1,2}$ und $v^{1,5}$.

Weitere große Unterschiede zwischen Mikroprozessoren und *ASICs* ergeben sich bei den Taktraten. In den letzten 10 Jahren sind die Taktraten der Mikroprozessoren um bis zu einem Faktor 100 gestiegen, während bei typischen *ASICs* für die Videosignalverarbeitung die Taktrate um einen Faktor 10 gestiegen ist. Als Konsequenz müssen bei hochgetakteten Mikroprozessoren mehrere Metalliserungsebenen für die Stromversorgung und die Taktverteilung eingesetzt werden [9]. Bei ***ASICs*** können vergleichsweise mehr Metallisierungsebenen im Hinblick auf ein kompaktes Layout eingesetzt werden [12].

Ein weiterer, signifikanter Unterschied zwischen Mikroprozessoren und *ASICs* wird durch die Applikationen beeinflußt. Bei Mikroprozessoren wird häufig eine Mischung aus Standardanwendungen *(Benchmark)* berücksichtigt, die einheitlich für alle Prozessoren angewendet wird. Neue Prozessorgenerationen müssen dann bei Erhalt der Abwärtskompatibilität eine

Geschwindigkeitssteigerung für die relevanten *Benchmarks* erbringen. Die Praxis zeigt, daß für rechenintensive Programme die Verarbeitungsleistung linear mit der Steigerung der Taktrate wächst [Anhang C]. In diesem Fall ist die Taktrate ein direkter Indikator für die Verarbeitungsleistung eines Mikroprozessors.

Im Gegensatz zu Mikroprozessoren sind bei *ASICs* allgemein anerkannte Benchmarks auf Grund der Anpassung an die jeweiligen Anforderungen der Applikationen nicht sinnvoll. Auch stellen sich oft zusätzliche Entwicklungsziele, z. B. ein möglichst geringer Stromverbrauch für mobile Anwendungen. Eine Bewertung der Performance für *ASICs* wird sich daher immer an der zu implementierenden Anwendung orientieren müssen. Auf Grund der hier geschilderten Unterschiede zwischen Mikroprozessoren und *ASICs* soll im Folgenden ein Ansatz entwickelt werden, der die spezifischen Anforderungen der *ASIC*-Entwicklung berücksichtigt.

2 .2 Ein generisches Architekturmodell für *ASICs*

Im Hinblick auf die einheitliche Charakterisierung von applikationsspezifischen *VLSI-Architekturen (ASICs)* wurde ein hierarchisches, generisches Architekturmodell entwickelt [38], das in Bild 2. 1 dargestellt wird und im weiteren Verlauf dieser Arbeit verwendet werden soll.

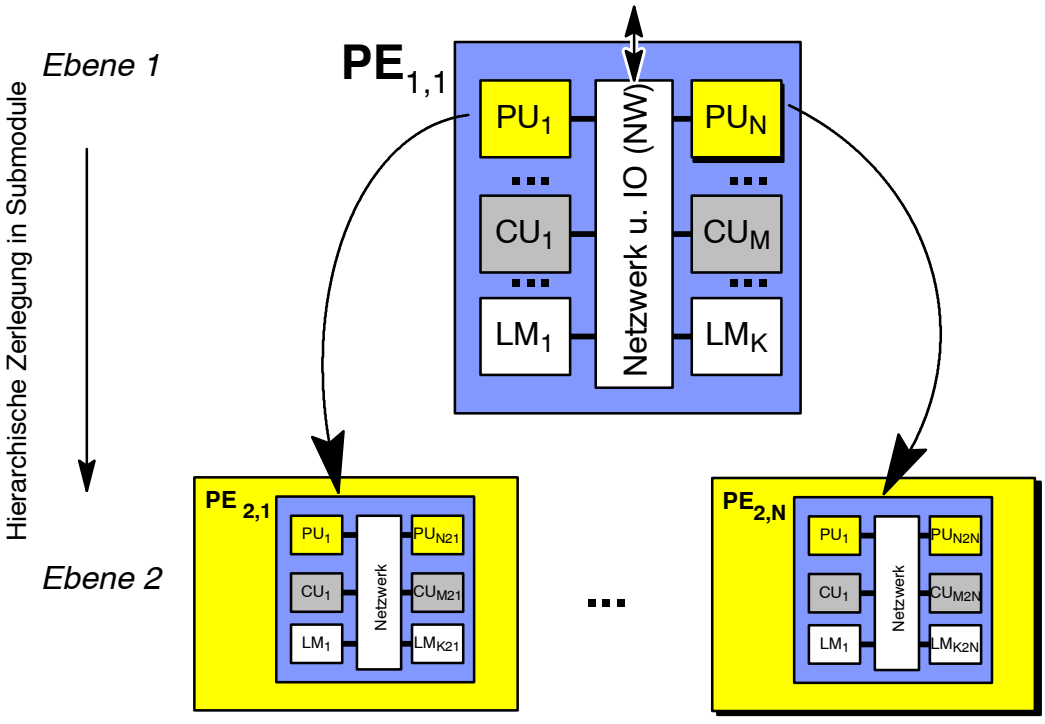


Bild 2. 1 . Ein generisches Architekturmodell (PU: Processing Unit, CU: Control Unit, LM: Local Memory, PE: Processing Element).

Ein VLSI-Schaltkreis kann auf verschiedenen Ebenen betrachtet werden [38]. Beginnend mit *Ebene 1* läßt sich eine VLSI-Architektur als Processing-Element (PE) interpretieren, das

z. B. für eine Echtzeit-Signalverarbeitungsaufgabe eingesetzt wird. Ein Processing-Element enthält in der Regel ein Netzwerk und aktive Komponenten (Netzwerk u. IO: NW) für den Datenaustausch zwischen dem Schaltkreis und der umgebenden Peripherie, sowie für die Kommunikation zwischen den internen Modulen. Lokale Datenspeicher (Local Memory: LM) bieten Entlastung für externe Datenzugriffe, indem mehrfach zu verarbeitende Datensätze schaltkreisintern zwischengespeichert werden. Wenn die Wiederholung von Datenzugriffsmustern nicht deterministisch festlegbar, aber statistisch erfaßbar ist und entsprechend genutzt wird, werden lokale Datenspeicher auch als Cache-Speicher bezeichnet. Steuerungseinheiten (Control Unit: CU) generieren Steuerungssignale für die Verarbeitungseinheiten und Speicher auf dem Schaltkreis. Hier sind verschiedene Realisierungsalternativen möglich. Bei dedizierten Architekturen kann eine Control Unit über die Ansteuerung des Schaltkreises festgelegte Steuerungssequenzen generieren. Bei programmierbaren Prozessoren kann eine Control Unit Befehlscode lesen und interpretieren.

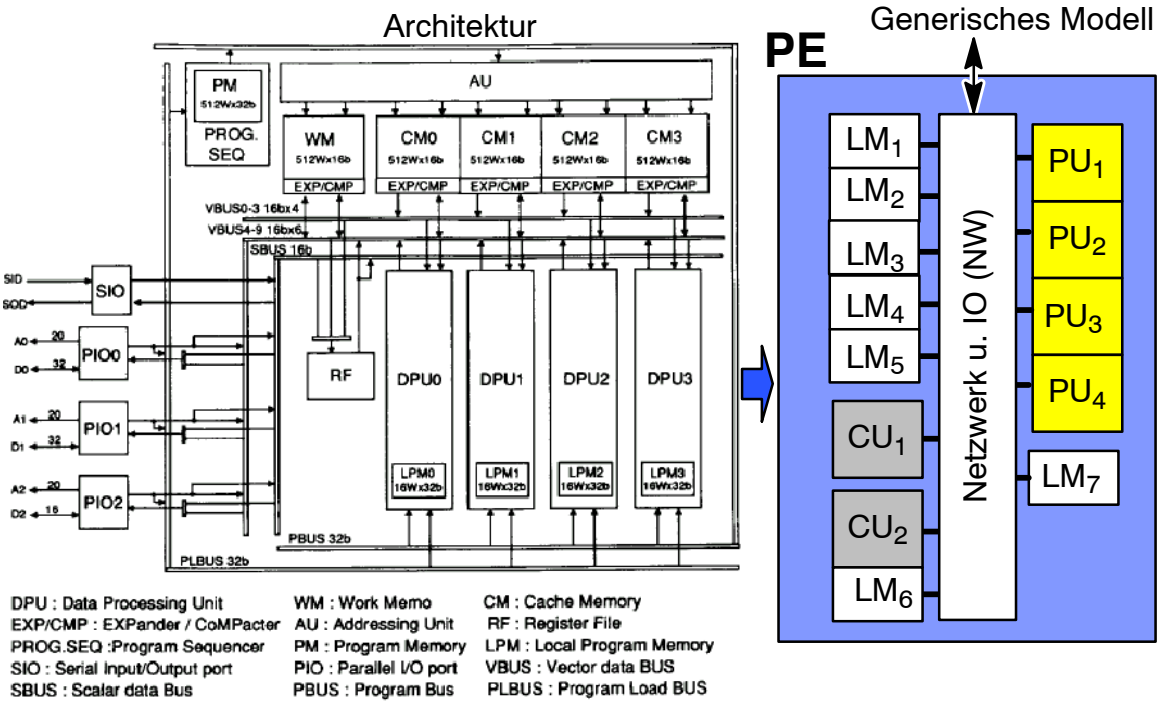


Bild 2. 2 . Architektur eines monolithischen SIMD-Prozessors für die Videocodierung (910.000 Transistoren) von 1991, Bild aus [60] und Abbildung auf das generische Modell.

Auch bei den Verarbeitungseinheiten (Processing Unit: PU) gibt es eine Vielzahl von Realisierungsmöglichkeiten. Im einfachsten Fall bestehen sie aus Datenpfadelementen, wie Addierern, Multiplizierern oder Artihmetisch Logischen Einheiten (ALU). Wenn man zuläßt, das eine Processing Unit selbst wieder ein Processing Element sein kann, ergibt sich eine hierarchische Zergliederung beliebig strukturierbarer Architekturen. Nach [38] können über die Zuweisung von Processing Units zu Control Units unterschiedliche, gängige Steuerungskon-

zepte der Parallelverarbeitung, wie *Multiple Instruction Stream Multiple Data Stream* (MIMD) oder *Single Instruction Stream Multiple Data Stream* (SIMD) erfaßt werden. Genauso ist es möglich, über die Abbildung von Datenpfadelementen auf Processing Units und durch eine feste Verschaltung der Processing Units, Pipelining Strukturen mit dem generischen Modell zu erfassen.

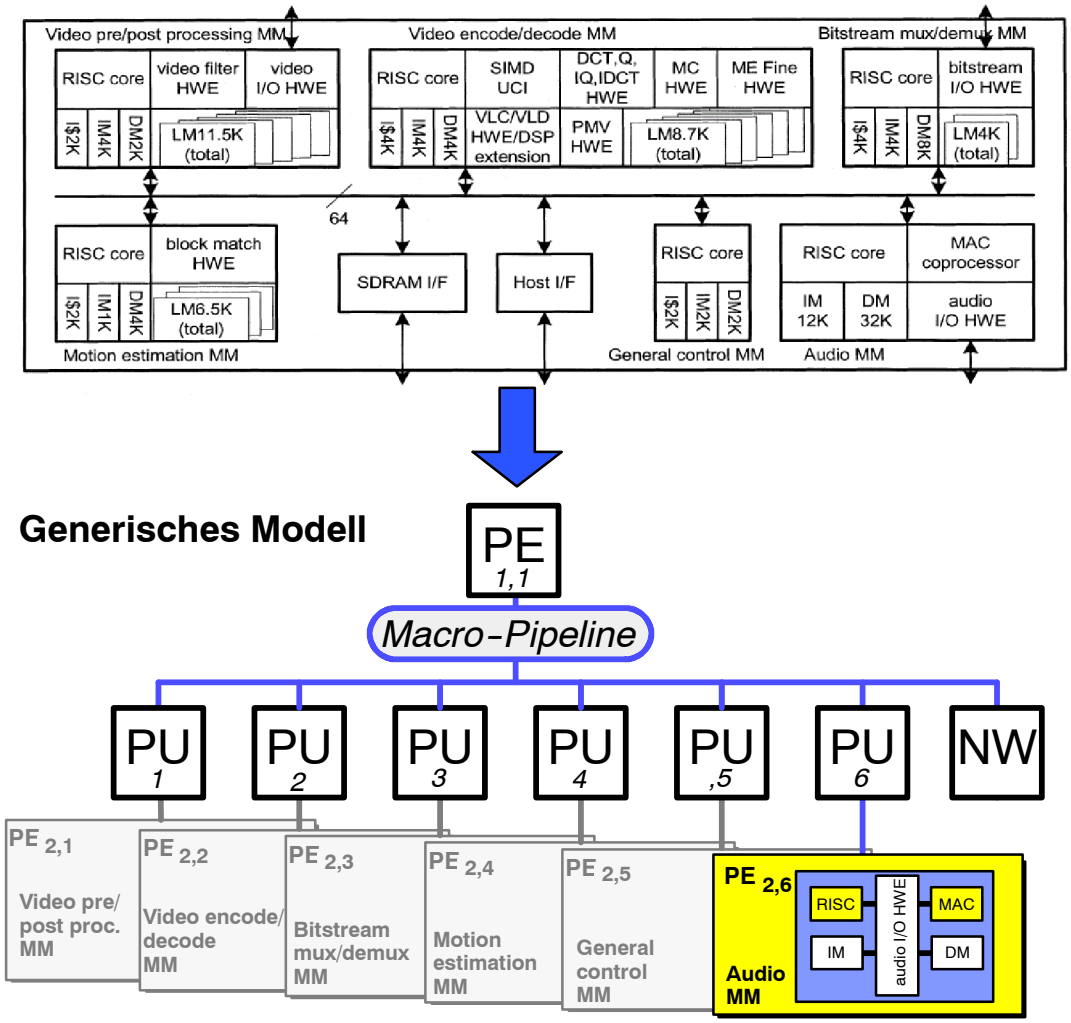


Bild 2. 3 . Architektur eines 2003 veröffentlichten monolithischen MPEG2 Video Codecs (4.000.000 Transistoren) , Bild aus [34]. Abbildung auf das generische Architekturmodell.

Die Bilder 2. 2 und 2. 3 zeigen beispielhaft, daß unterschiedliche Architekturen, die zu verschiedenen Zeitpunkten veröffentlicht wurden, auf das generische Architekturmodell abgebildet werden können.

2 .3 Pipelining, Parallelverarbeitung und Granularität

Das in 2 .2 eingeführte generische Modell bietet einen allgemeingültigen Ansatz zur hierarchischen Charakterisierung von VLSI-Architekturen und ist in erster Linie an der Schaltkreisstruktur orientiert ist. Eine erfolgreiche VLSI-Architektur erfordert eine ausbalancierte Lösung unter Berücksichtigung der Anforderungen zu verarbeitender Algorithmen.

Als Maßnahmen zur Geschwindigkeitssteigerung gegenüber einfachen sequentiellen Prozessoren werden heute in vielfältigen Variationen und Kombinationen *Pipelining* und *Parallelverarbeitung* realisiert.

2 .3 .1 Datenpfade mit Pipelining

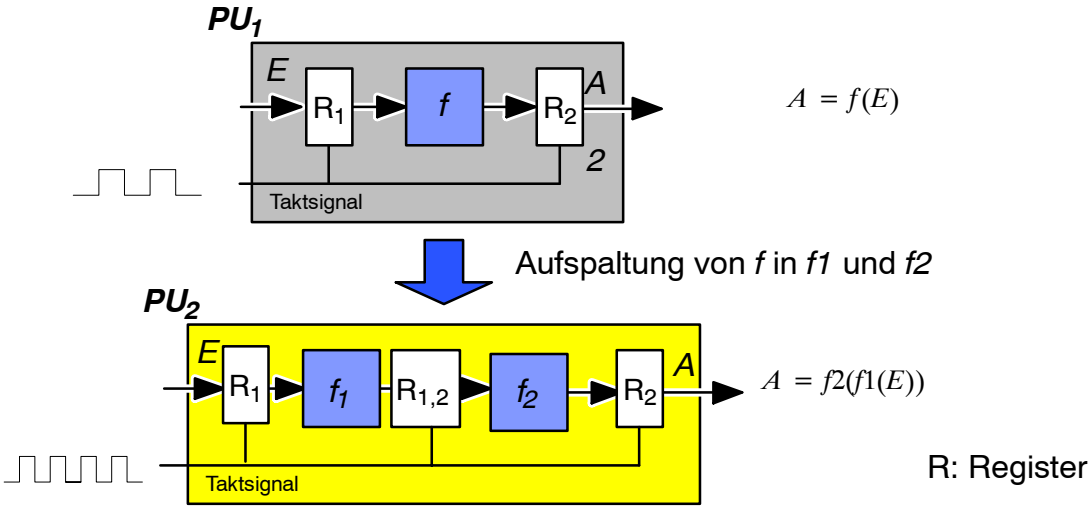


Bild 2. 4 . *Pipelining:* Erhöhung des Datendurchsatzes durch eine weitere Registerstufe.

Bild 2. 4 zeigt beispielhaft den Mechanismus des *Pipelining* als geschwindigkeitssteigernde Maßnahme. Eine *Processing Unit* PU_1 soll mit der Funktion f in jedem Taktzyklus aus dem Eingangsdatum E ein Ausgangsdatum A berechnen. Durch Aufteilung der Funktion f in zwei Teilfunktionen f_1 und f_2 mit etwa halber Verarbeitungszeit kann zwischen den Teilfunktionen ein weiteres Register $R_{1,2}$ zwischen den Funktionen f_1 und f_2 realisiert werden (PU_2). Als Folge kann die Frequenz des Taktsignales fast verdoppelt werden. Dadurch erhöht sich der Durchsatz des zu verarbeitenden Datenstromes E etwa um den Faktor 2. Nach [75], [16] kann der Datendurchsatz mit einer Unterteilung mit zusätzlichen Pipeline-Stufen weiter erhöht werden.

In Bild 2. 4 wird die Funktion f schaltungstechnisch statisch in zwei Teilfunktionen aufgeteilt. Diese Anordnung ist dann sinnvoll, wenn auf die Processing Unit PU_2 Algorithmen mit einer festen Abfolge von Teilfunktionen abgebildet werden, die zu f_1 und f_2 passen, was häufig bei Kernoperationen in der Signalverarbeitung zutrifft, z. B. bei Filteralgorithmen. Insbesondere bei Signalverarbeitungsaufgaben sind hohe *Pipelining-Tiefen* mit z. B. 12 oder 20 Register-Stufen durchaus sinnvoll implementierbar [27].

Die Verarbeitung einer größeren Menge unterschiedlicher Algorithmen mit dem selben VLSI-Schaltkreis erfordert eine flexiblere Gestaltung von Datenpfadarchitekturen. Bild 2. 5 zeigt beispielhaft eine entsprechende Datenpfadanordnung, die für Videosignalverarbeitungsanwendung vorgeschlagen wurde und als superskalarer Datenpfad bezeichnet wird [53]. Kennzeichen ist hier die dynamisch Verschaltung unterschiedlicher Verarbeitungeinheiten zu einer Pipeline per Programmanweisungen. So kann ein superskalarer Datenpfad *PU* zu unterschiedlichen Zeitpunkten durch Software gesteuert unterschiedliche Funktionen ausführen. Als weiterer Vorteil können die erforderlichen Steuersignale (Cin) recht gut aus kompilierten hochsprachlichen Programmbeschreibungen abgeleitet werden[53].

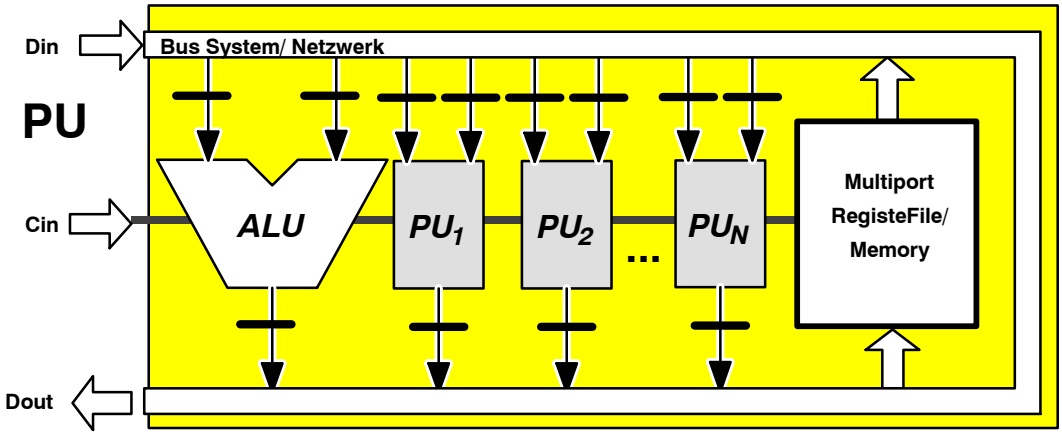


Bild 2. 5 . Superskalarer Datenpfad, nach [53]. Dynamische Verschaltung der Verarbeitungseinheiten zu Pipelines durch Programm-Code eines Prozessors.

2 .3 .2 Parallele Datenpfade

Neben dem *Pipelining* hat auch die *Parallelverarbeitung* eine große Bedeutung. In [38] wurde bei Datenpfaden für Videosignalverarbeitungsaufgaben eine steuerbare Wortbreitenumschaltung vorgeschlagen. Diese wird heute bei Standardprozessoren mit Multimedia Befehlssatzerweiterungen [73][54][11][55] und in vielen Signalverarbeitungsprozessoren in vergleichbarer Form [38][27][84][28] eingesetzt. Häufig werden hier auch die Multimediadatenpfade in Kombination mit Standard RISC Prozessoren monolithisch auf einem Schaltkreis integriert [27][84][28][34]. Bild 2. 6 zeigt beispielhaft, wie eine parallele Datenpfadarchitektur mit umschaltbaren Wortbreiten realisiert werden kann. In diesem Beispiel können entweder ein Operandenpaar *A* und *B* mit 32 bit Wortbreiten, 2 Operandenpaare mit 16 bit Wortbreiten oder mit 4 Operandenpaare mit 8 bit Wortbreiten verarbeitet werden. So kann z. B. für rechenintensive Bildverarbeitungsalgorithmen bei der Verarbeitung von benachbarten Bildpunkten eine vierfache Parallelität erreicht werden, während für weniger rechenintensive Folgeverarbeitungen eine höhere Genauigkeit mit 16 bit oder 32 bit Verarbeitung ermöglicht wird.

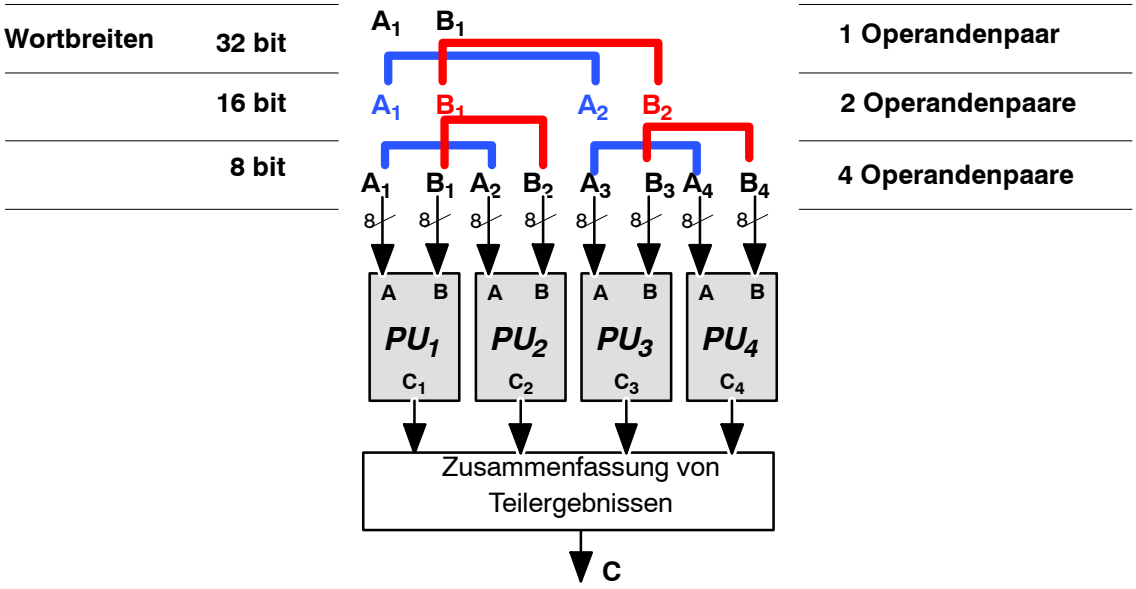


Bild 2. 6 . Beispiel: Parallele Datenpfadarchitektur, Mode-Umschaltung mit wählbaren Wortbreiten.

2 .3 .3 Parallelverarbeitung und Pipelining auf Task-Ebene

In den vorangehenden Abschnitten wurde diskutiert, wie die Verarbeitungsleistung für Datenströme aus 8 bit, 16 bit und 32 bit Worten durch *Pipelining* und *Parallelverarbeitung* in Datenpfaden gesteigert werden kann. Neben dieser wortorientierten Betrachtung, die sich an der Folge der in Datenpfaden zu verarbeitenden arithmetischen Operationen orientiert, kann man *Parallelverarbeitung* und *Pipelining* auch auf der *Task-Ebene* betrachten.

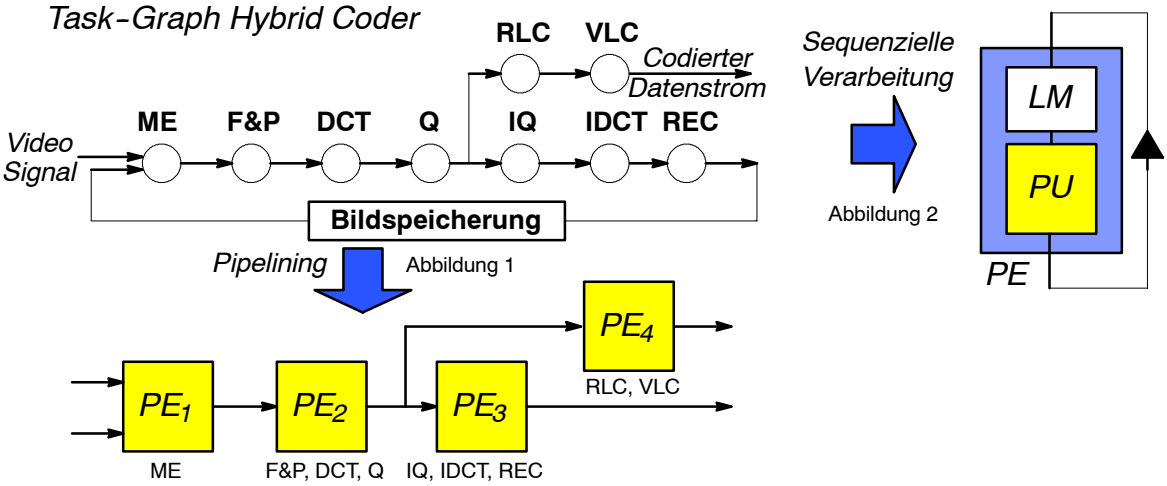


Bild 2. 7 . Darstellung eines Videocodierungsverfahrens auf Taskebene und Abbildungen auf Prozessoren [38].

Bild 2. 7 zeigt am Beispiel eines Videocodierungsverfahrens [38] grundsätzlich, wie ein komplexes Verfahren auf *Task-Ebene* abgebildet werden kann. Eine *Task* wird hier als Teilaufgabe

einer Anwendung definiert. In Bild 2. 7 wird das betrachtete Videocodierungsverfahren in die Schritte *Motion Estimation* (ME), *Filter und Prädiktor* (F&P), *Diskrete Cosinus Transformation* (DCT), *Quantisierung* (Q), *Inverse Quantisierung* (IQ), *Inverse Diskrete Cosinus Transformation* (IDCT), *Rekonstruktion* (REC), *Run Level Codierung* (RLC) und *Variable Length Codierung* (VLC) unterteilt. Die dargestellte Task-Abfolge kann weitgehend unabhängig für kleine Bildausschnitte mit 16 x 16 Bildpunkten bearbeitet werden.

Es gibt nun mehrere Möglichkeiten für die Abbildung der *Tasks* auf Prozessoren. Bei **Abbildung 1** werden einzelne oder mehrere Tasks jeweils auf ein *Processing Element (PE)* abgebildet. Mehrere *Processing Elemente* können dann auf Taskebene in einer Pipeline arbeiten, was häufig als *Macro-Pipelining* bezeichnet wird. Die Bezugsgröße für den Datenstrom der Pipeline sind dann nicht einzelne Datenworte, sondern in diesem Fall der Strom an Bildblöcken mit 16 x 16 Bildpunkten.

Bei **Abbildung 2** wird die Verarbeitung der *Tasks* einem *Processing Element* zugewiesen. In diesem Fall werden alle Tasks sequentiell verarbeitet. Wenn ein *Processing Element* die Rechenleistungsanforderungen nicht erfüllt, kann unter der Voraussetzung der unabhängigen Verarbeitung von Datenblöcken auf Datenebene parallelisiert werden.

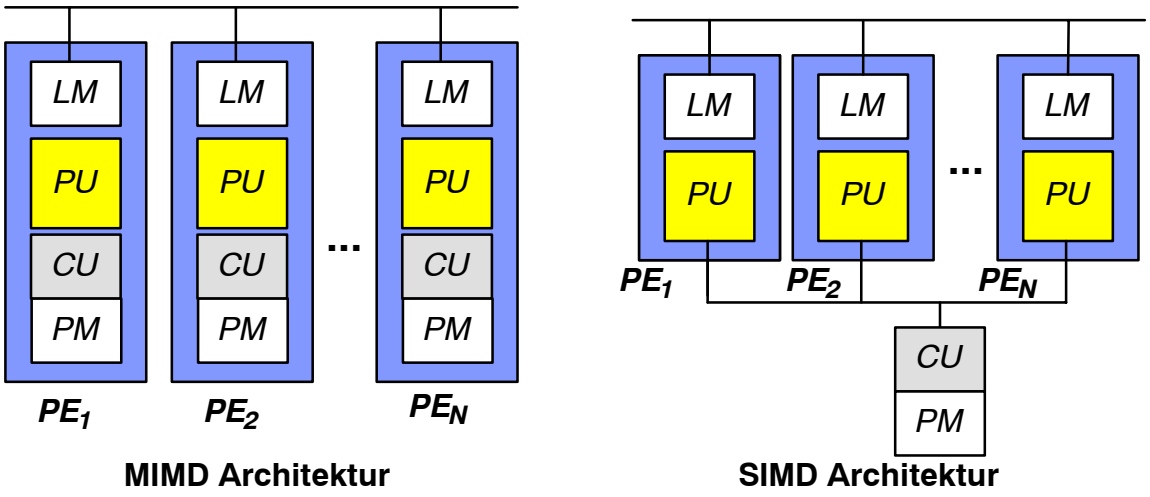


Bild 2. 8 . *MIMD* und *SIMD* Parallelarchitekturen, nach Flynn [15], *PU*: Processing Unit, *LM*: Local Memory, *CU*: Control Unit, *PM*: Program Memory).

Bild 2. 8 zeigt hier zwei mögliche Lösungen, deren Charakterisierung 1972 von Flynn [15] eingeführt wurde. Flynn betrachtete parallele Architekturen unter dem Aspekt der einheitlichen *(Single)* oder mehrfachen *(Multiple)* Ströme für Daten- und Steuerungssignale.

Hier ergeben sich grundsätzlich die folgenden Kombinationsmöglichkeiten:

- *SISD* Single Instruction Stream - Single Data Stream
- *MISD* Multiple Instruction Stream - Single Data Stream
- *MIMD* Multiple Instruction Stream - Multiple Data Stream
- *SIMD* Single Instruction Stream - Multiple Data Stream

2 .3 .4 Granularität der Verarbeitung nach Stone

In den Abschnitten 2 .3 .1 - 2 .3 .3 wurde diskutiert, daß *Pipelining* und *Parallelverarbeitung* auf verschiedenen Ebenen der Datenströme realisierbar sind, z. B. auf Ebene der arithmetischen Operationen und einzelner Datenworte oder auf *Task-Ebene* (blockorientiert bei Videocodierungsverfahren).

Stone hat im Zusammenhang mit Parallelverarbeitung ein Granularitätsmaß $\boldsymbol{R/C}$ definiert, mit dem beispielhaft bestimmt werden kann, ob es unter bestimmten Randbedingungen überhaupt sinnvoll ist, *Tasks* einer Anwendung parallel zu bearbeiten [86].

Eine Anwendung bestehe aus *N Tasks*, deren Ausführung auf zwei gleiche Prozessoren P_1 und P_2 verteilt werden sollen. Es werden *M Tasks* auf Prozessor P_1 und *N-M Tasks* auf P_2 abgebildet. Zur Vereinfachung sollen die Ausführungszeiten aller *Tasks* identisch sein und mit $\boldsymbol{R}$ bezeichnet werden. Dann soll jede *Task* einen Kommunikationsaufwand zu einer *Task* auf dem anderen Prozessor erfordern, dessen Zeit mit $\boldsymbol{C}$ bezeichnet wird.

$$T_{gesamt} = MAX\left(M \cdot R + M \cdot (N - M) \cdot C, (N - M) \cdot R + (N - M) \cdot M \cdot C\right) \tag{2.1}$$

Gleichung (2.1) beschreibt die resultierende Gesamtrechenzeit der parallel eingesetzten Prozessoren. Bild 2. 9 zeigt für $\boldsymbol{N = 100}$ *Tasks* beispielhaft den Verlauf der Gesamtrechenzeit in Abhängigkeit von $\boldsymbol{M}$ bei einer konstanten Task-Rechenzeit $\boldsymbol{R}$. Implementierungsabhängig (Einfluß der Speichergrößen und der I/O-Bandbreiten auf den Kommunikationsaufwand) soll die Kommunikationszeit $\boldsymbol{C}$ variieren.

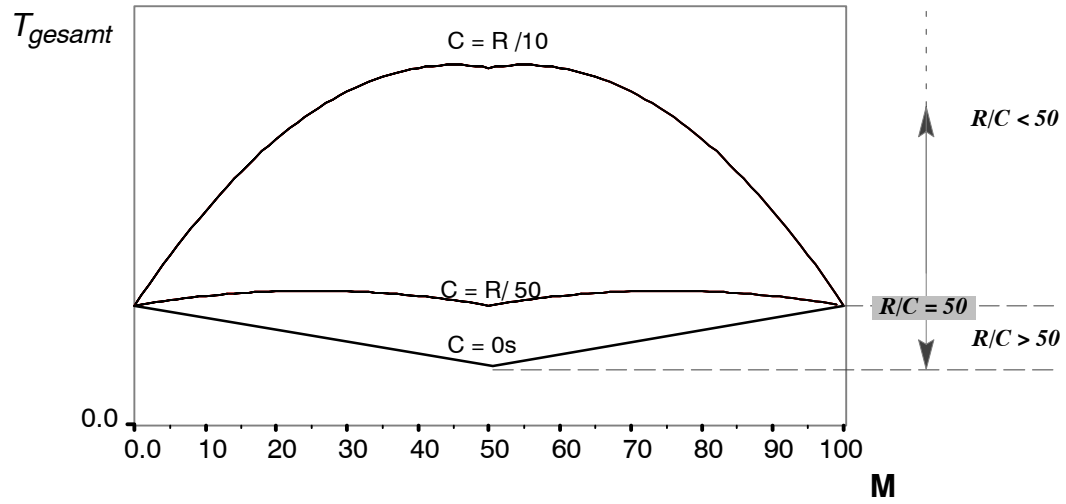


Bild 2. 9 . Beispiel für Rechenzeiten bei der Abbildung von *N=100* identischen *Tasks* auf 2 Prozessoren in Abhängigkeit von den Kommunikationsaufwendungen zwischen den Tasks auf verschiedenen Prozessoren ($\boldsymbol{C}$). *M* ist die Anzahl der *Tasks*, die einem Prozessor zugewiesen werden. $\boldsymbol{R/C}$: Granularität einer *Task*.

Für den Fall, daß alle Tasks ohne externe Kommunikationsanforderungen auf zwei Prozessoren abgebildet werden können ***(C = 0 s)***, ergibt sich die geringste Gesamtrechenzeit, wenn die *100 Tasks* gleichmäßig auf beide Prozessoren verteilt werden. Bei einem steigenden Kommunikationsaufwand ***C*** wächst die Gesamtrechenzeit T_{gesamt} für das Optimum.

Bis ***C = R / 50*** stellt die Gleichverteilung der Tasks auf beide Prozessoren die beste Lösung dar (kürzeste Gesamtrechenzeit). Darüber hinaus überwiegt für ***C > R / 50*** der Kommunikationsanteil zwischen den auf verschiedene Prozessoren abgebildeten *Tasks* so stark, daß eine Parallelverarbeitung nicht mehr sinnvoll ist.

Die von Stone definierte Granularität ***R/C*** ist eine einfache, dimensionslose Kenngröße, die es erlaubt, das Zusammenspiel zwischen Rechenzeiten und zusätzlicher Kommunikationszeiten von Tasks bei Abbildung auf Prozessoren einzuordnen. Führt das Ergebnis einer Abbildung von Tasks auf Prozessoren zu kleinen Granularitätswerten, muß besonders beachtet werden, ob eine Parallelverarbeitung überhaupt sinnvoll ist (feine Granularität). Im Beispiel in Bild 2. 9 zeigt sich für ***R/C < 50***, daß jede Abbildung auf zwei Prozessoren zu einer höheren Gesamtrechenzeit führt als die Abbildung auf einen Prozessor. Es wird zwecklos sein, mehr als einen Prozessor einzusetzen.

Aus der Modellrechnung von Stone können mehrere Schlußfolgerungen abgeleitet werden. Rechenzeiten und Zeiten für die Kommunikation zwischen Prozessoren werden immer von dem Zusammenspiel zwischen Anforderungen der zu verarbeitenden *Tasks* und den Rechenleistungen und den Kommunikationsbandbreiten der Zielarchitekturen abhängen. Das Modell zeigt, daß man mit sehr einfachen analytischen Überlegungen im Einzelfall untersuchen kann, ob Architekturansätze bei bestehenden Anforderungen der zu verarbeitenden Algorithmen überhaupt die Chance für sinnvolle Lösungen bieten. So braucht man in diesem Beispiel nicht über eine Parallelarchitektur nachzudenken, wenn sich die *Tasks* einer Anwendung nur fein granular auf eine Parallelarchitektur abbilden lassen. Zusätzlich kann das Modell bei der Konzeption eines Datenführungskonzeptes helfen. So kann es sinnvoll sein, mit Hilfe von Multiportspeichern einen Teil der Kommunikationsaufwendungen ***C*** in den Hintergrund der Bearbeitung der *Tasks* zu legen. Eine Vergrößerung der lokalen Datenspeicher und eine Erhöhung der Zugriffsraten auf externe Speicher oder eine Bandbreitenerhöhung in der Kommunikation zwischen Prozessoren können die Granularität der abgebildeten *Tasks* gleichfalls erhöhen.

Nachdem das Modell nach Stone [86] gezeigt hat, daß einfache analytische Modelle helfen können, sinnvolle Aussagen zu Architekturkonzepten zu ermitteln, soll im Folgenden näher untersucht werden, welches Potential analytische Modellansätze grundsätzlich für die Untersuchung von *VLSI*-Architekturen bieten.

2 .4 Analytische Modellfunktionen und Effizienzbetrachtungen

Nachdem ein generisches Architekturmodell (2 .2) eingeführt und Aspekte grundlegender Architekturansätze (2 .3 .1 - 2 .3 .3) diskutiert sind, soll im Folgenden untersucht werden, wie prinzipiell Aussagen zur Erfüllung der Ziele einer VLSI-Implementierung analytisch und modellgestützt ermittelt werden können.

Bei dem Entwurf eines VLSI-Schaltkreises sind in der Regel mehrere Entwurfsziele zu beachten, die im Zusammenspiel mit den zu verarbeitenden Algorithmen und über alle Hierarchieebenen des generischen Modells in Abschnitt 2 .2 unter Berücksichtigung von Kosten- und Performance-Anforderungen sinnvoll miteinander ausbalanciert werden müssen.

	Generische Funktionen	Beispiele und Referenzen
C_1	$f_{c,1} = \sum_{i=1}^{N} (c_{0,i} + c_{1,i}\, n_i)$	Schaltkreisgröße, [16], [38],[74] Kosten für Pipelineunterbrechungen [16], S. 70-73.
C_2	$f_{c,2} = \sum_{i=1}^{N} (c_{0,i} + c_{1,i}\, n_{1,i}\, n_{2,i}^{2})$	Leistungsverbrauch [77]
C_3	$f_{c,3} = c_0\, A\, e^{c_1 A}$	Fertigungskosten für ein Chip-Die [16], A: Die-Größe
P_1	$f_{p,1} = 1/\left(\frac{1}{p_0} + \frac{1}{p_1\, n_1}\right)$	Datendurchsatz als Funktion der Pipelinestufenanzahl n_1 [16] Datendurchsatz als Funktion des Parallelisierungsgrades [38] Speedup durch Parallelverarbeitung [4]
P_2	$f_{p,2} = \underset{i}{Min}(f_{p,i})$	Datendurchsatz nach [39] bei Submodulen in - Pipelines - parallelen Anordnungen
P_3	$f_{p,3} = 1/\sum_{i=1}^{N} 1/f_{p,i}$	Datendurchsatz über sequenziell arbeitende Submodule [39]

Tabelle 2.2 . Zusammenstellung wichtiger analytischen Modellfunktionen nach [39], C_1-C_3: Kostenfunktionen (cost functions), P_1-P_3: Performance-Funktionen

Tabelle 2.2 zeigt eine Auswahl der in der Literatur diskutierten analytischen Modellfunktionen. In der Praxis sind weitergehende Effekte, wie die Datenführung und die Steuerungsabläufe in einer VLSI-Architektur zusätzlich zu berücksichtigen.

2 .4 .1 Effizienz als Quotient aus Performance pro Kosten

Für die in Tabelle 2.2 dargestellten Kosten- und Performance-Funktionen läßt sich häufig zeigen, daß einzelne Parameter gleichzeitig Einfluß sowohl auf Kosten als auch auf Performance haben. Als Beispiel erhöht sich mit der Anzahl paralleler Verarbeitungseinheiten als Kostenfunktion die Schaltkreisgröße. Gleichzeitig steigt aber auch die Performance als Datendurchsatz bis ein sequentieller, nicht zu parallelisierender Anteil der Verarbeitung dominiert. Eine wichtige Fragestellung bei dem Entwurf neuer Architekturen ist die Ableitung von Architekturparametern, die den besten Kompromiß zwischen Kosten und Performance bieten. Hier läßt sich nach [39] unter der vereinfachenden Annahme, daß man bei einem Archi-

tekturkonzept Kosten, z. B. die Schaltkreisgröße, gegen Performance, z. B. den Datendurchsatz einer Echtzeitanwendung austauschen kann, ein Effizienzmaß η für die Architekturbewertung definieren.

$$\eta = \frac{f_p}{f_c} \tag{2.2}$$

Wenn in Gleichung (2.2) als Performance-Funktion der Datendurchsatz P_1 aus Tabelle 2.2 und C_1 als Kostenfunktion für die Schaltkreisgröße eingesetzt werden, ergibt sich der in Bild 2. 10 dargestellte Verlauf der Effizienz η_1 als Funktion des Parameters n_1.

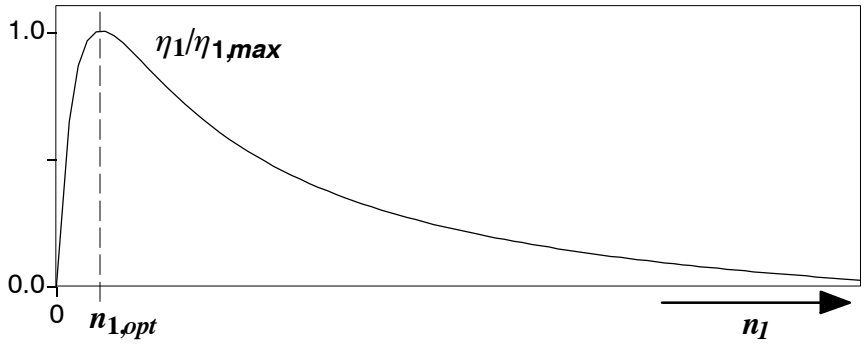


Bild 2. 10 . Auf den Maximalwert normierte Effizienz η_1 für $f_c = f_{c,1}$ und $f_p = f_{p,1}$ in Abhängigkeit des Parameters n_1, nach [39].

In diesem Fall gelingt es nach [39] mit Methoden der Differentialrechnung (Ableitung nach n_1) das Optimum für n_1 direkt zu berechnen:

$$n_{1,opt} = \sqrt{\frac{c_{0,1} p_0}{c_{1,1} p_1}} \qquad \text{für } f_c = f_{c,1} \text{ und } f_p = f_{p,1} \text{ (aus Tabelle 2.2)} \tag{2.3}$$

Das in Gleichung (2.3) dargestellte Ergebnis ist in mehreren Zusammenhängen bisher diskutiert worden. In [38] stellt $n_{1,opt}$ die optimale Anzahl der parallel eingesetzten Processing Elemente für die parallele Verarbeitung von Videocodierungsverfahren auf einem monolithischen Parallelprozessor dar. Mit $f_{c,1}$ werden die Kosten einer VLSI-Implementierung mittels der modellierten Schaltkreisfläche betrachtet. $f_{p,1}$ stellt in [38] den für ein parallel verarbeitetes Videocodierungsverfahren den erzielbaren Datendurchsatz dar. In [16] und [75] wird mit den gleichen Kosten- und Performance-Funktionen mit $n_{1,opt}$ das Optimum für die Anzahl der Pipeline-Stufen in Datenpfaden ermittelt. Während in [16] als Kostenfunktion die Unterbrechung des Datenflusses der Verarbeitung und damit verbundene Leerlaufzeiten einer Pipeline als Kosten betrachtet werden, berücksichtigt in [75] die Kostenfunktion den Anstieg der Siliziumfläche durch die Zunahme der eingesetzten Pipeline-Stufen. In [16] wird damit untersucht, bis zu welcher Anzahl der Pipeline-Stufen die Verarbeitungsleistung mit einer Zunahme des Datendurchsatzes steigt. In [75] wird untersucht, bis zu welcher Anzahl der Pipeline-Stufen der Datendurchsatz stärker steigt als die Kosten mit der benötigten Siliziumfläche auf einem Schaltkreis. In beiden Beispielen ist es dann nicht sinnvoll, mehr Pipeline-Stufen einzusetzen als durch $n_{1,opt}$ spezifiziert wird.

2 .4 .2 Alternative Effizienzbetrachtung: Lineare Kostenfunktion

Die in Abschnitt 2 .4 .1 eingeführte Effizienzbetrachtung als Quotient aus Performance pro Kosten ist nicht zwangsläufig das einzig mögliche Effizienzmaß. In [80] oder [42] werden als Zielfunktionen gewichtete Anteile einer linearen, zu minimierenden Funktion ξ definiert. Als Vorteil ergibt sich die Anwendung von verfügbaren Standardprogrammen für die lineare Optimierung. Mit $C_{1,1}$ und P_1 aus Tabelle 2.2 ergibt sich ζ als:

$$\zeta = \frac{(C_{0,1} + n_1 \cdot C_{1,1})}{w_c} + \frac{\frac{1}{p_0} + \frac{1}{p_1} \cdot \frac{1}{n_1}}{w_p} \quad \begin{array}{l} , w_c\text{: Gewichtung der Kosten} \\ , w_p\text{: Gewichtung des Performance-Anteils} \end{array} \tag{2.4}$$

Mit der ersten Ableitung nach n_1 und der zweiten Ableitung immer größer als 0 ergibt sich dann für ein optimales (minimales) ζ :

$$n_{1,opt} = \sqrt{\frac{w_c}{C_{1,1} \cdot w_p \cdot p_1}} \tag{2.5}$$

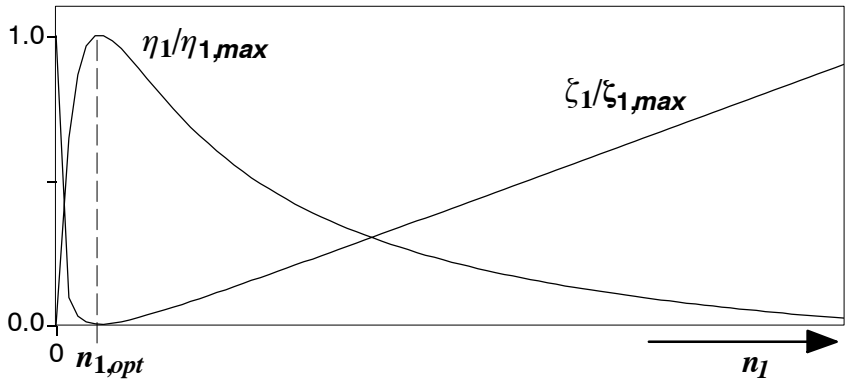


Bild 2. 11 . Beispiel für Äquivalenz zwischen der Effizienzbetrachtung als Quotient aus Performance pro Kosten und linearer gewichteter Kostenfunktion, $w_c = C_{0,1}$ und $w_p = 1/ P_0$.

Wenn man die rechten Seiten der Gleichungen (2.3) und (2.5) gleich setzt, ergibt sich im Hinblick auf den optimalen Parameter $n_{1,opt}$ eine Beziehung zwischen den Werten der Gewichte aus Gleichung (2.4) und den Kosten- und Performance-Funktionen C_1 und P_1 (Tabelle 2.2). Für spezielle Werte der Gewichte mit $w_c' = C_{0,1}$ und $w_p' = 1/ P_0$ führen sowohl lineare Zielfunktionen, wie auch die Quotientendarstellung aus Performance und Kosten zu einem identischen optimalen Architekturparameter $n_{1,opt}$. Bild 2. 11 zeigt beispielhaft für diesen Fall den Verlauf der äquivalenten Zielfunktionen. Auch wenn insbesondere bei mehreren Parametern die Bestimmung der speziellen Werte der Gewichte komplex werden kann, deutet das Ergebnis in Bild 2. 11 darauf hin, daß die unterschiedlichen, in der Literatur verfolgten Ansätze zu Zielfunktionen (η und ζ) zu vergleichbaren Modellierungsergebnissen führen.

2 .5 Bewertung analytischer Architekturmodelle

In diesem Abschnitt sollen die in 2 .3 und 2 .4 diskutierten Aspekte analytischer Modelle bewertet werden.

2 .5 .1 Vorteile der analytischen Modellierung

Ein entscheidender Vorteil der analytischen Modellierung liegt darin, daß man schon vor der vollständigen Spezifikation einer Architektur, z. B. in Form eines simulationsfähigen funktionalen Modells auf Register-Transfer-Ebene, Tendenzen für einen Entwurf ableiten kann.

Gleichung (2.3) zeigt als Beispiel, daß analytische Modelle in der Praxis recht gut handhabbar sind. Auch wenn die Bestimmung der eingehenden Konstanten im Einzelfall aufwendig sein kann, wird man bei Angabe von realistischen Größenordnungen durchaus richtige Tendenzen für den Architekturentwurf zu Pipelining und Parallelverarbeitung ableiten können oder ziemlich sicher unbrauchbare Lösungsmengen ausschließen können. Wenn z. B. abhängig von einem parallel zu verarbeitenden Algorithmus die Bestimmung von $n_{l,opt}$ auch über verschiedene Parametervariationen meist zu kleinen Zahlenwerten führt, kann man sicher sein, daß eine massive Parallelverarbeitung mit vielen Prozessoren wenig sinnvoll ist. Die Gleichungen (2.3)-(2.5) zeigen, daß im Einzelfall zwischen verschiedenen Sichtweisen zur Effizienz einer Architektur (Quotient aus Performance und Kosten gegenüber einer gewichteten Summe aus Performance und Kosten) Äquivalenzen im Hinblick auf optimale Architekturparameter bestehen können. Damit deutet sich zumindest auf einer anschaulichen Ebene an, daß die Wahl einer optimalen Effizienzfunktion weniger wichtig sein wird, als die Festlegung geeigneter Gewichte für Performance- und Kostenkriterien.

Die Granularitätsbetrachtung in 2 .3 .4 nach Stone [86] wurde in einem Zeitraum veröffentlicht, in dem massiv parallele Prozessorsysteme mit mehreren 1.000 Prozessoren untersucht wurden [72]. Häufig wurde hier - unter der Annahme, daß Prozessortaktraten jenseits der 100 MHz-Grenze nicht erwartbar sind - vorausgesetzt, daß nur besonders hohe Parallelitätsgrade der Verarbeitung die Chance zur Erfüllung der geforderten Verarbeitungszeiten bieten. Stone setzt dem auf einer sehr hohen Abstraktionsebene entgegen, daß man mit einfachen analytischen Berechnungen erkennen kann, daß eine Parallelverarbeitung im Einzelfall schon bei zwei Prozessoren sinnlos sein kann und eine massiv parallele Verarbeitung dann noch weniger sinnvoll ist. Mittlerweile werden viele der früher propagierten massiv parallelen Architekturen, z. B. Transputersysteme, durch aktuelle Prozessoren mit GHz-Taktraten übertroffen und nicht mehr weiter verfolgt.

Das Mikroprozessormodell von Fu [18],[17] hat schon bevor neue *AMD Athlon* Prozessoren am Markt verfügbar waren, vertrauenswürdige Performance-Schätzungen für *SPEC* -Benchmarks ermöglicht.

2 .5 .2 Grenzen der analytischen Modellierung

Trotz der Argumente, die für analytische Modelle zur Bewertung von VLSI-Architekturen sprechen, gibt es auch Grenzen, die im Folgenden an Hand der Begriffe *Gesamtmodelle für ASICs*, *Unschärfe der Daten* und *Grenzen der Effizienzmaße als Quotient oder Summe* diskutiert werden.

Fehlende Gesamtmodelle für ASICs

Bisher entwickelte analytische Modelle für *ASICs* beschränken sich meist auf Teiluntersuchungen, wie Pipelining, Parallelverarbeitung oder Scheduling [80]. Generelle Modellansätze, wie das von Fu entwickelte Mikroprozessormodell [18], die es erlauben, bei bestehenden Anforderungen einer komplexen Applikation Kosten- und Performance-Kriterien einer *VLSI*-Implementierung abzuschätzen, fehlen.

Das Modell nach Fu beschränkt sich auf *Floatingpoint*-Einheiten und *Caches* als primäre Einflußgrößen auf Kosten und Performance, was mehr oder weniger einheitlich für alle Mikroprozessoren anwendbar ist. Auf der Technologieseite orientiert sich das Modell von Fu auch an den Skalierungseigenschaften der für die Realisierung von Mikroprozessoren eingesetzten Halbleiterprozesse.

Bei *ASICs* ist auf Grund der vielfältigen Anwendungsgebiete und der jeweils anwendungsspezifischen Ausrichtung einer Architektur die Beschränkung auf wenige Modultypen, wie *Floatingpoint*-Einheiten und *Caches* wenig sinnvoll. Weiterhin wird in der Literatur darauf verwiesen, daß *ASIC*- und Mikroprozessor-Schaltkreisentwürfe bei dem Übergang zu einer neuen Technologie unterschiedlich skalieren (Tabelle 2.1).

Unschärfe der Daten

Verwendet man ein analytisches Modell, werden in der bisher diskutierten Form reelle Zahlen als Eingangsdaten eingesetzt. Indirekt wird damit die exakte Kenntnis der Modellkonstanten zu Realisierungsdaten vorausgesetzt, z. B. für die Transistordichten einer geplanten VLSI-Realisierung. Auf Grund vielfältiger Unwägbarkeiten des Entwurfsprozesses, können entsprechende Daten in frühen Konzeptphasen nicht exakt bekannt sein. Eine Berücksichtigung der entsprechenden Unschärfe der Daten mit Methoden der Statistik scheitert in der Regel daran, daß man nicht voraussetzen kann, genügend große Stichprobenmengen analysieren zu können, aus denen man übertragbare Modelldaten entwickeln kann.

Grenzen der Effizienzmaße als Quotient oder Summe

Die in Gleichung (2.2) definierte Effizienz als Quotient ist dann besonders sinnvoll, wenn Kosten- und Performance-Kriterien innerhalb eines monolithischen Gesamtsystemes miteinander austauschbar sind. So wird nach diesem Ansatz ein großer Prozessor-Kern genauso bewertet wie ein kleinerer Prozessor-Kern, dessen Performance proportional zu den Kosten sinkt. Im Idealfall ist dann innerhalb eines monolithisch integrierten Systemes ein großer Prozessor-Kern durch mehrere gleich effiziente kleinere Parallelprozessoren für eine Anwendung in etwa äquivalent ersetzbar.

Eine grundsätzliche Erweiterbarkeit der in den Gleichungen (2.2) und (2.4) definierten Effizienzmaße um weitere Kriterien, z. B. der Leistungsverbrauch, die Testbarkeit, die Entwurfskomplexität oder die Ausbeute, erscheint fraglich. So kann man beispielhaft auf der Kostenseite den Leistungsverbrauch als ein weiteres Kriterium hinzunehmen. Hier würde ein in der Performance leistungsfähiger Prozessor-Kern mit hohem Stromverbrauch formal als genauso effizient wie ein weniger leistungsfähiger Prozessor mit einer proportional geringeren Leistungsaufnahme bewertet. Bei einer batteriebetriebenen Anwendung kann es aber geforderte Mindestbetriebszeiträume bis zum Batteriewechsel geben, die der Prozessor-Kern mit höherer Leistungsaufnahme vielleicht nicht mehr ermöglicht. Damit können aus Anwendungssicht zwei Lösungen mit gleicher aus einem Modell bestimmten Effizienz nicht als gleichwertig betrachtet werden.

Neben der Problematik der Berücksichtigung vielfältiger Performance- und Kostenkriterien muß bei der linear gewichteten Kostenfunktion nach Gleichung (2.4) noch zusätzlich berücksichtigt werden, daß es oft sehr schwer fällt, aus den Anforderungen einer Anwendung sinnvolle Werte der Gewichte abzuleiten.

2 .5 .3 Anforderungen an eine verbesserte Modellierung

Im Zusammenhang mit den in 2 .5 .2 diskutierten Problemen wird ein Modellansatz gesucht, der die folgenden Anforderungen erfüllt:

- Integration analytischer Modellansätze, dort wo sie sich bewährt haben.
- Entwicklung neuer Modellansätze, die besonders gut an *ASICs* angepaßt sind.
- Erfassung unscharfer Realisierungsdaten trotz fehlender statistischer Grundlage.
- Erweiterung der Effizienz im Hinblick auf die Erfüllung der Implementierungsziele einer Anwendung.

Als grundlegender Lösungsansatz bei Problemen mit unscharfen Daten bietet sich nach [100] die Betrachtung möglicher Lösungsmengen an. Im Gegensatz zu statistisch erwarteten oder exakt bekannter Modelldaten sind mögliche Lösungsmengen im Einzelfall einfacher zu spezifizieren. Sie bieten die Chance, bei unscharfen Modellparametern und einer gleichzeitig fehlenden statistischen Modellbasis brauchbare Entscheidungen zu entwickeln.

Im Rahmen dieser Arbeit sollen mögliche Lösungsmengen mit Methoden der ***Fuzzy Set Theorie*** beschrieben und untersucht werden. Weiterhin soll untersucht werden, inwiefern die F*uzzy Set Theorie* sinnvolle Ansätze zur Erweiterung und Verallgemeinerung der Effizienzbetrachtungen bietet.

3 Fuzzy Modellierung

Im vorangehenden Abschnitt dieser Arbeit wurden Anforderungen an eine verbesserte Modellierung von VLSI-Architekturkonzepten entwickelt. Ein wichtiger Ansatzpunkt soll dabei die Berücksichtigung unscharfer Modellparameter über ihre Abbildung auf mögliche Lösungsmengen sein. Zusätzlich wird auch eine Erweiterung der Effizienzbetrachtung angestrebt. Es gibt in der Literatur einige wenige Vorschläge, die angesprochene Problemstellung mit Methoden der *Fuzzy Set Theorie* (Unscharfe Mengenlehre) zu lösen. So wird in [14] für die Bewertung unterschiedlicher Testkonzepte eine *Fuzzy-Multikriterienanalyse* nach Yager [99] vorgeschlagen. Die Erweiterung von analytischen Modellen mit *Fuzzy-Arithmetik* wird in [46] und [39] eingeführt. In [42] werden Kosten für auszuwählende *IP-Cores* mit *Fuzzy-Arithmetik* erfaßt. Mittels eines Ansatzes der linearen Optimierung (*Possibilistic Mixed Integer Linear Programming, PMILP*) wird dann automatisch aus einer Menge alternativer *IP Cores* die unter Kostengesichtspunkten günstigste Lösung ermittelt. Allen veröffentlichten Lösungsansätzen [14], [46],[39],[42] fehlt ein quantifizierter Nachweis, daß der Einsatz von *Fuzzy Methoden* zu einer verbesserten Modellierung führen kann.

Die *Fuzzy Set Theorie* wurde 1965 von L. A. Zadeh [101] eingeführt und stellt eine Erweiterung gegenüber der klassischen Mengenlehre dar. Während in der klassischen Mengenlehre Elemente entweder vollständig zu einer Menge gehören (Zugehörigkeitsgrad $\mu = 1$) oder vollständig nicht in einer Menge enthalten sind (Zugehörigkeitsgrad $\mu = 0$), ist der Grad der Zugehörigkeit von Elementen zu einem *Fuzzy Set* abgestuft mit einem Zugehörigkeitsgrad zwischen 0 und 1 definierbar. Formal kann nach [100] ein *Fuzzy Set X* auf eine Menge $\boldsymbol{A}$ durch Wertepaare $(x, \mu_A(x))$ charakterisiert werden:

$$\boldsymbol{X} \mapsto \boldsymbol{A} : \boldsymbol{A} = \left\{(x, \mu_{\boldsymbol{A}}(x) \,|\, x \in \boldsymbol{X}\right\} \tag{3.1}$$

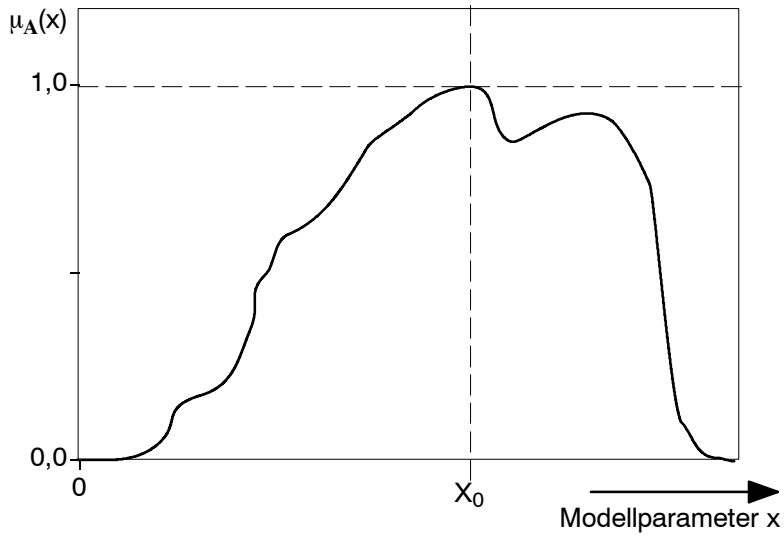


Bild 3. 1 . Beispiel für eine Zugehörigkeitsfunktion $\mu_A(x)$ für einen Modellparameter X zur unscharfen Menge $\boldsymbol{A}$.

Bild 3. 1 zeigt beispielhaft eine Zugehörigkeitsfunktion mit einem willkürlich festgelegten Verlauf. In diesem Beispiel könnte die unscharfe Menge bezeichnet werden als *etwa* X_0.

Auch wenn der Verlauf einer Zugehörigkeitsfunktion prinzipiell beliebig festlegbar ist, bietet es sich nach [43], S. 83-86, häufig an, über wenige Parameter beschreibbare Zugehörigkeitsfunktionen zu wählen, z. B. eine Trapezform. Damit kann der Rechenaufwand für die Verarbeitung unscharfer Mengen deutlich verringert werden. Ein in Trapezform beschriebener *Fuzzy Set A* wird dann eindeutig gekennzeichnet mit:

$$A = \left[m_1, m_2, a, b\right]$$

$$\text{mit:} \quad \mu_A(x) = \begin{cases} 0 & \text{, für} \quad x \le m_1 - a \\ \frac{1}{a}(x - m_1 + a) & \text{, für} \quad m_1 - a < x < m_1 \\ 1 & \text{, für} \quad m_1 \le x \le m_2 \\ 1 - \frac{1}{b}(x - m_2) & \text{, für} \quad x \ge m_2 + b \\ 0 & \text{, für} \quad m_2 < x < m_2 + b \end{cases} \tag{3.2}$$

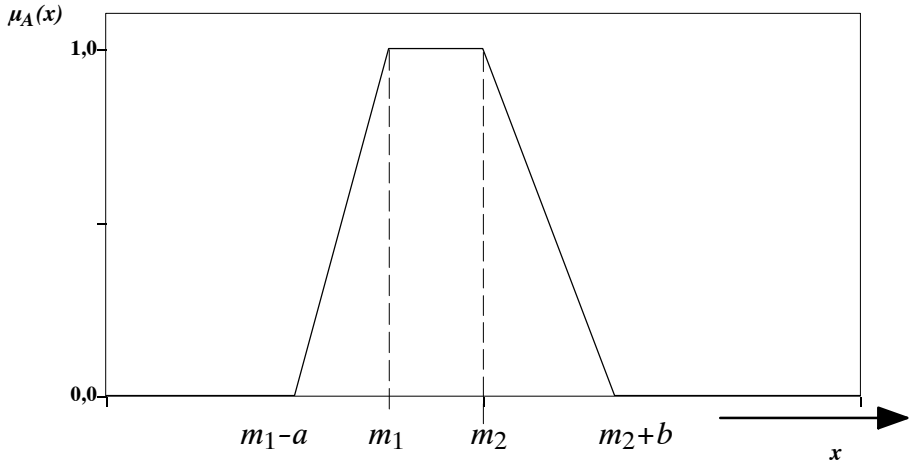


Bild 3. 2 . *Fuzzy Set:* Trapez-Darstellung einer Zugehörigkeitsfunktion.

3 .1 Charakterisierung möglicher Lösungsmengen

Grundsätzlich ist in der *Fuzzy Set Theorie* nicht festgelegt, wie der Zugehörigkeitsgrad eines *Fuzzy Sets* interpretiert werden muß. Die Interpretation als mögliche Lösungsmenge (*englisch:* possibility theory) im Vergleich mit einer erwarteten Lösungsmenge (*englisch:* probability theory), wird von Zimmermann in [100], S. 113-114 diskutiert (*Probability* versus *Possibility*).

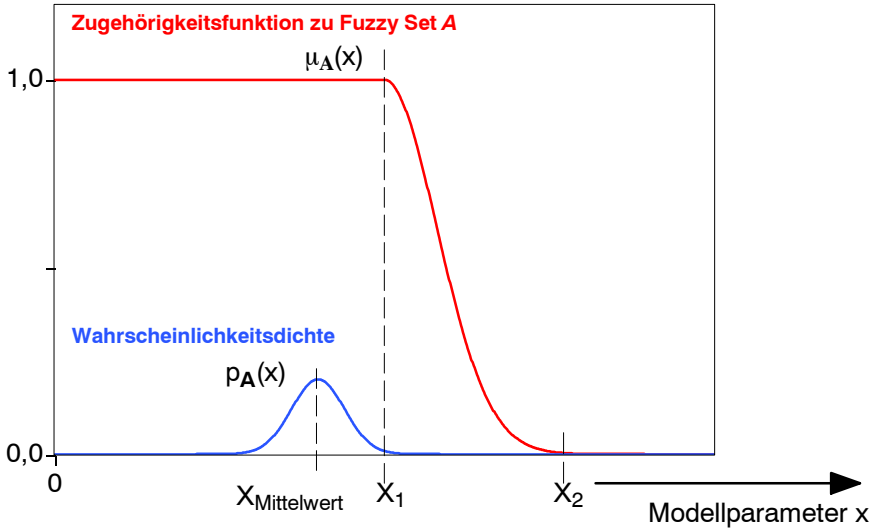


Bild 3. 3 . Eine mögliche Lösungsmenge (*Fuzzy Set* ***A***) schließt eine erwartbare Lösung ein.

Wenn man beispielhaft einen kontinuierlichen Parameter ***x*** über einen langen Zeitraum beobachtet und damit eine genügend große, statistisch auswertbare Stichprobenmenge erhält, kann man eine Wahrscheinlichkeitsdichte $p_A(x)$ bestimmen (Bild 3. 3). Alternativ kann aber auch eine mögliche Lösungsmenge über einen *Fuzzy Set* ***A*** definiert werden, die ohne statistische Grundlage, z. B. aus anschaulichen Überlegungen abgeleitet wird (Zugehörigkeitsfunktion $\mu_A(x)$ in Bild 3. 3). Dieser Zusammenhang soll im Folgenden an Hand eines einfachen Beispieles diskutiert werden.

Beispiel: Erwartete und mögliche Lösungsmengen

x sei die Geschwindigkeit eines Fahrzeuges im Stadtverkehr. Wenn das Fahrzeug einen Bordcomputer besitzt, kann man über einen längeren Meßzeitraum die Geschwindigkeit messen und eine Durchschnittsgeschwindigkeit ermitteln, z. B. $X_{Mittelwert}$ mit 31,2547 km/h. Alternativ kann man ohne vorliegende Messungen eine mögliche Lösungsmenge definieren. Man kann sicher sein, daß ein Fahrzeug im Stadtverkehr zwischen 0 und 50 km/ h fahren wird (μ_A = 1, bis X_1). Auf Grund der gesetzlichen Bestimmungen wird man genauso sicher sein können, daß in der Praxis bestimmt nicht schneller als mit 60 km/h gefahren wird (μ_A = 0, ab X_2). Zwischen 50 und 60 km/h gibt es dann vielleicht einen Übergang für μ_A von 1 auf 0. Der über einen *Fuzzy Set* von ***X*** auf ***A*** definierte, mögliche Lösungsraum in Bild 3. 3 schließt die erwartete, über eine Statistik absicherbare Lösung mit ein. Wenn jetzt das betrachtete Fahrzeug keinen Bordcomputer hat und falls niemals zuvor vergleichbare Statistiken mit anderen Fahrzeugen ermittelt wurden, könnte man bei diesem Modellierungsproblem trotz fehlender statistischer Grundlage wenigstens auf einen möglichen Lösungsraum zurückgreifen, der aus anschaulichen Überlegungen gewonnen wird.

Vergleichbar zu dem diskutierten sehr allgemeinen Beispiel kann man die Modellierung von VLSI-Architekturen auf Konzeptebene betrachten. Wenn man für ein neues Anwendungsgebiet oder mit einem neuen Halbleiterprozeß eine optimierte VLSI-Architektur entwickelt,

gibt es vielleicht nur sehr wenige, vergleichbare Realisierungsbeispiele. Es kann daher weder mit statistisch abgesicherten Daten gerechnet werden, noch können exakte Zahlenwerte für die Realisierungsdaten einer VLSI-Implementierung angegeben werden.

Auf Grund dieser Überlegungen sollen im Zusammenhang mit der Modellierung von VLSI-Architekturkonzepten im Folgenden mögliche Lösungsmengen mit *Fuzzy Sets* charakterisiert werden.

3 .2 Erweiterung analytischer Modelle mit *Fuzzy-Arithmetik*

Nachdem *Fuzzy Sets* für die Charakterisierung möglicher Lösungsmengen eingeführt worden sind, soll im Folgenden die Erweiterung analytischer Modelle um *Fuzzy-Arithmetik* diskutiert werden.

3 .2 .1 Rechnen mit *Fuzzy Sets* in Trapezform: Das Erweiterungsprinzip

Nach Zimmermann [100] und Klir [43] können *Fuzzy Sets* auch als Zahlen interpretiert werden (*Fuzzy Numbers*). Mit Hilfe des Erweiterungsprinzips nach [100] können dann arithmetische, für reelle Zahlen definierte Operationen für die Anwendung auf *Fuzzy Numbers* erweitert werden.

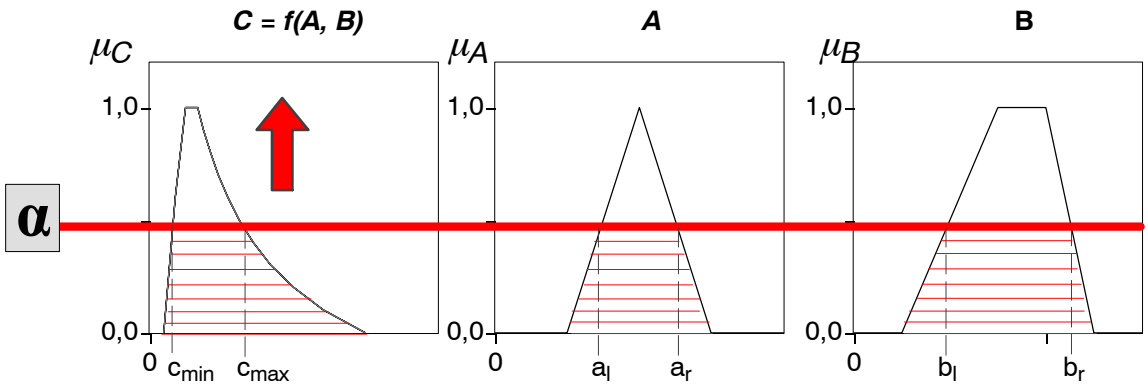


Bild 3. 4 . Berechnung einer *Fuzzy-Zahl* ***C*** als Funktion von zwei *Fuzzy-Zahlen* ***A*** und ***B*** mit Hilfe von α-cuts nach Klir [43], z. B. ***C = A / B***.

Nach Klir [43] wird die Erweiterung in zwei Stufen realisiert. Im ersten Schritt wird bei der Berechnung eines *Fuzzy Sets* ***C*** als Funktion zweier *Fuzzy Sets* ***A*** und ***B*** schrittweise ein Niveau der Zugehörigkeitsfunktionen von ***A*** und ***B*** durchlaufen. Diese, sog. α*-cuts* beginnen mit $\alpha = 0$ und gehen dann bis $\alpha = 1$. Auf dem jeweiligen Niveau eines α*-cuts* werden - vergleichbar zur klassischen Intervallarithmetik - innerhalb der linken und rechten Intervallgrenzen von $\boldsymbol{A}_\alpha$ (a_l und a_r) und $\boldsymbol{B}_\alpha$ (b_l und b_r) über alle Wertepaare **a** und **b** aus beiden Intervallen $\boldsymbol{A}_\alpha$ und $\boldsymbol{B}_\alpha$ die minimalen und maximalen Ergebniswerte zu $\boldsymbol{C}_\alpha$ ermittelt ($\mathbf{c_{min}}$ und $\mathbf{c_{max}}$). Parameterwerte **c** die innerhalb dieses Intervalls $[\mathbf{c_{min}}, \mathbf{c_{max}}]$ liegen, haben für diese Teillösung

einen Zugehörigkeitsgrad α. Außerhalb des Intervalls $[\mathbf{c_{min}}, \mathbf{c_{max}}]$ ist der Zugehörigkeitsgrad 0. Das Gesamtergebnis des ***Fuzzy Sets C*** wird aus der Vereinigungsmenge der Teillösungen aller betrachteten α*-cuts* bestimmt (Maximum der Teillösungen aller α*-cuts*, Bild 3. 5).

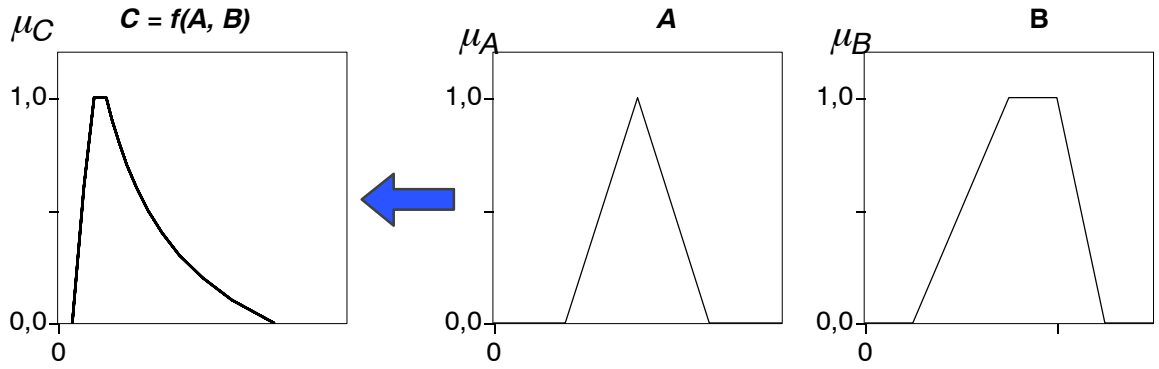


Bild 3. 5 . Ergebnis der Berechnung einer *Fuzzy-Zahl* ***C*** als Funktion aus zwei *Fuzzy-Zahlen* ***A*** und ***B,*** z. B. ***C = A / B***.

Bild 3. 5 zeigt am Beispiel der *Division*, daß das Ergebnis ***C*** nicht zwangsläufig wieder eine Trapezform hat. Die Trapezform ergibt sich exakt nur für die *Fuzzy Addition* oder die *Fuzzy Subtraktion*. In diesen Fällen kann dann das Ergebnis direkt aus den Trapez-Parametern berechnet werden. Für andere Operationen müßte dann jeweils das Erweiterungsprinzip über alle α*-cuts* angewendet werden. Dadurch würde der Rechenaufwand stark ansteigen. Zusätzlich wird es im Einzelfall kaum möglich sein, resultierende nicht-lineare Übergangsfunktionen zwischen $\mu=0$ und $\mu=1$ sinnvoll zu interpretieren. Daher werden im Folgenden sämtliche Ergebnisse bei der Berechnung von *Fuzzy-Zahlen* durch *Fuzzy Sets* in Trapezdarstellung approximiert. Das wird praktisch realisiert, indem nur die α*-cuts* für $\alpha = 0$ und $\alpha = 1$ bei der Funktionsberechnung berücksichtigt werden.

3 .2 .2 Problem der zunehmenden Intervallbreite bei *Fuzzy-Zahlen*

Ein grundsätzliches Problem der *Fuzzy-Arithmetik* wird in Bild 3. 6 mit einem einfachen Beispiel dargestellt. Eine reelle Zahl ***A***=5 wird fortgesetzt mit einer *Fuzzy-Zahl* ***B*** = [1, 1, 0.1, 0.1] multipliziert. Jedes Teilergebnis wird jeweils *A* zugewiesen. Schritt für Schritt weitet sich das Ergebnisintervall ***A*** auf. Überträgt man dieses Beispiel auf die Zusammenfassung von *Fuzzy-Zahlen* in komplexen Modellen ergibt sich vergleichbar: Mit zunehmender Modellkomplexität und je häufiger unscharfe *Fuzzy-Zahlen* miteinander verknüpft werden, steigt die Unschärfe (Intervallbreite) der Ergebnisse. Einerseits ist das insofern plausibel, daß man vom Ergebnis einer unscharfen Modellierung keine höhere Genauigkeit erwarten darf, als sie durch die verwendeten Modellparameter vorgegeben wird. Andererseits muß auch darauf hingewiesen werden, daß man dort, wo in einem *Fuzzy Modell* die Daten präzise bekannt sind, diese auch einheitlich mit in Trapezform beschriebenen *Fuzzy-Zahlen* erfasst werden ($m_1=m_2$, $a=0$, $b=0$).

Die Berechnung arithmetischer Operationen nach dem Erweiterungsprinzip reduziert sich automatisch auf die Berechnung reeller Zahlen, ohne daß weitere Unschärfen entstehen.

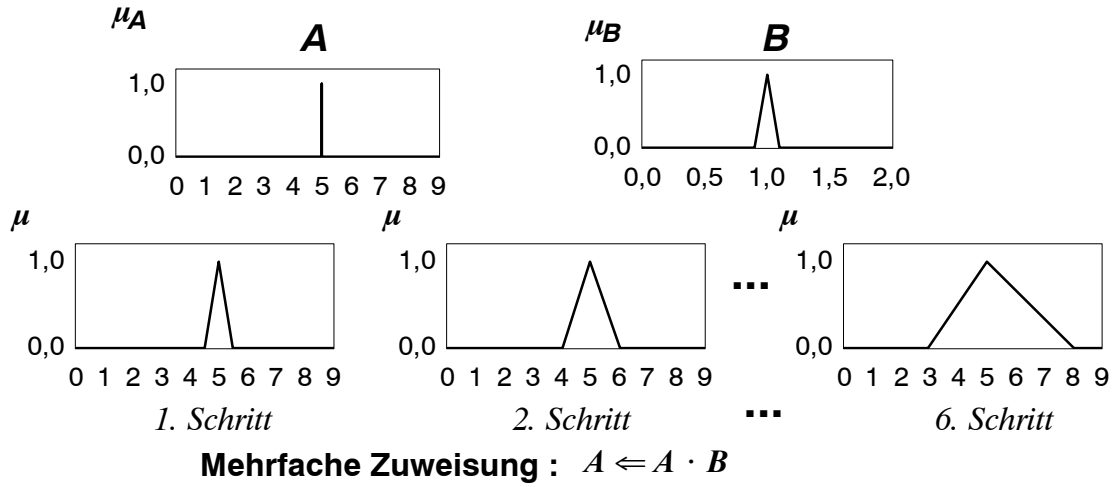


Bild 3. 6 . Mehrfache *Fuzzy Multiplikation* der reellen Zahl ***A***=[5,5,0,0] mit *Fuzzy-Zahl* ***B***=[1,1,0.1,0.1].

3 .3 Verallgemeinerung der Effizienz: *Fuzzy-Multikriterienanalyse*

Yager hat 1978 [99] einen *Fuzzy-Multikriterienansatz* eingeführt, der nach [39] im Zusammenhang mit den Effizienzbetrachtungen als Verallgemeinerung der Gleichungen (2.2) und (2.4) betrachtet werden kann:

$$\mu_f = \mu_{f,1}{}^{w_1} \cdot \mu_{f,2}{}^{w_2} \cdot \ldots \cdot \mu_{f,M}{}^{w_M}, \quad 0 \le \mu_{f,i} \le 1 \qquad \textit{kompensatorisch} \tag{3.3}$$

An Stelle direkter Beziehungen aus Performance- und Kostenwerten einer Modellierung wird bei dem Ansatz von Yager die Erfüllung eines einzelnen formulierten Zieles ***i*** gestellt. Ziele können beispielhaft die gewünschte Fläche oder der aus einer Echtzeitanwendung geforderte Datendurchsatz sein. Der Grad der Zielerfüllung wird dann über den Zugehörigkeitsgrad eines Kriteriums zur Ziellösungsmenge spezifiziert ($\mu_{f,i}$). Für $\mu_{f,i} = 0$ wird ein Ziel gar nicht erfüllt. Für $\mu_{f,i} = 1$ wird ein Ziel vollständig erfüllt. Die einzelnen Ziele können jeweils mit w_i gewichtet werden. Bei einer großen Anzahl zu berücksichtigender Kriterien folgt dann das Problem, wie die einzelnen Gewichte sinnvoll bestimmt werden können. Hier gibt es einen Ansatz von Saaty [79], der ausnutzt, daß man noch relativ einfach für zwei zu vergleichende Kriterien ihre Gewichtung paarweise festlegen kann. Die jeweils paarweise festgelegten Gewichte werden dann in eine Matrix eingetragen. Zu dieser Matrix wird dann ein Eigenvektor bestimmt. Die Komponenten des Eigenvektors werden dann als Gewichte für den Multikriterienansatz in Gleichung (3.3) verwendet. Im Zusammenhang mit dem Ansatz von Saaty [79] kann man also erwarten, daß der Multikriterienansatz nach Yager auch für komplexe Entscheidungsprozesse mit vielen Kriterien einsetzbar ist.

Als Alternative zur Darstellung in Gleichung (3.3), bei der ein besonders gut erfülltes Kriterium die schlechte Zielerfüllung eines anderen Kriteriums kompensieren kann, schlägt Yager auch eine nicht-kompensatorische Betrachtung vor :

$$\mu_f = Min\left\{\mu_{f,1}{}^{w_1}, \mu_{f,2}{}^{w_2}, ..., \mu_{f,M}{}^{w_M}\right\}, \quad 0 \le \mu_{f,i} \le 1 \quad \textit{nicht-kompensatorisch} \tag{3.4}$$

Bei der in Gleichung (3.4) dargestellten nicht-kompensatorischen *Fuzzy-Multikriterienanalyse* minimiert das am schlechtesten erfüllte Kriterium die Gesamtbewertung.

Die Fragestellung, wie man bei Modellierungen aus den Zahlenwerten eines zu bewertenden Kriteriums den jeweiligen Erfüllungsgrad $\mu_{f,i}$ bestimmen kann, ist in [98] untersucht worden.

Hier sollen sowohl Modellierungsziele als auch Modellierungsergebnisse als *Fuzzy-Zahlen* in Trapezform vorliegen. Für ein Kriterium $\boldsymbol{K}$ und sein Ziel $\boldsymbol{K_{Ziel}}$ kann dann eine *Fuzzy-Schnittmenge* $\boldsymbol{S}$ gebildet werden:

$$\boldsymbol{S} = \boldsymbol{K} \cap \boldsymbol{K}_{Ziel} \tag{3.5}$$

Im nächsten Schritt werden die Flächen unterhalb der Schnittmenge $\boldsymbol{S}$ $(\boldsymbol{A_S})$ und unterhalb von von $\boldsymbol{K}$ $(\boldsymbol{A_K})$ bestimmt (Bild 3. 7):

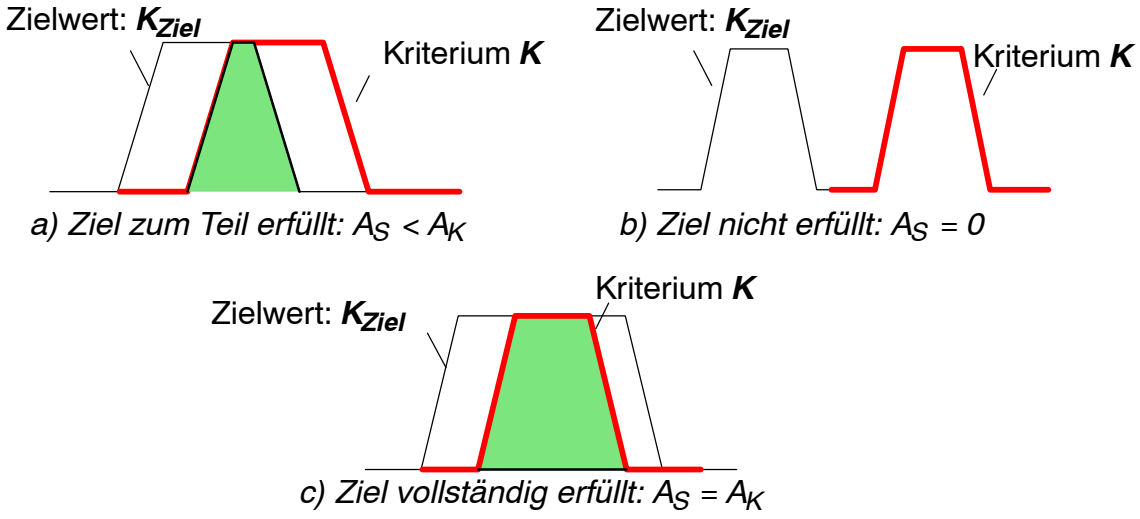


Bild 3. 7 . Zielerfüllung für ein Kriterium $\boldsymbol{K}$, Schnittmenge aus dem Kriterium und dem Zielwert und Fläche unter der Schnittmenge.

Nach [98] kann dann der Zielerfüllungsgrad eines Kriteriums aus dem Verhältnis der Fläche unter der Schnittfläche zur Fläche des betrachteten Kriteriums berechnet werden:

$$\mu_{f,i} = \frac{A_s}{A_K} \tag{3.6}$$

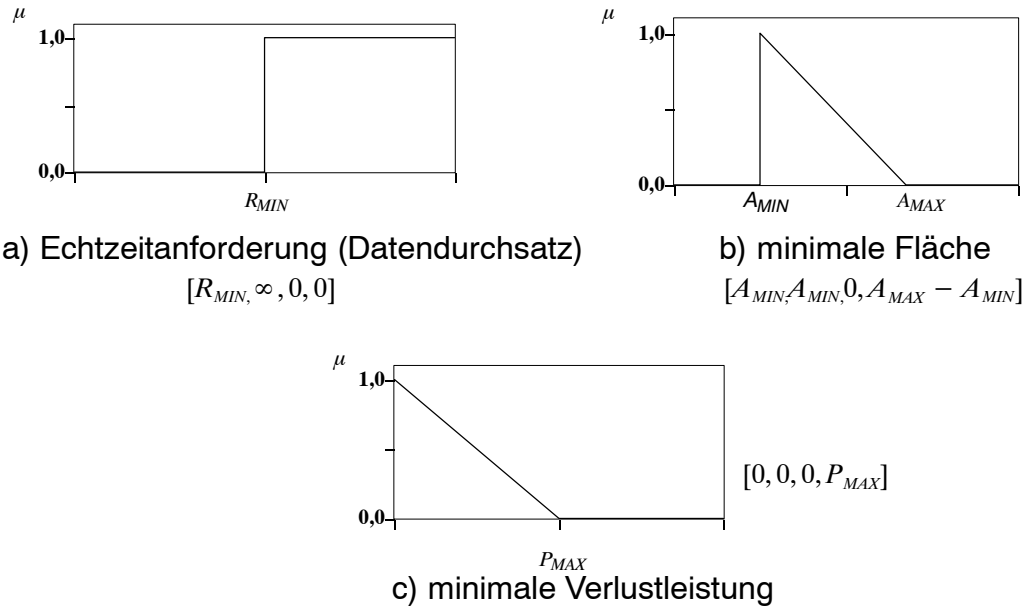


Bild 3. 8 . Charakterisierung von Implementierungszielen für einen *VLSI*-Schaltkreisentwurf mit *degenerierten Fuzzy-Trapezen.*

In Bild 3. 8 wird beispielhaft dargestellt, wie mit Hilfe der Trapezdefinition aus Gleichung (3.2) Entwurfsziele als *Fuzzy Sets* beschrieben werden können. Parameterwerte eines Modellkriteriums mit μ=0 sind nicht gewünscht. Parameterwerte mit μ=1 stellen angestrebte Lösungen dar. In Übergangsbereichen zwischen μ=0 und μ=1 sind Parameterwerte umso mehr gewünscht,je größer ihr Zugehörigkeitsgrad μ ist. Für das Beispiel einer Echtzeitanforderung (Datendurchsatz *R*) berücksichtigen die Zielparameterwerte $R > R_{MIN}$ mit μ=1, daß alle *VLSI*-Implementierungen sicher gewünscht sind, die einen von der zu verarbeitenden Anwendung mindestens geforderten Datendurchsatz R_{MIN} überschreiten. Lösungen unterhalb R_{MIN} sind nicht echtzeitfähig und stellen daher kein brauchbares Ziel dar (μ=0).

Die Herstellungskosten eines Schaltkreises werden stark von der Schaltkreisfläche beeinflußt, weil die Größe eines Schaltkreisentwurfes exponentiell in den Anteil der nach einer Fertigung als defekt ermittelbaren Schaltkreise eingeht. Das Ziel möglichst niedriger Herstellungskosten wird daher umso besser erreicht, je kleiner eine Schaltkreisfläche ist. Das Ziel einer minimalen Fläche wird beispielhaft in Bild 3. 8 b dargestellt. A_{MAX} stellt eine obere Grenze für eine Schaltkreisgröße dar, nach deren Überschreitung für eine geplante Anwendung keine akzeptable Ausbeute mehr erzielbar ist. A_{MIN} ist eine untere Grenze für die Schaltkreisgröße die sich aus der Gesamtkostenbetrachtung ergibt. Sie ist dann bergündbar, wenn der von einem Flächenmodell nicht erfaßte Kostenanteil für ein Schaltkreisgehäuse unterhalb von A_{MIN} überwiegt.

Auf Grund der in diesen Beispielen nicht mehr direkt erkennbaren Trapezform werden die dargestellten *Fuzzy Sets* oft auch als *Fuzzy Sets* mit degenerierter Trapezform bezeichnet ([43], S. 173).

3 .4 Anforderungen an eine Fuzzy-Modellierungs-Software

Untersuchungen in frühen Konzeptphasen einer Architekturentwicklung werden häufig mit Tabellenkalkulationsprogrammen durchgeführt. Ein Schwerpunkt ist hier die Abschätzung der Schaltkreisfläche [81].

Eine detaillierte Untersuchung der in dieser Arbeit vorgeschlagenen *Fuzzy-Modellierung* stellt neue Anforderungen, die weit über den Leistungsumfang bekannter Tabellenkalkulationsprogramme hinaus gehen und mit dem in *Anhang A* beschriebenen *VSP Decision Program* realisiert wurden:

- Hierarchische Zerlegung von Kosten- und Performance-Modellen (Kapitel 2 .2)
- Freie Spezifikation analytischer Kosten- und Performance-Modelle (Tabelle 2.2)
- Erweiterung analytischer Funktionen auf den Datentyp : ***Fuzzy-Zahl*** (Kapitel 3 .2 .1)
- Unterstützung einer ***Fuzzy-Multikriterienanalyse*** (Kapitel 3 .3)
- Nichtlineare Optimierung des Erfüllungsgrades einer *Fuzzy-Multikriterienanalyse* nach Kapitel 3 .3 , für frei wählbare analytische Kosten- und Performance-Funktionen, z. B. nach Tabelle 2.2 .

4 Ein Modellansatz für applikationsspezifische VLSI-Schaltkreise *(ASICs)*

Dem in Bild 1. 1 dargestellten *Design Gap,* d. h. der wachsenden Lücke zwischen den Fortschritten der *VLSI*-Technologie und der nur langsam ansteigenden Produktivität im Entwurfsprozeß kann auf verschiedenen Ebenen begegnet werden. Nach [78] soll der Schlüssel zur Beherrschung des *Design Gap* in der Erhöhung des Abstraktionsgrades der Schaltungssynthese liegen (*High Level Synthese)*. Ob eine *High Level Synthese* zu einer im Hinblick auf Performance und Kosten optimalen *VLSI*-Implementierung führt, wird in [78] offen gelassen und ist insofern fraglich, weil frühzeitig Architekturmerkmale festgelegt werden müssen, z. B. bei der Auswahl einzusetzender Prozessorkerne. Variationen im Lösungsraum erfordern erneute Spezifikationen alternativer Architekturansätze. Die Bewertung von Alternativen ist in der Regel erst nach rechenzeitintensiven Syntheseschritten und Simulationen möglich. Als Folge wird eine *High Level Synthese* aus Aufwandsgründen nur diskrete Punkte des für ein neues Architekturkonzept zur Verfügung stehenden Lösungsraumes untersuchen können. Dieser iterative Ansatz schließt nicht aus, daß brauchbare und teiloptimierte Lösungen gefunden werden. Eine Annäherung einer Architekturentwicklung an das globale Optimum des Entwurfsraumes ist fraglich. Es werden daher neue Lösungen benötigt, die auf Architekturebene und vor den Syntheseschritten eine flexiblere Erkundung des Entwurfsraumes unterstützen.

Hier bietet der in Kapitel 2 .1 eingeführte Modellansatz für Mikroprozessoren (nach Fu [18]) eine erste, vielversprechende Lösung. In [17] wird berichtet, daß mit diesem Modell schon frühzeitig und vor Verfügbarkeit des *AMD Athlon Prozessors* vertrauenswürdige Kosten- und Performance-Abschätzungen durchgeführt werden konnten, die im Internet weltweit eine starke Beachtung fanden und die eine grundsätzliche Anwendbarkeit vereinfachter Architekturmodelle belegen.

In diesem Kapitel soll auf Basis des in den Kapiteln 2 und 3 dargestellten Standes der Technik ein neuer, leistungsfähiger Ansatz für die Modellierung von *ASICs* entwickelt und begründet werden. Im Hinblick auf die Entwicklung und die Verifikation des neuen Ansatzes konzentriert sich die folgende Untersuchung auf programmierbare *Videosignalprozessoren (VSPs)*, die ein besonders anspruchsvolles Anwendungsgebiet darstellen. Hier sind einerseits für die Echtzeitverarbeitung von Bewegtbildern besonders hohe Rechenleistungen gefordert. Andererseits dominiert bei Konsumeranwendungen das Ziel möglichst geringer Herstellungskosten. Als Folge wird häufig eine möglichst hohe Effizienz, d. h. eine hohe Rechenleistung bei kleiner Schaltkreisgröße gewünscht. Bei vielen Anwendungen, z. B. in Mobiltelefonen, muß gleichzeitig noch eine geringe Leistungsaufnahme gewährleistet sein. Eine optimale *ASIC*-Architektur muß dann den besten Kompromiß zwischen teilweise gegensätzlichen Entwurfszielen bieten.

In Kapitel 4 .1 wird gezeigt, daß im Vergleich zu einfachen Mikroprozessormodellen die Anforderungen an die Modellierung von *ASICs* steigen. In Kapitel 4 .2 wird ein aus diesen Anforderungen resultierender Gesamtansatz spezifiziert. In Kapitel 4 .3 folgt eine am generischen Architekturmodell (2 .2) orientierte technologieunabhängige Modellebene. Kapitel 4 .4 diskutiert die Abbildung auf eine Zieltechnologie.

4 .1 Übergang von einfachen Mikroprozessormodellen zu komplexen *ASIC*-Modellen

Der von Fu entwickelte Ansatz zur Modellierung von Mikroprozesoren ([18])läßt sich in zwei Ebenen unterteilen (Bild 4. 1). In der ersten Ebene werden unabhängig von der Zieltechnologie einer VLSI-Implementierung Größen von *Floating-Point-Einheiten* und *Caches* sowie Verarbeitungszeiten für bekannte Benchmarks abgeschätzt. Grundlage der Technologieunabhängigkeit ist der Einsatz einer speziellen Architekturbibliothek, die aus der Analyse bekannter Lösungen für *Floating-Point-Units (FPUs)* und *Caches* gewonnen wird. Als charakteristische Kosten- und Performance-Kennwerte werden hier die Siliziumfläche und die Verzögerungszeit berücksichtigt. Sowohl die Siliziumfläche, als auch die Verzögerungszeit werden auf die Kennwerte eines *NAND2* normiert. Bei Anwendung in einer neuen Zieltechnologie müssen dann lediglich Siliziumflächen und Verzögerungszeiten gemäß der Fläche und den Verzögerungszeiten eines *NAND2* umgerechnet werden.

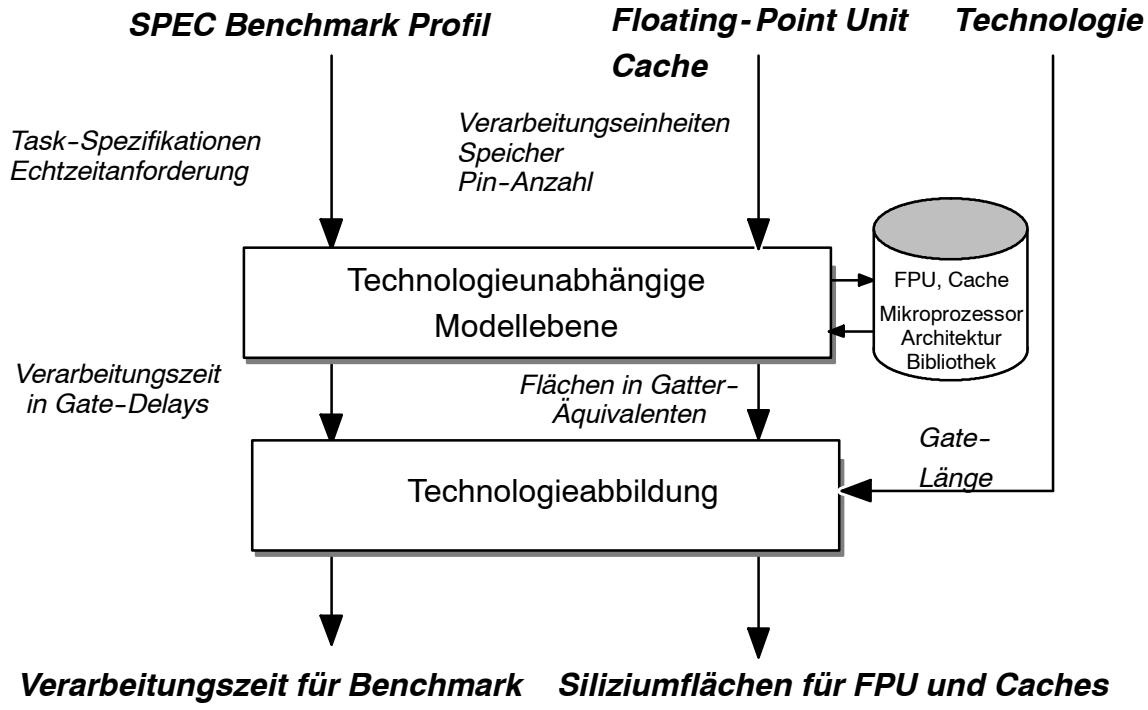


Bild 4. 1 . Zweistufiger Modellansatz von Fu [18] zur Kosten- und Performance-Modellierung von Mikroprozessoren.

Der Ansatz von Fu zeigt, daß man mit stark vereinfachten Modellen und ohne detaillierte Simulationsmodelle bereits brauchbare Aussagen zu Kosten und Performance der *VLSI*-Implementierung eines Mikroprozessors gewinnen kann. Im Hinblick auf die geschlossene Handhabung liegen wesentliche Vereinfachungen darin, daß sich das Modell auf die Komponenten beschränkt, die den größten Anteil an der Siliziumfläche von Mikroprozessoren belegen und die den größten Einfluß auf die Performance haben (*Floating-Point-Units*, *Caches*).

Neben der grundsätzlichen Handhabbarkeit des Modells erweist sich die zweistufige Unterteilung in eine technologieunabhängige Modellebene und in eine Technologieabbildung als sinnvoll. Auf der technologieunabhängigen Modellebene ist auf Basis der normierten Zeit- und Flächeneinheiten ein einheitlicher Vergleich unterschiedlicher Architekturvarianten möglich. In der zweiten Ebene kann dann untersucht werden, wie sich Kosten- und Performance-Werte eines Mikroprozessors bei zukünftigen VLSI-Technologien entwickeln können.

Im Folgenden soll versucht werden, vergleichbar zu Fu's Modell *ASIC*-Architekturen zu modellieren. Will man den Ansatz von Fu auf die Modellierung von *ASICs* übertragen, erkennt man, daß die Normierung auf eine einheitliche, technologieunabhängige Ebene hier auch sinnvoll ist. So werden in [74] *ASICs* durch Normierung auf eine Technologiebasis abgebildet und einheitlich verglichen. Es gibt aber auch Gründe, die einer direkten Übertragung des Modells von Fu auf die Modellierung von *ASICs* entgegenstehen.

Auf der Anwendungsseite gibt es kein anerkanntes Benchmark Profil, aus dem einheitliche Anforderungen an *ASICs* abgeleitet werden können. Ein *ASIC*-Modell muß die applikationsspezifischen Anforderungen der zu verarbeitenden Algorithmen einer Anwendung erfassen können. Weiterhin sind *ASIC*-Architekturen im Vergleich zu Mikroprozessoren häufig heterogener strukturiert, mit einzelnen, an Teilverarbeitungsschritte besonders angepassten Verarbeitungseinheiten. An Stelle von Caches werden häufig auch lokale Speicher eingesetzt, z. B. wenn die Abfolge von lokalen und externen Speicherzugriffen deterministisch beschreibbar ist. Als Folge darf die Modellierung von Schaltkreisflächen und von Verarbeitungszeiten nicht alleine auf *Floating-Point-Units* und auf *Caches* beschränkt bleiben und muß eine allgemeinere Spezifikation von Verarbeitungseinheiten und Speichern unterstützten.

Neben den Anforderungen an die Verarbeitung und den zu untersuchenden Architekturansätzen, ist auch für die *VLSI*-Technologien zu berücksichtigen, daß es gravierende Unterschiede zwischen Mikroprozessoren und *ASICs* gibt, die sich unter anderem in der unterschiedlichen Skalierung zeigen (Tabelle 2.1). Wenn man die diskutierten Unterschiede zwischen Mikroprozessoren und *ASICs* und die in Kapitel 3 eingeführten Ansätze aus der *Fuzzy Set Theorie* berücksichtigt, läßt sich die in Bild 4. 2 dargestellte Zielsetzung für einen neuen Modellansatz für *ASICs* definieren, die sich an [39] orientiert.

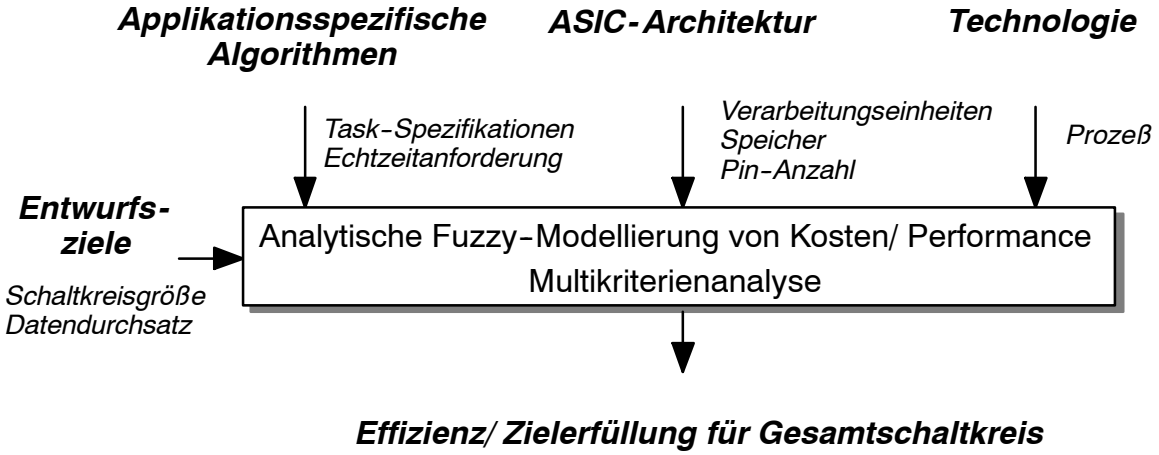


Bild 4. 2 . Zielsetzung für einen neuen Modellansatz für *ASICs*

4 .2 Spezifikation eines neuen mehrstufigen Modells für *ASICs*

In diesem Kapitel soll die in Bild 4. 2 dargestellte Zielsetzung in ein neues Modell für *ASICs* umgesetzt werden. Zunächst soll die von Fu [18] vorgeschlagene zweistufige Lösung auf die Modellierung übernommen werden und um eine zusätzliche Bewertungsebene (Multikriterienanalyse nach Kapitel 3 .3) ergänzt werden. Das in Bild 4. 3 spezifizierte Modell beginnt mit Eingangsdaten für die zu verarbeitenden applikationsspezifischen Algorithmen, für die zu untersuchende *ASIC*-Architektur und für die geplante Zieltechnologie einer späteren *VLSI*-Implementierung. Die Beschreibung von Algorithmen soll auf *Task*-Ebene erfolgen. Eine *Task* soll aus einer Abfolge von Grundoperationen bestehen, z. B. *Addition, Subtraktion, Multiplikation.* Gleichzeitig soll die Spezifikation einer Task auch Datenzugriffsanforderungen in Abhängigkeit vom verfügbaren Speicher enthalten.

Eine *ASIC*-Architektur soll mit dem in Kapitel 2 .2 eingeführten generischen Architekturmodell definiert werden. Hierzu gehören Angaben über Verarbeitungseinheiten (PU), lokale Speichermodule (LM), Steuerungseinheiten (CU), Vernetzung der Module (NW) und die Anzahl der Anschlüsse *(Pins)* eines Schaltkreises. Da *ASICs* überwiegend als kundenspezifische Schaltkreise realisiert werden, z. B. mit Standardzellentwürfen, soll das Modell auf einer Architekturbibliothek basieren, die aus synthetisierten Grundmodulen gewonnen wird. Im nächsten Schritt soll eine Abbildung auf eine Zieltechnologie erfolgen. Das Abbildungsergebnis soll dann mit dem in Kapitel 3 .3 entwickelten *Fuzzy-Multikriterienansatz* im Hinblick auf spezifizierte Entwurfsziele bewertet werden.

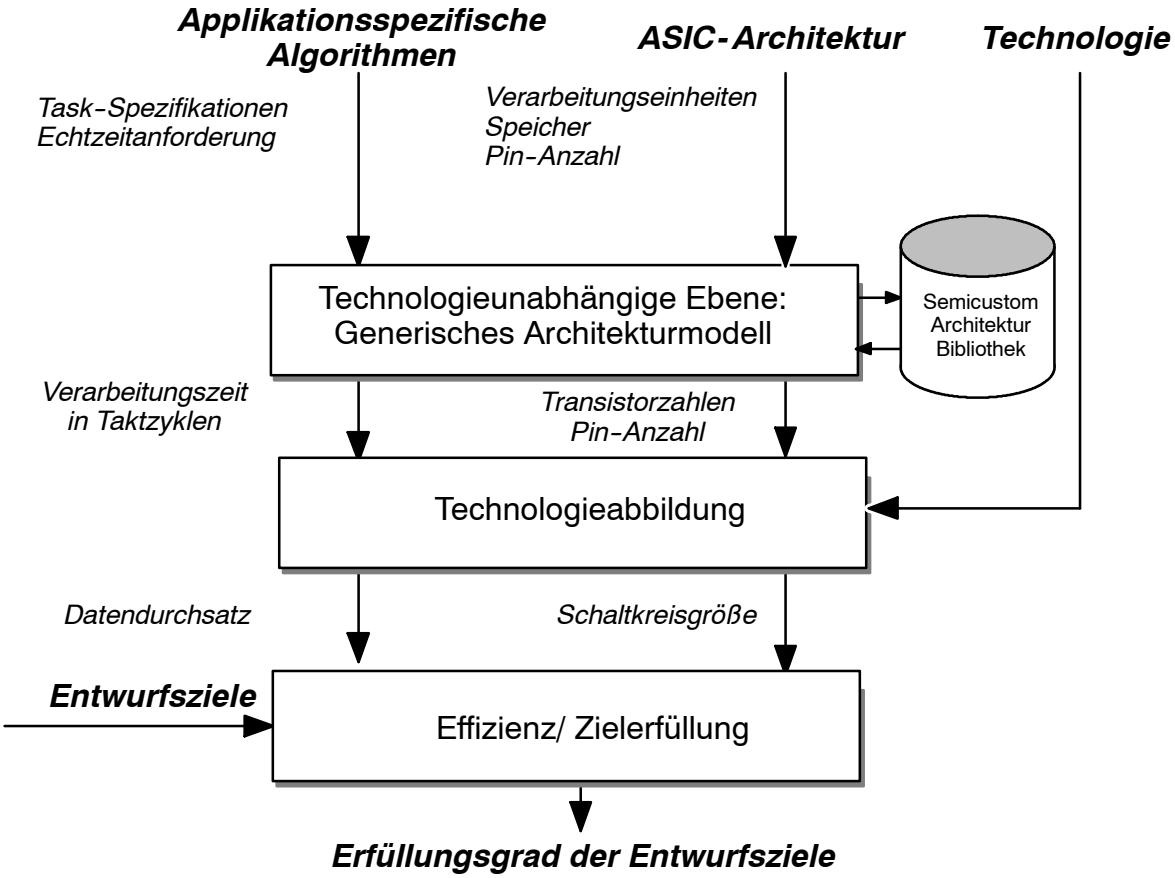


Bild 4. 3 . Neuer, dreistufiger Modellansatz für *ASICs*

Im weiteren Verlauf der Arbeit soll der in Bild 4. 3 eingeführte Modellansatz detaillierter spezifiziert und untersucht werden.

4 .3 Technologieunabhängige *ASIC*-Modellierung

In diesem Abschnitt soll der technologieunabhängige Teil des in Kapitel 4 .2 , Bild 4. 3 eingeführten *ASIC*-Modells entwickelt werden. Basis soll das in Kapitel 2 .2 eingeführte generische Architekturmodell sein. Im Hinblick auf eine applikationsspezifische Architekturbewertung werden in Kapitel 4 .3 .1 am Beispiel besonders rechenleistungsintensiver Algorithmen der Videosignalverarbeitung die Anforderungen einer Applikation diskutiert. In 4 .3 .2 wird eine technologieunabhängige Charakterisierung von Datenpfaden und Speichern als wichtigste Architekturelemente diskutiert. In Kapitel 4 .3 .3 wird die Verknüpfung der Anforderungen der zu verarbeitenden Algorithmen mit den Merkmalen einer Architektur entwickelt. Häufig wird für derartige Rechenzeitbestimmungen gefordert, daß die Verteilung von *Tasks* auf *Prozessorelemente* (*Assignment*) und die Festlegung der zeitlichen Reihenfolge der Verarbeitungsschritte (*Scheduling*) zu berücksichtigen sind ([80], [46]). Eine alternative Untersuchung in [41] hat für den Vergleich von Architekturen gezeigt, daß eine Beschränkung auf ein *Assignment* unter Vernachlässigung des Einflusses des *Schedulings* auch zulässig sein kann,

sofern eine genügend grobe *Task*-Granularität gegeben ist. Daher soll auch der folgende Lösungsweg in Kapitel 4 .3 .3 auf einem *Assignment*-Ansatz basieren.

4 .3 .1 Charakterisierung von Algorithmen

Die Rechenleistungsanforderungen für die Echtzeitverarbeitung von Videosignalen wird stark von der Bildgröße und der Bildwechselfrequenz einer zu verarbeitenden Signalquelle beeinflußt [74]. Tabelle 4.1 stellt beispielhaft für ausgewählte Bildformate die zugehörigen Parameter und resultierende Quellendatenraten R_S dar.

	Bild-punkte	**Bildwechselfrequenz/ Hz**	**Quellendatenrate R_S / Mbyte/ s**
QCIF	174 · 144	10	0.3758
CIF	352 · 288	10 - 30	1.52 - 4.56
CCIR	720 · 576	25	20.74

Tabelle 4.1 . Quellendatenraten R_S für bekannte Videosignalformate, nach [74].

Neben der Bildgröße des verarbeiteten Videosignales und der zugehörigen Bildwechselfrequenz hat auch der Charakter der zu verarbeitenden Algorithmen einen großen Einfluß auf die Architektur der für die Verarbeitung einzusetzenden VLSI-Architekturen.

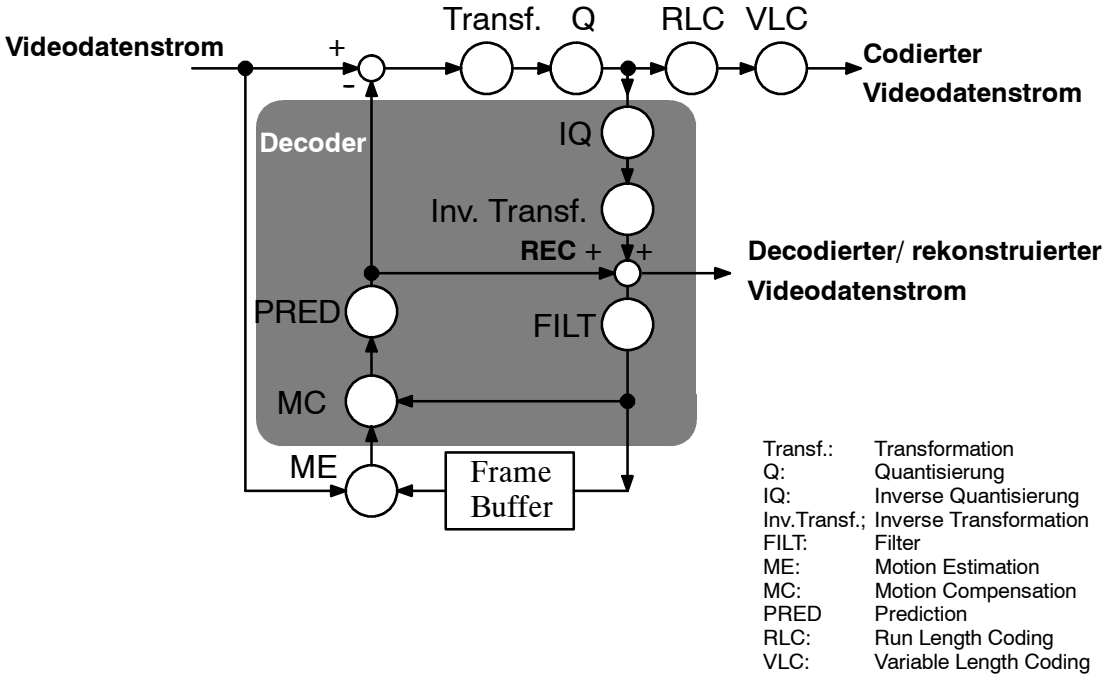


Bild 4. 4 . Beispiel für einen Codierungsalgorithmus für Videodaten, nach [95].

In Bild 4. 4 wird als Beispiel das Blockdiagramm eines Videocodierungsalgorithmus dargestellt. Diese wird häufig auch als Hybrid-Codierung bezeichnet [74],[38] und findet sich in Variationen in vielen als Produkt realisierten Videocodierungssystemen wieder, z. B. in Bildtelefonen, DVD-Playern, digitalen Videorecordern und dem digitalen Fernsehen (DVB-T).

Die Darstellung in Bild 4. 4 ist an den grundlegenden Teilaufgaben des Verfahrens *(Tasks)* orientiert. Eine *Task* besteht jeweils aus einer Abfolge von arithmetischen Operationen, z. B. Addition, Subtraktion, Multiplikation und Akkumulation. Die Abfolge der arithmetischen Operationen kann sich entweder zyklisch und statisch über wechselnde Datensätze wiederholen *(Low Level Verarbeitung)* oder ihr Charakter der Verarbeitung hängt stärker von den zu verarbeitenden Bildinhalten ab *(Medium* und *High Level Verarbeitung)*. Nach [84] gewinnt der inhaltsabhängige Charakter der Verarbeitung auch im Hinblick auf die Frequenz von *Low Level Tasks* zunehmend an Bedeutung und kann im Hinblick auf eine effiziente *ASIC-Hardware* ausgenutzt werden. Im weiteren Verlauf dieser Arbeit wird der Einfluß der Verarbeitung auf die Häufigkeit der einzelnen *Tasks* vereinfachend als statisch angesetzt. Bei konkreten Entwurfsprojekten, mit vom Dateninhalt abhängigen Rechenleistungsanforderungen, muß im Einzelfall der in [84],[85] diskutierte, inhaltsabhängige Charakter der Verarbeitung berücksichtigt werden.

Bild 4. 5 zeigt an Hand einfacher Beispiele eine mögliche Basis für die Charakterisierung von Signalverarbeitungsalgorithmen nach [46]. Vereinfachend werden hier die in den einzelnen *Tasks* einer Anwendung zu verarbeitenden arithmetischen und logischen Operationen charakterisiert als *skalare Operationen* oder als *Datenpfadoperationen*. *Skalare Operationen* sollen als Einzeloperationen interpretiert werden, wie z. B. *Addition, Subtraktion, Multiplikation, Division* oder *Shift*. *Datenpfadoperationen* sollen aus einer festen Abfolge von skalaren Operationen zusammengesetzt sein, die sich periodisch wiederholen, z. B. *Multiplikation* und *Akkumulation* für Filteralgorithmen, Transformationen oder Korrelationen.

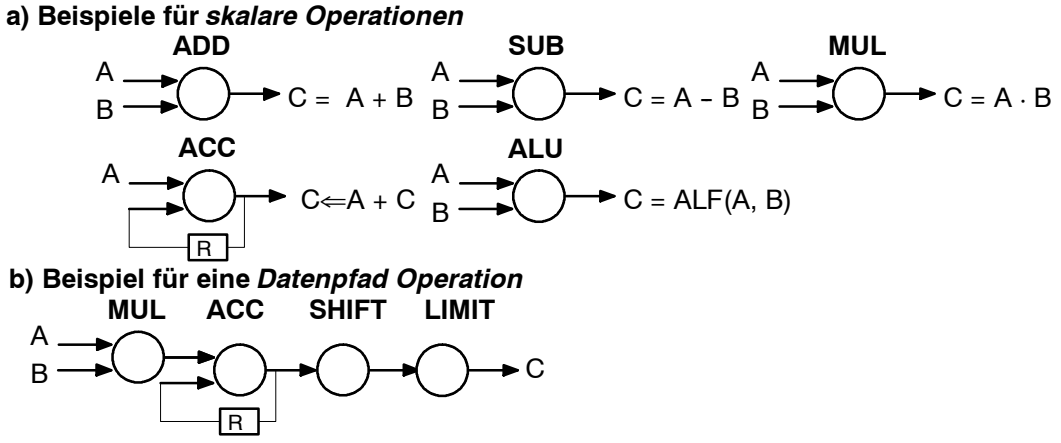


Bild 4. 5 . Beispiele für skalare Operationen und eine Datenpfad Operation, die aus vier skalaren Operationen besteht (MUL, ACC, SHIFT, LIMIT).
ALF: **A***rithmetisc*h **L***ogische* **F***unktion*

Auf Basis der gewählten Klassifikation zeigt Tabelle 4.2 eine quantifizierte Darstellung der Operationsanforderungen von Signalverarbeitungsalgorithmen (nach [46]).

	N_{OP}, Anzahl der ***Operationen pro byte***		
	$N_{OP,S}$: Skalar	**$N_{OP,DP}$: Datenpfad**	**$N_{OP,AS}$: Alle Skalar**
ME +/- 7	0.1	18	54.1
FILT/PRED	0	9	20
Transf. (DCT)	0	16	34
Q	6	1	10
IQ	0	1	4
Inv.Transf.(IDCT)	0	16	34
RLC	4	0	4
VLC	8.1	0	8.1
REC	1	0	1
Summe	**19.2**	**61**	**169.2**

Tabelle 4.2 . N_{OP}: Operationen pro byte im Videodatenstrom für einen Video-Encoder (Bild 4. 4), nach [46] (***Skalar:*** Summe der skalaren Operationen nach Bild 4. 5 a, ***Datenpfad:*** zusammengesetzte Datenpfadoperationen nach Bild 4. 5 b, ***Alle Skalar****:* Summe aus direkt angegebenen skalaren Operationen und der skalaren Operationen die aus den Einzeloperationen der Datenpfadoperationen abgeleitet werden).

Jede *Datenpfadoperation* in Tabelle 4.2 besteht jeweils aus einer festen Abfolge von *skalaren Operationen*. Entsprechend stellt die letzte Spalte der Tabelle die Summe aus den einzelnen *Skalaroperationen* und der Anzahl der jeweils in einer *Datenpfadoperation* enthaltenen, fest verknüpften *skalaren Operationen* dar.

Mit den Tabellen 4.1 (Quellendatenrate) und 4.2 und den Gleichungen (4.1) und (4.2) können nun die jeweiligen Rechenleistungsanforderungen an einen echtzeitfähigen Prozessor bestimmt werden.

$$R_{OP,S} = N_{OP,S} \cdot R_S \tag{4.1}$$

$$R_{OP,DP} = N_{OP,DP} \cdot R_S \tag{4.2}$$

Mit Gleichung (4.3) kann eine für *Skalaroperationen* und *Datenpfadoperationen* zusammenfassende Kennzahl zur Rechenleistungsanforderung definiert werden:

$$R_{OP,AS} = N_{OP,AS} \cdot R_S \tag{4.3}$$

Neben den bisher diskutierten Rechenleistungsanforderungen müssen weitere Einflußgrößen berücksichtigt werden. Hierzu gehört die für die Verarbeitung arithmetischer Operationen geforderte Wortbreite. Für besonders rechenintensive *Tasks*, die direkt auf Bildpunkten arbeiten, genügt meist eine Auflösung von *8 bit,* während nachfolgende Schritte *16 bit* oder *32 bit* erfordern. Im Hinblick auf eine effiziente *VLSI*-Implementierung ist daher in [38] eine in der Wortbreite zeitlich veränderliche Genauigkeit vorgeschlagen worden, die bei niederigen Wortbreiten eine höhere Parallelität und Verarbeitungsleistung unterstützt. Mit der in [84] beschriebenen Lösung wird gezeigt, daß sich diese Richtung noch konsequenter erweitern läßt und auch in aktuellen Technologien ein wesentliches Merkmal für besonders effiziente *VLSI*-Implementierungen darstellt.

Neben Rechenleistungsanforderungen und Wortbreitenbetrachtungen muß für die Entwicklung einer leistungsfähigen *VLSI*-Architektur auch die Datenführung und Zwischenspeicherung berücksichtigt werden. Der Einsatz von lokalen Zwischenspeichern kann die Anzahl und Rate externer Zugriffe reduzieren, sofern Zwischenergebnisse lokal weiterverwendbar sind.

Hier erlaubt die Software-Implementierung des Modellansatzes mit dem *VSP Decision Program* eine flexible Spezifikation von Datenzugriffsanforderungen in Abhängigkeit von verfügbaren Speichergrößen auf Basis hinterlegter Listen. Bild 4. 6 zeigt hier ein Beispiel für die Spezifikation der Zugriffe eines *Video-Coders*. In dem dargestellten Beispiel variiert die Zugriffsanforderung vom Faktor [35,135,0,0] (ohne lokale Speicher) bis [3,10,0,0] (lokaler Speicher mit mehr als 1000 byte). Insgesamt ergibt sich eine für die Echtzeitsignalverarbeitung geforderte Zugriffsrate $\boldsymbol{R_{I/O}}$ als

$$R_{I/O} = R_S \cdot N_{I/O}(Speichergröße) \tag{4.4}$$

Die in Bild 4. 6 dargestellten Datenzugriffe stellen die minimalen Anforderungen dar, die sich aus den zu implementierenden Algorithmen ergeben. In der Praxis kann es passieren, daß der Programmablauf in einem Prozessor größere Speicher voraussetzt oder daß auf Grund von zeitlichen Lücken in der Datenzufuhr höhere Zugriffsraten notwendig sind. Weitere Details zur Charakterisierung von Algorithmen werden tabellarisch in ***Anhang C*** dargestellt.

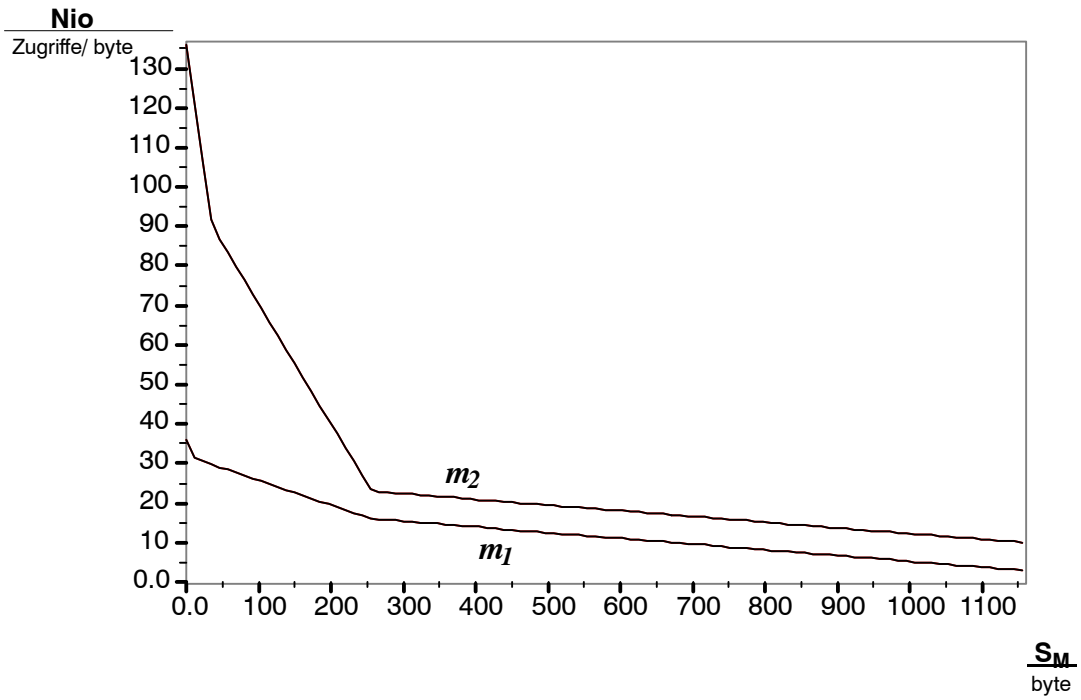


Bild 4. 6 . Beispiel zu Datenzugriffsanforderungen in Abhängigkeit der verfügbaren Speichergröße$\boldsymbol{S_M}$ in Intervallbeschreibung [m_1,m_2,*0,0*] für einen Video-Codierungsalgorithmus.

4 .3 .2 Charakterisierung von Architekturelementen

In Kapitel 2 .2 (Bild 2. 1) ist ein generisches Architekturmodell eingeführt worden. In diesem Abschnitt der Arbeit soll diskutiert werden, wie das generische Modell so quantifiziert werden kann, daß es für konkrete Anwendungen aus der Modellierung einsetzbar ist. Vorausgesetzt wird hier *Pipelining* mit synchron getakteten Schaltungen.

Ziel dieser Charakterisierung von Architekturelementen ist eine möglichst technologieunabhängige Beschreibung der Zusammenhänge zwischen Schaltkreisgrößen und Verarbeitungszeiten. Erfahrungsgemäß hängt der Zusammenhang zwischen Modulgrößen und Verarbeitungszeiten stark vom Entwurfsstil ab. Hier bieten *Fullcustom Entwürfe*, bei denen ein Schaltkreisentwickler von der entworfenen Logik bis zum endgültigen Layout auf jeder Entwurfsebene eingreifen kann, besondere Freiheitsgerade für die Entwicklung besonders kompakter und leistungsfähiger Schaltkreisrealisierungen. Gleichzeitig ist aber ein sehr großer Entwicklungsaufwand erforderlich, der sich häufig nur noch für hohe Stückzahlen begründen läßt. Eine weniger aufwendige Alternative für die *ASIC-Entwicklung* sind *Semicustom Entwürfe,* die heute überwiegend auf *Standardzellen* basieren. Dabei werden die Funktionseinheiten eines *ASICs* zunächst funktional beschrieben und dann mit einem Syntheseprogramm auf eine verfügbare Zellbibliothek abgebildet.

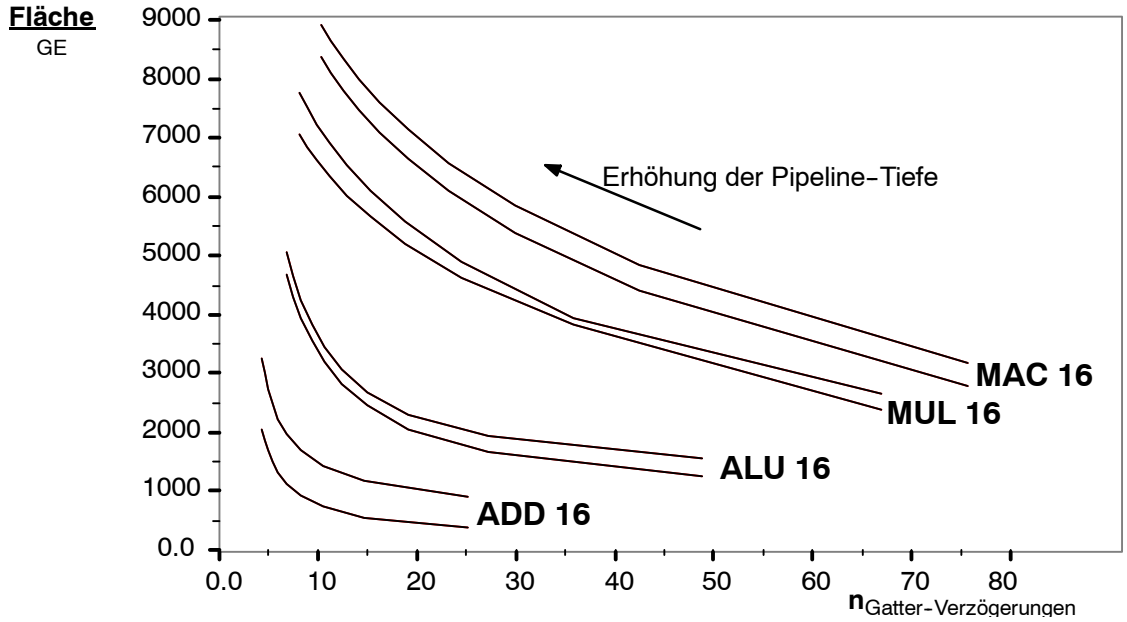


Bild 4. 7 . Beispiel für die Charakterisierung von 16 bit Architekturelementen nach [1], mit jeweils 1 bis 10 Pipeline-Stufen, $n_{Gatter\text{-}Verzögerungen}$: Periodendauer des Pipeline-Taktrate, normiert auf die Gatterverzögerungszeit eines *NAND2* Gatters, GE: Gatter-Flächenäquivalente.

In [1] ist dieser Abbildungsprozeß für gängige Operationen aus der Signalverarbeitung und die zugehörige Abbildung auf Datenpfadelemente untersucht worden. Hier wurde mit einer Vielzahl von Einzelschritten der mögliche Entwurfsraum im Hinblick auf Operationen, Wort-

breiten der Verarbeitung und Pipelinetiefen untersucht. Verarbeitungszeiten und Modulgrößen wurden auf die Kennwerte eines *NAND2* Gatters normiert. Bild 4. 7 zeigt beispielhaft die Synthese-Ergebnisse für Datenpfadelemente mit 16 bit Eingangsdaten.

Die in [1] ermittelten Modellkurven für Flächen und Verzögerungszeiten in Abhängigkeit von der Anzahl der Pipeline-Stufen können für die quantitative Modellierung von *ASIC*-Architekturen eingesetzt werden. Die Umrechnung von Flächen und Verzögerungszeiten in eine Zieltechnologie kann dann recht einfach über die in der Zieltechnologie bekannten Parameter eines *NAND2* erfolgen. Sofern nicht in Gatter-Äquivalenten sondern mit Transistorzahlen gerechnet wird, soll im Folgenden vereinfachend vorausgesetzt werden, daß ein Gatter äquivalent zu 4 Transistoren ist.

Eine hohe Rechenleistung der skalaren und datenpfadorientieren Verarbeitungseinheiten eines *ASICs* ist eine notwendige Voraussetzung für einen bei Echtzeitanwendungen geforderten hohen Datendurchsatz. Trotzdem kann es passieren, daß ein Schaltkreis eine hohe numerische Rechenleistung bietet, die für die Verarbeitung benötigten Daten aber nicht zum geforderten Zeitpunkt an den zugeordneten Verarbeitungseinheiten verfügbar sind. Es entstehen dann Lücken in der Verarbeitung bei einer gleichzeitig schlechten Auslastung vorhandener numerischer Resourcen. Hier können lokaler Datenspeicher Abhilfe bieten. Sie unterstützen mehrfache lokale Zugriffe auf Datensätze oder auf berechnete und lokal erneut verwendbare Zwischenergebnisse bei gleichzeitig kurzen *On-Chip-Zugriffszeiten*. Bild 4. 6 hat hier beispielhaft dargestellt, wie sich die Anzahl der externen Zugriffe eines Prozessors reduzieren läßt, indem ein schneller lokaler Speicher auf dem Schaltkreis implementiert wird. Neben dem Performance-Gewinn müssen auch die Kosten für *On-Chip-Speicher* erfaßt werden. Im Hinblick auf eine in der ersten Modellebene technologieunabhängige Beschreibung sollen die Kosten eines Speichers über der Anzahl der Transistoren erfaßt werden. Die Mehrheit der auf dem Gebiet der Videosignalverarbeitung veröffentlichten *ASICs* [2], [3], [5], [10], [12], [21], [27], [29], [30], [34], [36], [40], [44],[47] ,[50], [51], [60], [61], [62], [67], [68], [69], [70], [71], [87], [88], [89], [90], [35] hat statische lokale Speicher *(SRAM)*. Nur in wenigen Fällen [50] findet man auf dem Schaltkreis zusätzlich implementierte, dynamische Speicher.

Bild 4. 8 zeigt das Grundschaltbild einer 6-Transistor-Speicherzelle. Die Informationsspeicherung wird von den Transistoren T_1 bis T_4 übernommen. Dei Speicherzelle wird über das Signal *Wordline* selektiert, das den Weg der Signale *Bitline* und des invertierten Signales *Not Bitline* über zwei als Pass-Transistoren eingesetzte *NMOS*-Transistoren (T_5, T_6) freischaltet. Bei Multiport-Speichern kann die Anzahl der Ports erhöht werden, indem weitere als Pass-Transistoren eingesetzte NMOS-Transistoren, vergleichbar zu T_5, T_6 eingesetzt werden. Vernachlässigt man den Aufwand für die Logik zur Ansteuerung, kann die Transistoranzahl für *Multiport-On-Chip-SRAM-Speicher* als Funktion aus der Port-Anzahl (n_{Port}) und der Speicherkapazität in bit (n_{bit}) charakterisiert werden mit:

$$n_{Transistoren,Speicher} \approx (4 + 2 \cdot n_{port}) \cdot n_{bit} \tag{4.5}$$

In ***Anhang D*** werden veröffentlichte Lösungen analysiert, indem für *On-Chip-Speicher* veröffentlichte Gesamtzahlen für die Transistoren anteilig auf im Layout identifizierte Speichermodule umgerechnet werden und auf die angegebenen Speichergrößen normiert werden. Trotz erkennbarer Ungenauigkeiten in der Meßmethode und Unsicherheiten zu einzelnen Implementierungen bestätigt diese Analyse über die resultierende Mittelwerte Gleichung (4.5).

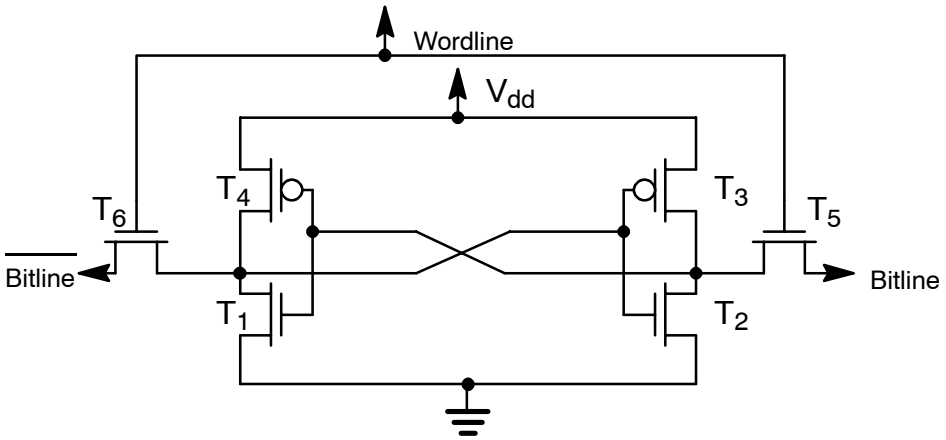


Bild 4. 8 . 6 Transistor *SRAM*-Zelle, *Single-Port* Speicher.

Zu der in Bild 4. 8 dargestellten SRAM-Zelle gibt es Alternativen. Sofern der eingesetzte Halbleiterprozeß eine zweite hochohmige Polysiliziumebene besitzt, können die *PMOS*-Transistoren T_3 und T_4 durch in der zweiten Polysiliziumebene gefertigte Widerstände ersetzt werden, was zu einem kompakteren Layout führt. Dann gibt es seit einigen Jahren sogenannte *1 Transistor-SRAM-Zell-Macros* der Firma Mosys [66], die von zahlreichen Halbleiterherstellern eingesetzt werden. Es handelt sich hierbei um *DRAM-Speicher* mit einer integrierten Refresh-Logik und einer zu *SRAM-Speichern* vergleichbaren Ansteuerung. Wenn man hier mit den verfügbaren Modellierungswerkzeugen Flächenberechnungen durchführt [91], ist tendenziell erkennbar, daß die *1 Transistor-SRAM-Zelle* eine Fläche belegt, die etwa der Hälfte einer *6 Transistor-SRAM-Zelle* entspricht.

4 .3 .3 Verknüpfung von Algorithmen mit dem generischen Architekturmodell

In Kapitel 2 .2 , Bild 2. 1 ist ein generisches Architekturmodell eingeführt worden, mit dem die Architektur eines *VLSI-Prozessors* hierarchisch spezifiziert werden kann. Eine Prozessorarchitektur wird charakterisiert durch *Processing Elemente (PEs). PEs* enthalten Einheiten, die den Programmablauf steuern *(CU: Control Unit)*, Verarbeitungseinheiten für arithmetische und logische Operationen *(PU: Processing Unit)* und lokale Speicher *(LM: Local Memory)*. Eine *PU* kann als *PE* interpretiert werden. Mit diesem Schritt läßt sich eine geschlossen hierarchische Modellierung einer Architektur spezifizieren.

In Bild 4. 9 wird eine Schnittstelle zwischen zu verarbeitenden Algorithmen und einer Prozessorarchitektur definiert, die sich an den in Kapitel 2 .4 eingeführten analytischen Modellfunktionen und ihrer Erweiterung um *Fuzzy- Arithmetik* (Kapitel 3) orientiert. Ein Algorithmus sei durch eine *Task* T_i beschrieben, die wiederum in *Subtasks* zerlegt werden kann.

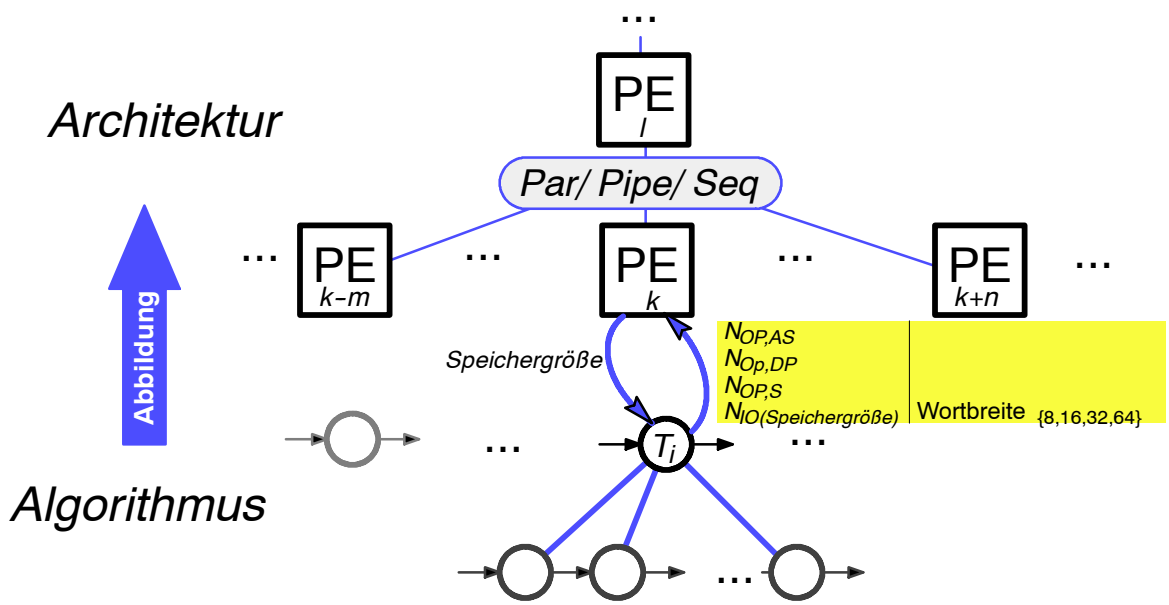


Bild 4. 9 . Abbildung von *Tasks (T)* eines Algorithmus auf *Prozessorelemente (PEs)* einer Architektur. Zusammenfassung der Verarbeitungszeiten für Parallelverarbeitung *(Par*, Pipeline-Verarbeitung *(Pipe)* oder sequentielle Verarbeitung *(Seq)*.

Für die Schnittstelle zwischen einem Algorithmus und einer Architektur gibt es verschiedene, grundlegende Möglichkeiten. Eine Vielzahl bekannter Lösungen basieren auf *Assignment*- und *Scheduling*-Ansätzen [80]. Hier wird versucht, eine Menge von *Tasks* den verfügbaren *Processing Elements* zuzuweisen und den zeitlichen Ablauf zu optimieren. In [80] wird gezeigt, daß insbesondere heterogene Multiprozessoranordnungen so recht gut optimiert werden können. Es gibt aber auch alternative Untersuchungen [41], die zeigen, daß einfachere *Assignment*-Ansätze zu ähnlich guten Architekturbewertungen führen, sofern die Granularität der Abbildung von *Tasks* auf Prozessoren (Verhältnis zwischen Rechenzeit zu Kommunikationszeit) relativ groß ist. Hier werden dann unter Vernachlässigung der zeitlichen Abläufe die von Algorithmen geforderten Rechenleistungen und Transferraten in Beziehung gesetzt mit der von einer Architektur bereitgestellten Rechenleistung und ihrer I/O-Bandbreite. Da bei typischen Videocodierungsverfahren relativ kleine Datenblöcke einmal in einen Prozessor geladen und dann in längeren Zeitintervallen eigenständig bearbeitet werden, kann hier generell eine höhere *Task*-Granularität erwartet werden. Daher soll im Folgenden die einfachere, auf *Assignments* basierende Lösung, zur Rechenzeitbestimmung angewendet werden.

Die Schnittstelle zwischen einer *Task* und einem *PE* wird über wenige Parameter beschrieben, die in Tabelle 4.2 und Bild 4. 6 eingeführt wurden. Diese Parameter werden unter Angabe der jeweils von der Task geforderten Genauigkeit der Eingangsdaten angegeben.

Die Rechenzeit pro Bildpunkt, die ein PE für Datenpfadoperationen *(DP)*, skalare Operationen *(S)* und für die Kommunikation *(IO)* aufbringen muß, ergibt sich dann als :

$$T_{PE,DP,Task} = N_{OP,DP,Task} \cdot \frac{N_{Cycles,PE,DP}}{f_{CLK}} \tag{4.6}$$

$$T_{PE,S,Task} = N_{OP,S,Task} \cdot \frac{N_{Cycles,PE,S}}{f_{CLK}} \tag{4.7}$$

$$T_{PE,IO,Task} = N_{IO}Task \cdot \frac{N_{Cycles,PE,IO}}{R_{IO,PE}} \tag{4.8}$$

$N_{Cycles,PE,x}$ stellt die jeweils benötigte Anzahl an Taktzyklen dar. f_{CLK} ist die Taktrate. Die für ein *PE* und eine *Task* erforderliche Rechenzeit ergibt sich zu :

$$T_{PE,Proc,Task} = \begin{cases} Max(T_{PE,DP,Task}, T_{PE,S,Task}) & \text{, für Datenpfadoperationen parallel oder pipelined} \\ T_{PE,DP,Task} + T_{PE,S,Task} & \text{, für Datenpfadoperationen sequentiell} \end{cases} \tag{4.9}$$

Die Gesamtrechenzeit eines PEs mit zugewiesenen Tasks ist dann unter Berücksichtigung der externen Kommunikation (IO) :

$$T_{PE,Task} = \begin{cases} Max(T_{PE,IO,Task}, T_{PE,Proc,Task}) & \text{, für Verarbeitung und IO gleichzeitig} \\ T_{PE,IO,Task} + T_{PE,Proc,Task} & \text{, für Verarbeitung und IO sequentiell} \end{cases} \tag{4.10}$$

Werden Rechenzeiten in der Hierarchie über mehrere PE-Ebenen zusammengefaßt, soll die folgende Beziehung verwendet werden:

$$T_{PE} = \begin{cases} \underset{i=N}{Max}(T_{Sub,PE,1}, ..., T_{PE,N}) & \text{, für Sub-PEs parallel oder pipelined} \\ \sum_{i=1} T_{Sub,PE}(i) & \text{, für Sub-PEs sequentiell} \end{cases} \tag{4.11}$$

Der modellierte Gesamtdatendurchsatz, der für ein Verfahren erzielbar ist, ergibt sich als:

$$R_{S,Modell} = \frac{1}{T_{PE}} \tag{4.12}$$

Als alternativer Ansatz wird bei Veröffentlichungen von Videosignalprozessoren oft eine Leistungsangabe als skalare Operationsrate ($R_{PE,OpAS}$) angegeben, die meist in ***MOPS*** *(Mega Operationen pro Sekunde)* oder ***GOPS*** *(Giga Operationen pro Sekunde)* spezifiziert wird:

$$R_{PE,OpAS} = f_{CLK} \cdot \sum_{i=1}^{N_{Funktionseinheiten}} N_{PE,OpAS}(i) \tag{4.13}$$

Ein alternativer Ansatz zur Bestimmung des Datendurchsatzes ergibt sich, wenn die in Gleichung (4.1) definierte Zusammenfassung aller Operationen (in Datenpfaden und in Skalareinheiten) als skalare Operationen vorausgesetzt wird. Mit Gleichung (4.1) kann dann eine notwendige Echtzeitbedingung spezifiziert werden :

$$R_{PE,OpAS} \geq R_{OP,AS} \tag{4.14}$$

Mit Ungleichung (4.14) und Gleichung (4.1) ergibt sich dann ein maximaler, von einer *VLSI*-Architektur nicht überschreitbarer Datendurchsatz von :

$$R_{S,OpAS,max} = \frac{f_{clk}}{N_{OP,AS}} \cdot \sum_{i=1}^{N_{Funktionseinheiten}} N_{PE,OpAS}(i) \tag{4.15}$$

Der Vorteil im Einsatz von $\boldsymbol{R_{PE,OpAS}}$ und $\boldsymbol{R_{S,OpAS,max}}$ liegt in der einfachen Anwendbarkeit für einen ersten Vergleich der Performance von *ASICs*. Daß $\boldsymbol{R_{S,OpAS,max}}$ den Datendurchsatzanforderungen einer zu verarbeitenden Anwendung entspricht, ist aber allenfalls eine notwendige und keinesfalls eine hinreichende Bedingung. Insbesondere dann, wenn ein *ASIC* die von einer Anwendung geforderten Operationstypen nicht unterstützt, wird er trotz höchster Operationsraten die Anwendung nicht in Echtzeit verarbeiten können.

4 .4 Abbildung auf eine *ASIC*-Zieltechnologie

In diesem Abschnitt soll der technologieabhängige Einfluß auf Performance und Kosten bei der Abbildung einer *VLSI-Architektur* in eine Zieltechnologie untersucht werden. Die Performance soll hier über die erzielbare Taktrate mit einer Zieltechnologie verknüpft werden. Die Kostenmodellierung konzentriert sich auf die Schaltkreisgröße. Hier sind weitere Kriterien sinnvoll anwendbar, wie z. B. die Leistungsaufnahme eines Schaltkreises, wenn er für batteriebetriebene Anwendungen eingesetzt werden soll.

4 .4 .1 Bestimmung der Taktrate

In Kapitel 4 .3 .2 ist ein technologieunabhängiger Ansatz zur Bestimmung der Periodendauer der Verarbeitungseinheiten eingeführt worden. Vereinfachend wird die erzielbare Periodendauer als Vielfaches einer Standard-Gatterverzögerungszeit angegeben ($n_{Gatterverzögerungen}$). Bezugspunkt ist die Verzögerungszeit eines *NAND2-Gatters*, auf die die modellierten Syntheseergebnisse in Bild 4. 7 vom eingesetzten Synthesetool normiert wurden. Eine für die Verarbeitungseinheiten erzielbare Taktrate ergibt sich dann vereinfacht modelliert mit einer Gatterverzögerungszeit $t_{Gatterverzögerungszeit}$ zu

$$f_{CLK} = \frac{1}{n_{Gatterverzögerungen} \cdot t_{Gatterverzögerungszeit}} \tag{4.16}$$

In der Praxis kann das Problem auftreten, daß auf Grund von kapazitiven Lasten und wegen abweichenden Skalierungen von *Setup-* und *Hold-Zeiten* an den Registerstufen die in Gleichung (4.16) angegebene Taktrate in einer neuen nicht im gleichen Maße steigt, wie die Gatterverzögerungszeit $t_{Gatterverzögerungszeit}$ sinkt.

4 .4 .2 Besondere Bedeutung der Schaltkreisfläche für die Realisierungskosten

Die Herstellungskosten eines funktionsfähigen Schaltkreises hängen von einer Vielzahl von Einflußgrößen ab [16], S. 83-90. Zunächst müssen die Herstellungskosten eines Wafers berücksichtigt werden. Diese können dann auf die pro Wafer herausschneidbaren funktionsfähigen *Dies* (*Die* = Schaltkreis ohne Gehäuse) aufgeteilt werden. Der Anteil an funktionsfähigen *Dies* (Ausbeute *Yield : Y*) ergibt sich zu:

$$Y = e^{-A \cdot D} \qquad \textit{A: Größe eines Schaltkreises, D: Defektdichte} \tag{4.17}$$

Gleichung (4.17) zeigt, daß die Ausbeute mit sinkender *Die-Fläche* steigt. Indirekt sinkt dann auch der Anteil der auf den einzelnen Schaltkreis umzulegenden Herstellungskosten eines Wafers. Zusätzliche Kostenanteile für einen Schaltkreis ergeben sich noch für das Gehäuse, die Verpackung und für den Test eines Schaltkreises. Im Hinblick auf günstige Herstellungskosten hat das Ziel einer möglichst kleinen Schaltkreisfläche oder einer auf die Performance bezogenen, angemessenen Fläche eine große Bedeutung.

Die Abschätzung der Schaltkreisfläche während der Konzeptphase eines applikationsspezifischen Schaltkreises ist relativ schwierig, da man oft zu einem frühen Zeitpunkt, d. h. noch vor den Entwurfsschritten, wie der Schaltungssynthese kaum weiß, wieviele Transistoren oder Gatteräquivalente für einzelne Module benötigt werden. Darüber hinaus bestehen auch bei bekannten Transistorzahlen noch genügend Unwägbarkeiten in der Flächenabschätzung, die unter anderem auch in der Komplexität des Entwurfsablaufes begründet sind.

4 .4 .3 Stand der Technik zur Schätzung von Schaltkreisflächen

Wenn man in der Konzeptphase eines *ASIC*-Entwurfes die Schaltkreisgröße abschätzen will, ist man in der Regel auf Herstellerdaten angewiesen. Hier gibt es Angaben zu Transistor- oder Gatterdichten und zur Realisierungsgröße von Speichern. Ein Beispiel findet man im Internet bei dem taiwanesischen Hersteller UMC [91] mit dem *Chipsizer Tool*. Dort kann man Gatterzahlen, Speichergrößen und Pin-Zahlen angeben und erhält eine erste Abschätzung zur späteren Schaltkreisgröße und zu den möglichen Fertigungskosten in einem ausgewählten Halbleiterprozeß.

Neben der Betrachtung für einen spezifischen Halbleiterprozeß ist es häufig auch von Interesse, die Daten bestehender Lösungen auf eine einheitliche Strukturgröße zu normieren. Alternativ kann es auch gewünscht sein, zu untersuchen, welches Potential eine Architektur bei zukünftigen Technologieschritten bietet. Ein Modellansatz zur Abschätzung der Schaltkreisfläche sollte daher auch die Skalierung auf neue Technologien berücksichtigen.

Ein einfacher Ansatz für Mikroprozessoren basiert auf ausgemessenen Transistordichten der Prozessoren SPARC 64 und HP PA 8000 und skaliert sie quadratisch auf einheitliche oder neue Prozeßgrößen [81]. In [57] wird mit einer Auswertung bekannter Lösungen gezeigt, daß insbesondere eine quadratische Skalierung mit der Strukturgröße für Mikroprozessoren zu optimistisch ist und sich die in Tabelle 2.1 dargestellten, von Fu [18] vorgeschlagenen, Skalierungsexponenten als passender erweisen.

Die in [57] dargestellte Untersuchung für Mikroprozessoren hat gezeigt, daß vorgeschlagene Lösungsansätze widersprüchlich sind und im Einzelnen falsch sein können. Die Lösung der bestehenden Widersprüche oder die Verifikation bekannter Ansätze erfordern daher eine genauere Analyse bekannter Lösungen.

4 .4 .4 Analyse realisierter applikationsspezifischer Schaltkreise

In diesem Abschnitt wird untersucht, welche Transistordichten in der Realität bei applikationsspezifischen Schaltkreisen erreicht werden. Ziel ist die Ableitung eines geeigneten Modells, das die Konzeption neuer Lösungen möglichst gut unterstützt und das auf neue VLSI-Technologien angewendet werden kann. Die Untersuchung konzentriert sich auf Prozessoren für die Videosignalverarbeitung. Hier muß im Hinblick auf Massenanwendungen mit niedrigen Herstellungskosten auf möglichst kleine Schaltkreisflächen geachtet werden, z. B. MPEG2-Decoder für DVD-Player oder MPEG2-Codecs für digitale Videorecorder. Da gleichzeitig aber auch Echtzeitanforderungen über hohe Rechenleistungen erfüllt werden müssen, besteht bei Videosignalprozessoren ein großer Bedarf an besonders effizienten Lösungen. Auf Grund der kontinuierlichen Weiterentwicklung der Verfahren bieten sich zunehmend programmierbare Prozessoren für die Implementierung von Videocodierungsanwendungen an.

Wenn man die Literatur auf dem Gebiet der programmierbaren Videosignalprozessoren seit 1990 analysiert, findet man etwa 70 Literaturstellen, aus denen man versuchen kann, Schaltkreis-Layouts im Hinblick auf Transistordichten zu analysieren. Zum Teil fehlen aber entweder Schaltkreisfotos oder detailliertere Angaben zu verwendeten Halbleiterprozessen. Bei genauerer Betrachtung sind daher nur etwa 30 Literaturstellen auswertbar, die im Folgenden näher untersucht werden, [2], [3], [5], [10], [12] [21], [27], [29], [30], [34], [36], [40], [44], [47] ,[50], [51], [60], [61], [62], [67], [68], [69], [70], [71], [87], [88], [89], [90], [35]. Bei diesen Veröffentlichungen handelt es sich um *Semi-Custom Designs,* bei denen Module für Arithmetik und die Steuerung der Verarbeitungsabläufe mit Standardzellen realisiert wurden. Auf den Schaltkreisen integrierte Speicher wurden in der Regel als fertige Macro-Blöcke integriert.

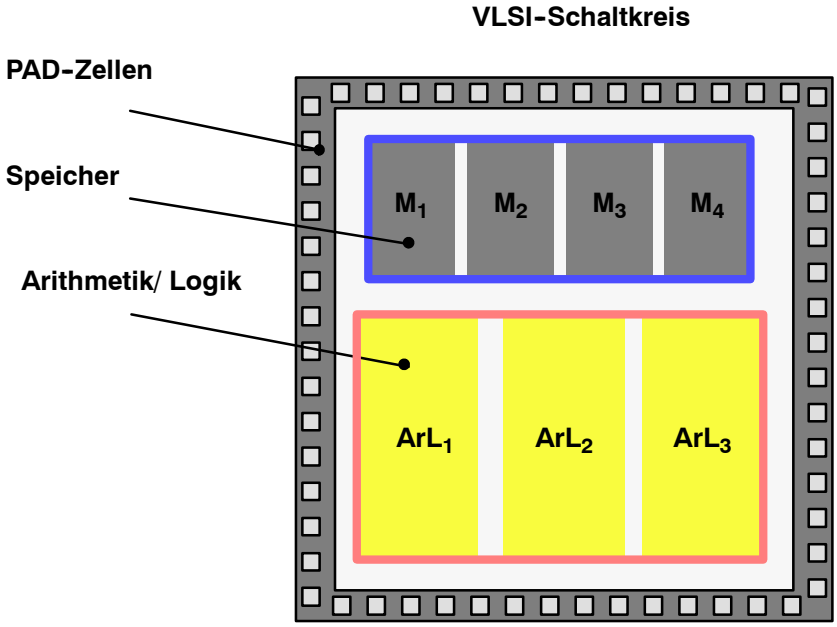


Bild 4. 10 . Klassifikation von Layout-Modulen eines Schaltkreises nach *Speicher (M), Arithmetik und Logik (ArL)* und *PAD-Zellen*.

Bild 4. 10 zeigt die für die folgende Analyse angewendete Klassifikation von Layout-Daten. Es wird hier unterteilt in Speicher, Arithmetik und Logik und PAD-Zellen, die jeweils separat analysiert und modelliert werden sollen. Verbleibende, nicht gekennzeichnete Flächen dienen der globalen Verdrahtung zwischen den Modulen auf dem Chip.

Bild 4. 11 zeigt den Ablauf der Analyse veröffentlichter VLSI-Schaltkreise. Im ersten Schritt werden Schaltkreis-Layout-Fotos in das Programm *Layout Analyse* eingelesen. Hier werden dann Flächen der Module ausgemessen und tabellarisch erfaßt. Über alle berücksichtigten Schaltkreise wird dann aus einer Gesamttabelle automatisch eine Datenbasis in einer relationalen Datenstruktur generiert, die vom *VSP Decision Programm* importiert und dort ausgewertet werden kann.

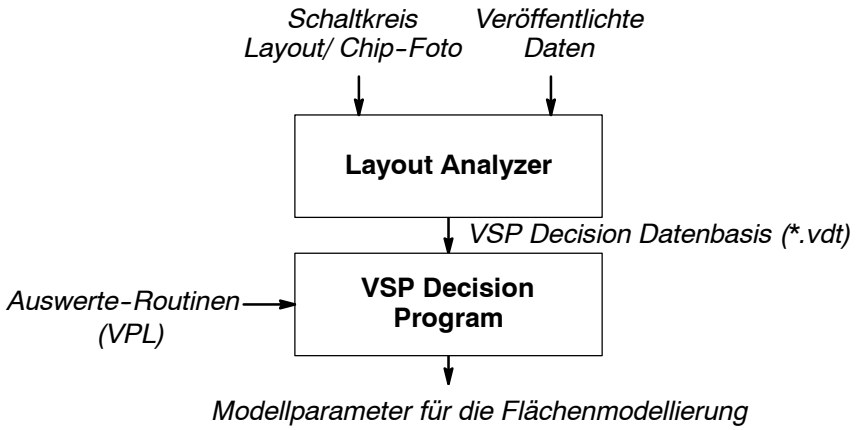


Bild 4. 11 . Ablauf der Auswertung von Schaltkreis-Layouts mit den Programmen ***Layout Analyzer*** und ***VSP Decision Program***.

Da die untersuchten Schaltkreise in verschiedenen Strukturgrößen realisiert sind, die durch die Gatter-Länge ***s*** gekennzeichnet sind, soll im ersten Schritt eine geeignete Umrechnung der erfaßten Layout-Daten auf eine einheitliche Technologie angestrebt werden. In der Praxis wird hier häufig eine Skalierung auf eine Zieltechnologie vorgeschlagen [74], die meist auf den von Mead und Conway eingeführten Skalierungsgesetzen basieren[59]. Danach sinkt eine Schaltkreisfläche A_0 bei einem Übergang von einem Prozeß mit einer Gatter-Länge s_0 zu einem neuen Prozeß mit s_1 quadratisch mit der Strukturverkleinerung, wenn alle Geometrieabmessungen um den selben Faktor verkleinert werden (ideale Skalierung) :

$$A_1 = A_0 \cdot \left(\frac{s1}{s0}\right)^2 \tag{4.18}$$

Meist wird die in Gleichung (4.18) dargestellte quadratische Skalierung für Mikroprozessoren vorgeschlagen [81], [33]. Nach Kapitel 2 .1 gibt es zumindest für Mikroprozessoren auch Hinweise auf andere Skalierungsexponenten [18], [57].

Auf Grund der zumindest für Mikroprozessoren widersprüchlichen Standpunkte zur Skalierung von *VLSI*-Schaltkreisen und der Bedeutung von Skalierungsschritten für eine einheitliche Modellierung von *VLSI*-Technologien, soll untersucht werden, wie sich schritthaltend mit der Strukturverkleinerung *VLSI*-Schaltkreisflächen verringern, bzw. die Transistordichten erhöhen.

Bild 4. 12 zeigt die aus den Layouts der untersuchten Schaltkreise ermittelten Transistordichten, die sich aus dem Quotienten der jeweils veröffentlichten Transistorzahlen für Arithmetik und den im Layout ausgemessenen Flächen für Arithmetikeinheiten ergeben.

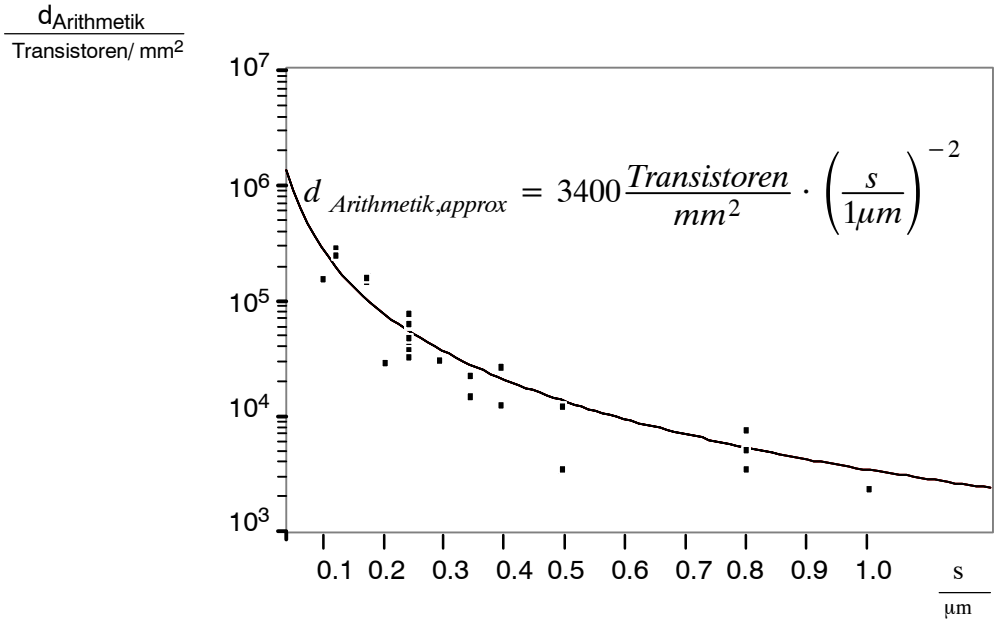


Bild 4. 12 . Transistordichten für Arithmetik und Logik $d_{Arithmetik}$ in Abhängigkeit von der Gatter-Länge ***s***.

In Bild 4. 13 wird die gleiche Untersuchung auf die Skalierung für On-Chip-Speichermodule angewendet. Es zeigt sich, daß sich auch die Skalierung von Speichermodulen an der quadratischen Skalierung orientiert.

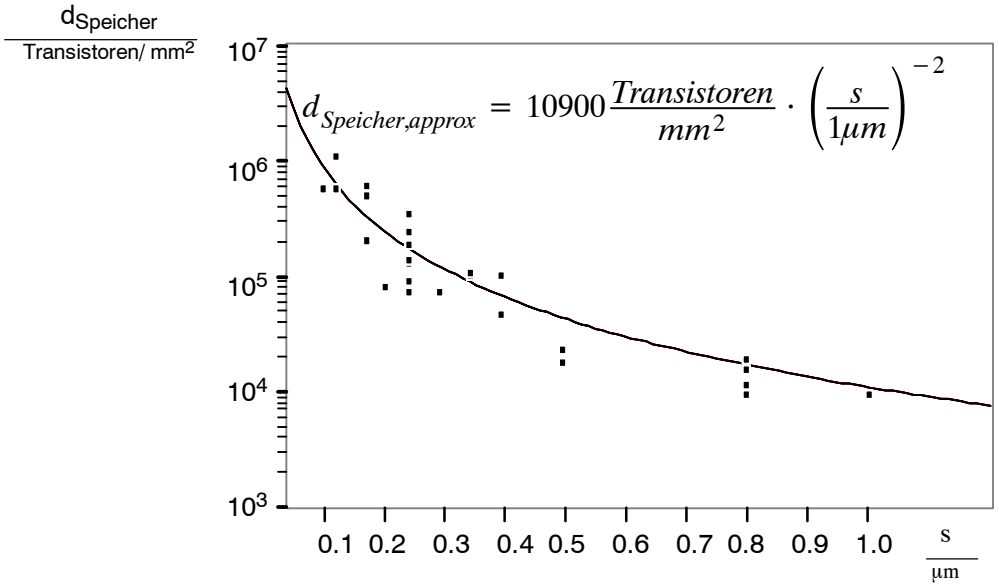


Bild 4. 13 . Transistordichten für Speicher $d_{Speicher}$ in Abhängigkeit der Gatter-Länge **s.**

Die Approximation der Meßwerte durch Trendlinien mit einer quadratischen Skalierung führt zu den eingezeichneten Ergebnissen

$$d_{Arithmetik,approx} = 3400 \cdot s^{-2} \tag{4.19}$$

$$d_{Speicher,approx} = 10900 \cdot s^{-2} \tag{4.20}$$

Die Bilder 4. 12 und 4. 13 zeigen, daß im Mittel eine quadratische Skalierung für *ASICs* zutreffend ist. Daher sollen im weiteren Verlauf dieser Arbeit ausgemessene Transistordichten mittels einer quadratischen Skalierung auf eine Strukturgröße von 0.1 µm normiert werden.

In Bild 4. 14 ist das Ergebnis der Normierung der Transistordichten auf 0.1 µm für analysierte arithmetische Einheiten dargestellt. Es ist erkennbar, daß die normierten Transistordichten stark variieren. Sofern die angewendete quadratische Skalierung korrekt ist, muß die starke Streuung der normierten Werte eigene Ursachen haben, die diskutiert werden sollen. Ein möglicher Grund liegt in den von Halbleiterherstellern angebotenen unterschiedlich charakterisierten Prozessen [91], deren Standardzellen entweder in Richtung einer kompakten Lösung gehen oder die für schnelle aber weniger kompakte Realisierungen ausgelegt sind. Weitere Ursachen können im Entwurfsprozeß liegen. In einigen Fällen werden vorhandene *VLSI*-Entwürfe in eine neue Technologie überführt, indem die Dimensionen der Transistor-Gates verkleinert werden *(Gate Shrinking)*. Die Plazierung und das übergeordnete Layout bleiben unverändert. Als Gewinn ergibt sich dann eine höhere Taktrate in der neuen Technologie, während die Schaltkreisgröße nicht sinkt und die Transistordichte nicht ansteigt. In einigen Fällen

führen auch begrenzte Resourcen für den Schaltkreisentwurf dazu, daß die maximal möglichen Transistordichten einer Technologie nicht erreicht werden.

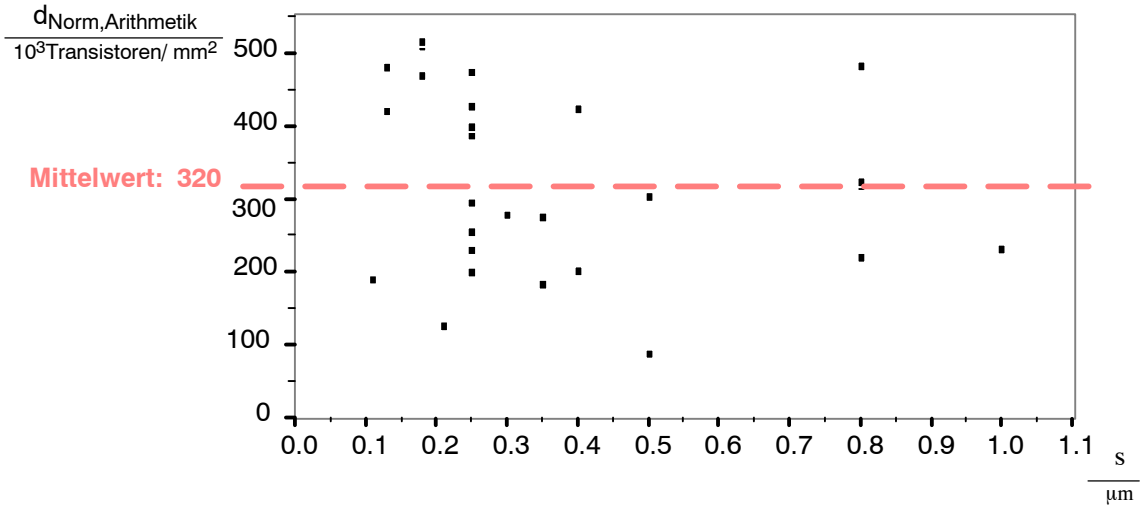


Bild 4. 14 . Normierung der Transistordichten für arithmetische und logische Einheiten von einer Strukturgröße *s* auf ***s'***=0.1µm.

Ein weiterer Aspekt, der zu Variationen in den Transistordichten bei einer Strukturgröße *s* führen kann, liegt in der unterschiedlichen Nutzung der zur Verdrahtung der Zellen verfügbaren Metallisierungsebenen. Die in dieser Arbeit untersuchten *ASICs* nutzen 2 Metallisierungsebenen [67] bis zu 8 Metallisierungsebenen [35].

Es sollen nun die normierten Transistordichten der analysierten *Semicustom-ASICs* über der Anzahl der Metallisierungsebenen grafisch dargestellt werden (Bild 4. 15). Bei der Betrachtung von Bild 4. 15 stellt sich die Frage, wie die dargestellten Daten im Hinblick auf einen Modellansatz ausgewertet werden können. Ein Modellansatz sollte für zusätzliche Metallisierungsebenen erfassen, wie sich die Transistordichte erhöhen kann.

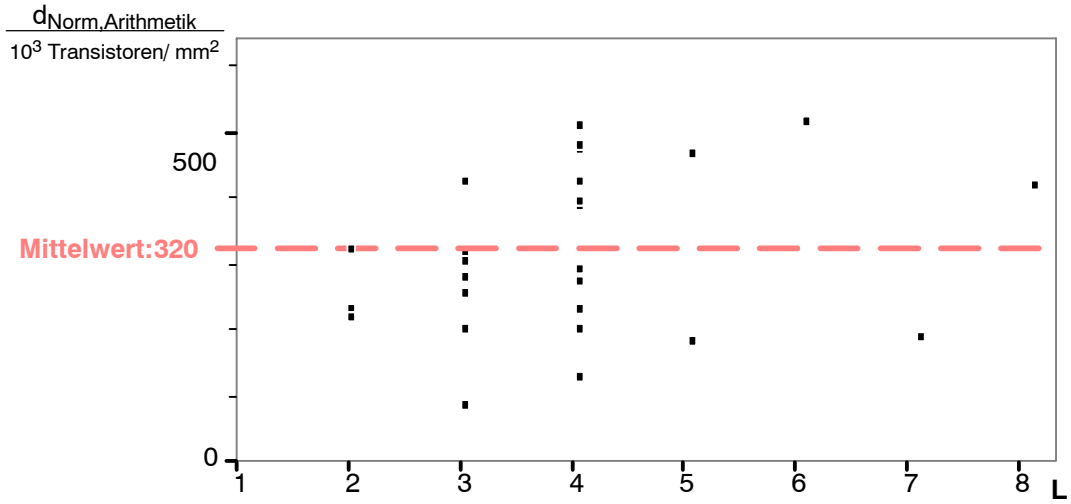


Bild 4. 15 . Auf 0.1 µm normierte Transistordichten für Arithmetik, aufgetragen über der Anzahl der Metallisierungsebenen **L**.

Als erste Idee könnte man jeweils für eine vorgegebene Anzahl der Metallisierungsebenen annehmen, daß Prozesse mit den höchsten normierten Transistordichten in ihrer Ursprungstechnologie die Prozesse vertreten, die unter Einbeziehung der Metallisierungsebenen die höchste Transistordichte ermöglichen und die in der Qualität als vergleichbar angesehen werden können. Wenn man eine Trendlinie für die besten Werte sucht, ergibt sich eine Gerade, die kaum ansteigt (Bild 4. 16). Hier kann in der untersuchten Stichprobe die geringe Anzahl an Lösungen mit 5, 6, 7, oder 8 Metallisierungsebenen (L) dazu geführt haben, daß ab der fünften Metallisierungsebene nicht die maximal erreichbaren Transistordichten erfaßt wurden.

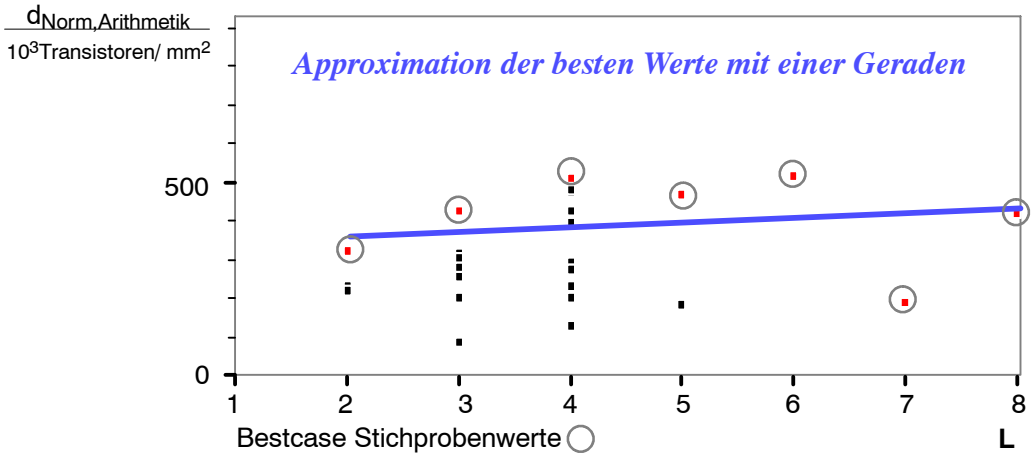


Bild 4. 16 . Auf 0.1 µm normierte Transistordichten für Arithmetik, aufgetragen über der Anzahl der Metallisierungsebenen **L**. Approximation der besten Werte mit einer Geraden.

Wenn man annimmt, daß jede weitere Metallisierungsebene zu einem Anstieg der Transistordichte führen muß, können für eine Modellbildung nur noch die Werte der Stichprobe herangezogen werden, die einen monotonen Anstieg belegen. Unter dieser Voraussetzung bleiben nur drei Stichprobenwerte für 2, 3 und 4 Metallisierungsebenen (L) für die Modellbildung übrig (Bild 4. 17).

Aus den *Bestcase*-Stichprobenwerten für L= 2,3,4 ergibt sich für die Approximation mit einer Exponentialfunktion die in Bild 4. 17 eingezeichnete Trendkurve eines verallgemeinerten *Bestcase-Flächen-Modells (BCF)*.

Nach Herleitung des *BCF-Modells* aus nur 3 Stichprobenwerten konnte eine neue Literaturstelle gefunden werden, die einen weiteren ***ASIC*** für Videosignalverarbeitungsaufgaben *(IMAP CE von NEC)* beschreibt, der L=7 Metallisierungsebenen aufweist [52]. Details zu den Realisierungsdaten dieses Schaltkreises findet man im ***Anhang F***. In Bild 4. 17 ist die normierte Transistordichte für Arithmetik und Logik des *IMAP CE* eingezeichnet. Es ist zu erkennen, daß diese Realisierung dicht an der geschätzten, normierten Transistordichte des *BCF-Modells* liegt.

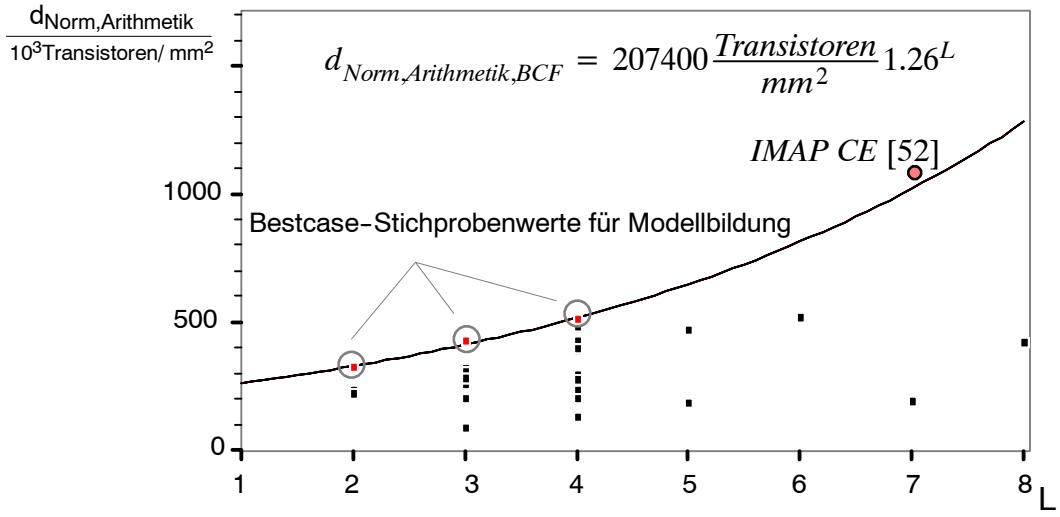


Bild 4. 17 . ***Bestcase Flächenmodell (BCF):*** Auf 0.1 µm normierte Transistordichten für Arithmetik und Approximation der besten Werte mit einer Exponentialfunktion.

Vergleichbar zu arithmetischen und logischen Einheiten kann das ***BCF-Modell*** auch für *On-Chip*-Speichermodule entwickelt werden. Auch in diesem Fall basiert das Modell wieder nur auf 3 Stichprobenwerten (Bild 4. 18).

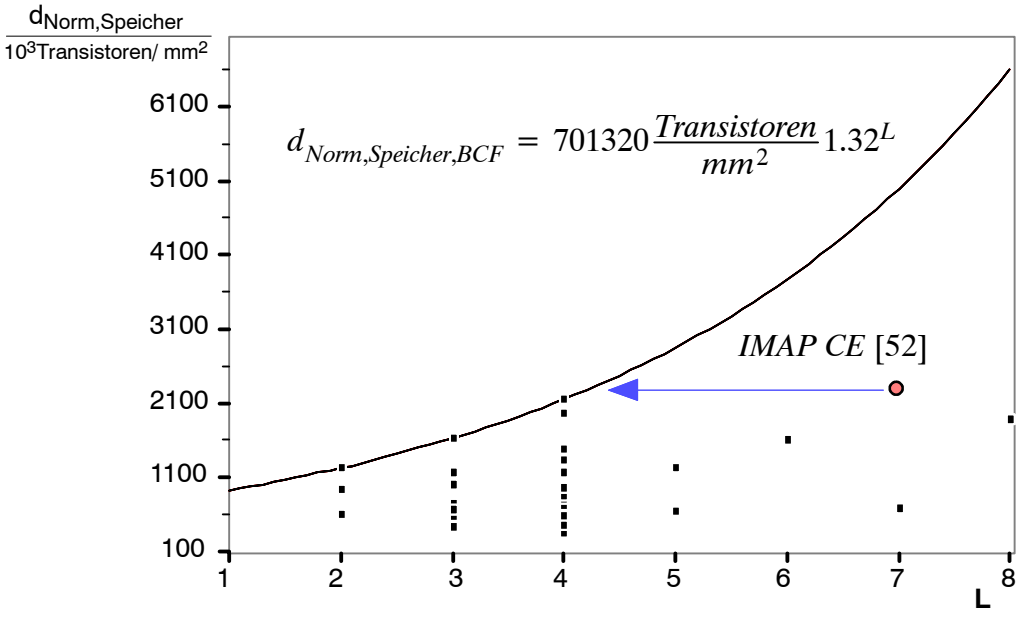


Bild 4. 18 . ***Bestcase Flächenmodell (BCF):*** Auf 0.1 µm normierte Transistordichten für Speicher und Approximation der besten Werte.

Trägt man in Bild 4. 18 die aus Layout-Messungen gewonnene normierte Transistordichte des *IMAP CE* Prozessors ein, ist erkennbar, daß sie äquivalent ist zu einem *Bestcase-Prozeß* mit 4 Metallisierungsebenen. Das ist insofern plausibel, weil *ASIC*-Hersteller häufig in ihren *Design Rules* Beschränkungen in der Nutzung der verfügbaren Metallisierungsebenen für Speicherbereiche definieren.

Nach Arithmetik/ Logik und Speichern fehlt noch die Modellierung der PAD-Zellen. Bild 4. 19 zeigt eine Darstellung der ausgemessenen Größen der PAD-Zellen, aufgetragen über der Strukturgröße *s*. Im Folgenden soll die Größe einer PAD-Zelle durch die in Bild 4. 19 eingetragene Trendlinie approximiert werden.

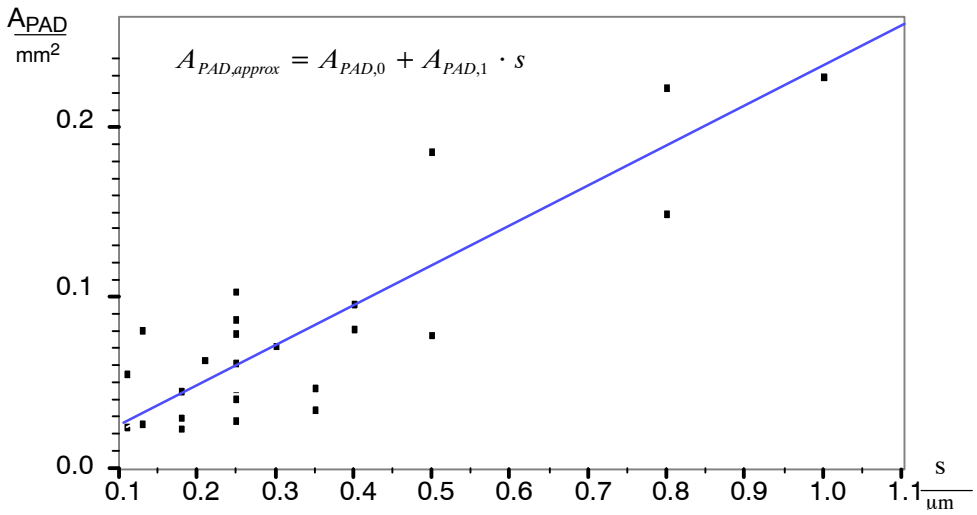


Bild 4. 19 . Modellierung der Fläche von PAD-Zellen, $A_{PAD,0}$= 0.01 mm^2, $A_{PAD,1}$ = 0.233 mm^2/ µm

In Bild 4. 10 verbleiben neben den Flächen für Arithmetik und Logik, Speicher und PAD-Zellen unklassifizierte Bereiche, die für die Verdrahtung zwischen den klassifizierten Modulen genutzt werden. Im Hinblick auf diese Flächen soll ein globaler Verdrahtungsfaktor (GVF) eingeführt werden:

$$GVF = \frac{Fläche_{Schaltkreis} - Fläche_{PAD-Zellen}}{Fläche_{Arithmetik} + Fläche_{Speicher}} \tag{4.21}$$

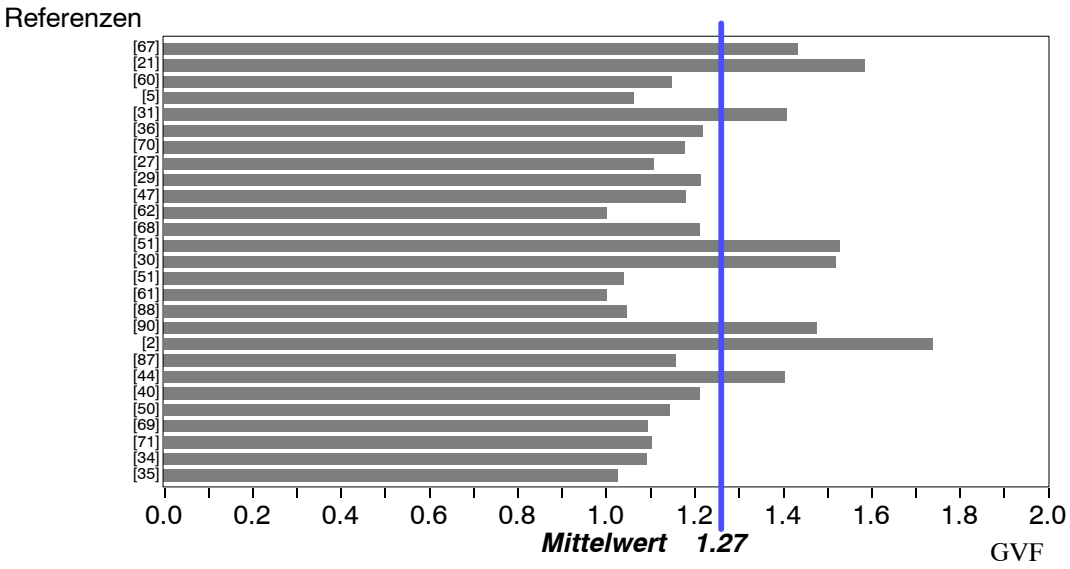


Bild 4. 20 . GVF - Globale Verdrahtungsfaktoren für veröffentlichte Layouts.

Bild 4. 20 zeigt die aus den Layout-Analysen berechneten globalen Verdrahtungsfaktoren. Der resultierende Mittelwert ist GVF = 1.27.

Das Gesamt-Flächenmodell ergibt sich dann wie nachfolgend beschrieben, als

$$DieSize = \left(A_{PAD,0} + A_{PAD,1} \cdot s\right) \cdot N_{Pins+} GVF \cdot \left(\frac{N_{Trans,Arithmetik}}{d_{Norm,Arithmetik}} + \frac{N_{Trans,Speicher}}{d_{Norm,Speicher}}\right) \cdot \left(\frac{s}{0.1\mu m}\right)^2 \quad (4.22)$$

In ***Anhang E*** wird das Modell in einer Übersicht gemeinsam mit den ermittelten Modellparametern dargestellt.

4 .4 .5 Diskussion des entwickelten *Bestcase-Flächenmodells*

Die Entwicklung des *Bestcase-Flächenmodells (BCF)* zeigt ein grundlegendes Problem bei dem Versuch, herstellerunabhängige, generell gültige Modelle zu entwickeln. Von etwa 70 seit 1990 veröffentlichten, programmierbaren *ASICs* für Videosignalverarbeitungsaufgaben sind nur 27 auswertbar, weil bei den übrigen Veröffentlichungen einzelne, relevante Daten, wie die Schaltkreisgröße oder die Strukturgröße fehlen. Von den verbleibenden analysierten 27 Schaltkreisen tragen nur jeweils drei Stichprobenwerte mit maximalen Transistordichten für Arithmetik/ Logik oder für Speicher zur Gewinnung der Modellparameter bei. Bei jährlich nur wenigen neuen veröffentlichten *ASICs* für die Videosignalverarbeitung besteht zur Zeit keine realistische Chance, das *BCF-Modell* statistisch abgesichert nachzuweisen. Es bleibt daher nur der Schluß, daß das hier entwickelte *BCF-Modell* für die Klasse von veröffentlichten *ASICs* für die Videosignalverarbeitung zur Zeit nicht statistisch belegbar ist.

Da die Gültigkeit des vorgeschlagenen Modells nicht belegbar ist, kann man wenigstens versuchen, die Plausibilität des *BCF-Modelles* zu untersuchen oder Widersprüche zu finden, die gegen die Gültigkeit des Modells sprechen. Zunächst soll daher diskutiert werden, ob mit dem *BCF-Modell* die Gesamtfläche für den *IMAP CE Prozessor* [52] in der richtigen Größenordnung errechnet wird. Wenn sich hier schon für eine erste modellierte Lösung die modellierte Schaltkreisfläche stark von der bekannten Größe unterscheiden würde, erwiese sich das *BCF-Modell* als unbrauchbar.

Auch ließe sich das *BCF-Modell* widerlegen, wenn die aktuellen Halbleiterprozesse von vornherein geringere normierte Transistordichten aufweisen, als sie vom *BCF-Modell* für eine größere Anzahl an genutzten Metallisierungsebenen prognostiziert werden. Daher soll das BCF.Modell mit den Daten eines modernen CMOS-Halbleiterprozesses verglichen werden.

Anschließend soll für die untersuchten Prozessoren analysiert und diskutiert werden, warum ggfs. so viele Beispiele in den Transistordichten erheblich schlechter sind, als es mit dem BCF-Modell erwartet würde. Im letzten Schritt sollen dann noch alternative Transistordichtenmodelle eingeführt werden, die im folgenden Kapitel 5 gemeinsam mit dem ***BCF-Modell*** angewendet und untersucht werden sollen.

Modellierung des *IMAP CE Prozessors*

Als erstes Beispiel zur Plausibilisierung des entwickelten Flächenmodells soll der *IMAP CE Prozessors* [52] mit dem *BCF-Modell* modelliert werden *(Anhang F)*. Dieser Prozessor ist nicht in die Modellbildung einbezogen worden und soll als erstes Beispiel für die Überprüfung der Transistordichtenmodellierung eingesetzt werden (Bild 4. 17 , Bild 4. 18). Die normierten Transistordichten liegen recht nahe bei den zugehörigen Modellwerten. Jetzt soll die in Bild 4. 18 angedeutete geringere Nutzung der Metallisierungsebenen für Speicher als Annahme vorausgesetzt werden (4 statt 7 Metallisierungsebenen für Speicher). Die Originalfläche des Schaltkreises ist 121 mm^2. Die mit dem *BCF-Modell* modellierte Fläche liegt bei 128 mm^2. Für dieses Beispiel ergibt sich damit eine Abweichung von 5.87 % zwischen dem bekannten Flächenwert und dem herstellerunabhängigen *Bestcase*-Flächenmodell.

Erreichbarkeit der *BCF-Transistordichten* in aktuellen Halbleiterprozessen

Bild 4. 21 zeigt mit der unteren, durchgezogenen Line das entwickelte *Bestcase*-Flächenmodell *(BCF)*. Zum Vergleich ist beispielhaft aus Herstellerangaben die normierte Transistordichte für einen aktuellen 130 nm CMOS Halbleiterprozeß (max. 8 Metallisierungsebenen) eingezeichnet (obere Linie, [83]). Die veröffentlichten Daten aus diesem Halbleiterprozeß belegen, daß die normierten Transistordichten des *BCF-Modells* in aktuellen CMOS Prozessen durchaus realisierbar sind, was eine weitere, plausible Begründung für das *BCF-Modell* darstellt. Langfristig können die exponentiell über die Anzahl der Metallisierungsebenen ansteigenden Transistordichten die maximale Transistordichten der unverdrahteten Standardzellen nicht überschreiten, was das exponentielle Wachstum in absehbarer Zeit begrenzen wird.

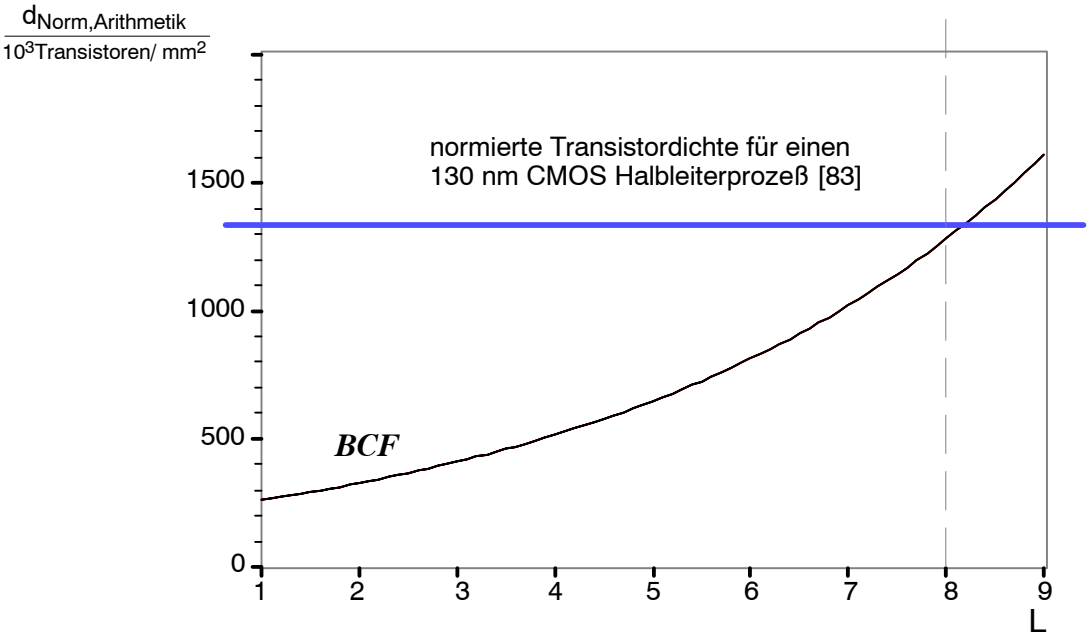


Bild 4. 21 . ***Bestcase Flächenmodell (BCF)*** – Vergleich zu maximaler Transistordichte in einem 130 nm – CMOS Prozeß mit 8 Metallisierungsebenen [83].

Gründe für Realisierungen mit geringeren Transistordichten als nach dem *BCF-Modell*

Es kann eine Reihe von notwendigen (aber nicht hinreichenden) Gründen dafür geben, daß die bei einer Schaltkreisrealisierung erreichten Transistordichten schlechter sind als die theoretisch erreichbaren Werte des *BCF-Modells:*

- Bei Prototypenrealisierungen wird nicht in jedem Fall die Schaltkreisfläche vollständig ausgefüllt, z. B. aus Gründen eines PAD-limitierten Entwurfs [2].
- Eine *ASIC*-Entwicklung wird in einer älteren Technologie begonnen. Bei einem folgenden Wechsel zu kleineren Strukturen werden nur die *Gates* der Transistoren verkleinert (*Shrink*), wodurch sich eine kürzere Entwurfszeit ergibt und der Schaltkreis in der neuen Zieltechnologie aber schneller wird.
- Es werden nicht alle verfügbaren Metallisierungsebenen maximal genutzt.
- Es fehlt Zeit für die Optimierung des Layouts.

Mit Ausnahme von PAD-limitierten Entwürfen [2], wo auf Schaltkreisfotos schnell erkennbar ist, daß eine schlechte Flächenausnutzung vorliegt, wird in Veröffentlichungen nicht berichtet, warum im Layout geringe Transistordichten auftreten. So ist auch bei [47] kein Hinweis zu finden, warum sich mit $d_{Norm,Arithmetik}$ = 126 000 Transistoren/ mm^2 eine sehr geringe normierte Transistordichte ergibt. Unter der Annahme, daß das *BCF-Modell* gültig ist, kann man versuchen, über eine Suche der passenden Strukturgröße s' und einer passenden Anzahl der Metallisierungsebenen L' eine zur ursprünglichen Realisierung äquivalente Technologie bestimmen, die eine maximale Übereinstimmung der nach dem *BCF-Modell* berechneten Schaltkreisgröße mit der originalen Schaltkreisgröße bietet.

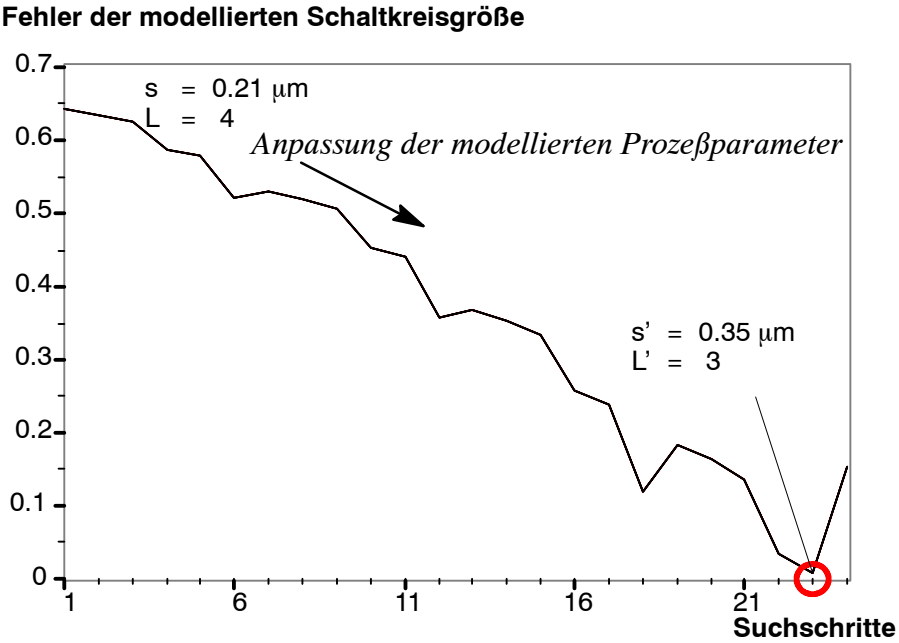


Bild 4. 22 . Suche nach optimalen Prozeßparametern ***s'*** und **L'** mit dem ***Bestcase-Flächenmodell (BCF)*** für einen VLSI-Prozessor [47].

Es ergibt sich ein Suchraum, der auf Grund der geringen Komplexität mit einem einfachen Suchverfahren vollständig durchsucht werden kann. Bild 4. 22 zeigt den Verlauf des relativen Fehlers der mit dem *BCF-Modell* berechneten Schaltkreisgröße bei Variation der Strukturgröße ***s*** und der Anzahl der Metallisierungsebenen **L**. An Stelle der veröffentlichten Technologie mit ***s***=0.21 µm und **L**=4 Metallisierungsebenen deutet das beste Ergebnis der Modellierung der Schaltkreisfläche mit dem *BCF-Modell* darauf hin, daß hier mit einer zu ***s'***=0.35 µm und **L'**=3 äquivalenten Technologie realisiert wurde. Bild 4. 23 zeigt daß unter Annahme der äquivalenten Technologie mit ***s'***=0.35 µm die normierte Transistordichte zu den *Bestcase*-Werten mit 3 Metallisierungsebenen paßt.

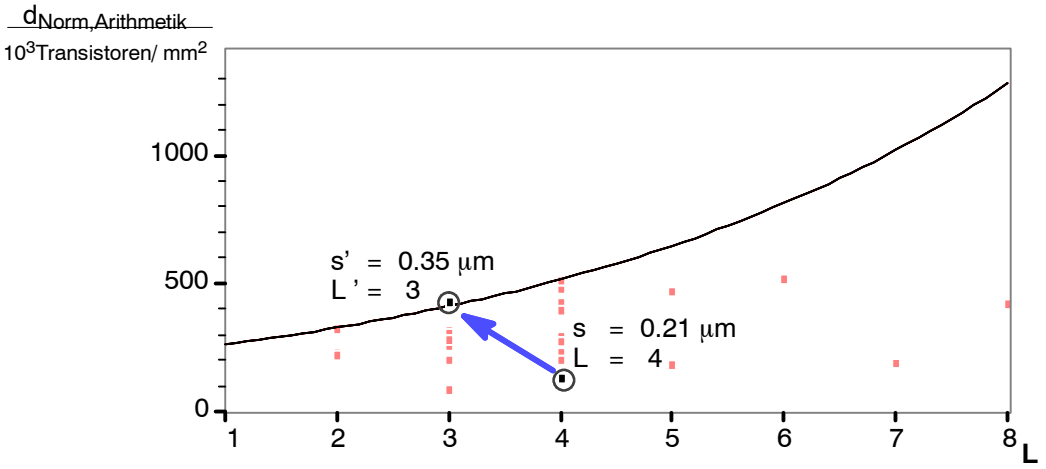


Bild 4. 23 . Ergebnis der normierten Transistordichte nach der Anpassung der Modellparameter an das ***Bestcase Flächenmodell (BCF)*** für einen in [47] beschriebenen VLSI-Prozessor.

Einer der Autoren von [47] hat auf Anfrage berichtet [48], daß das ursprüngliche Design in einer älteren Technologie begonnen wurde und durch *Shrink* der *Gates* umgesetzt wurde auf 0.21 µm. Zusätzlich sei für das endgültige Layout zu wenig Zeit für weitere Optimierungen gewesen. Diese Aussage unterstützt das in Bild 4. 23 dargestellte Ergebnis. In ***Anhang G*** wird für alle analysierte Prozessoren dargestellt, wie sich die veröffentlichten Prozeßparameter auf eine zum *BCF-Modell i*n der Fläche äquivalente Technologie abbilden. Im Folgenden sollen die äquivalenten Prozeßparameter ***s'***, $L'_{Arithmetik}$ und $L'_{Speicher}$ angewendet werden, sofern das *BCF-Modell* eingesetzt wird.

Alternative Modellansätze für Transistordichten

Die bisher in Kapitel 4 .4 .5 diskutierten Aspekte zeigen, daß es plausible Gründe für die Anwendung des *BCF-Modells* gibt. Andererseits kann auf Grund der geringen Stichprobenmenge nicht behauptet werden, daß das Modell generell gültig ist. Grundsätzlich sollte daher das *BCF-Modell* nicht kritiklos und unüberlegt angewendet werden.

Auf Grund dieser Überlegungen soll im Folgenden das *BCF-Modell* mit weiteren Modellansätzen, die im Einzelnen einfacher begründbar sind, diskutiert werden. Alle entwickelten Flächenmodelle werden in einer Übersicht im ***Anhang E*** geschlossen dargestellt.

Im ersten Schritt soll das *BCF-Modell* mit Hilfe der *Fuzzy Set Theorie* auf abgestufte, mögliche Lösungsmengen erweitert werden (Bild 4. 24). Der Zugehörigkeitsgrad der normierten Transistordichte zum möglichen Lösungsraum soll von μ=1 bis μ=0 definiert werden. μ=1 entspricht dem bisherigen *BCF-Modell*. Sofern das *BCF-Modell* stimmt, kann man sicher sein, daß ein *Bestcase*-Prozeß gefunden wird, der diese maximal Transistordichte unterstützt. Für alle geringeren Transistordichten gibt es dann einen linearen Übergang bis zur unteren eingezeichneten Linie (μ=0) in Bild 4. 24 , die sich aus einer Exponentialfunktion als Trendlinie ergibt, die sich an den geringsten, erfaßten Transistordichten orientiert. Das entstehende neue Transistordichtenmodell soll im Folgenden mit *BCMinF* bezeichnet werden.

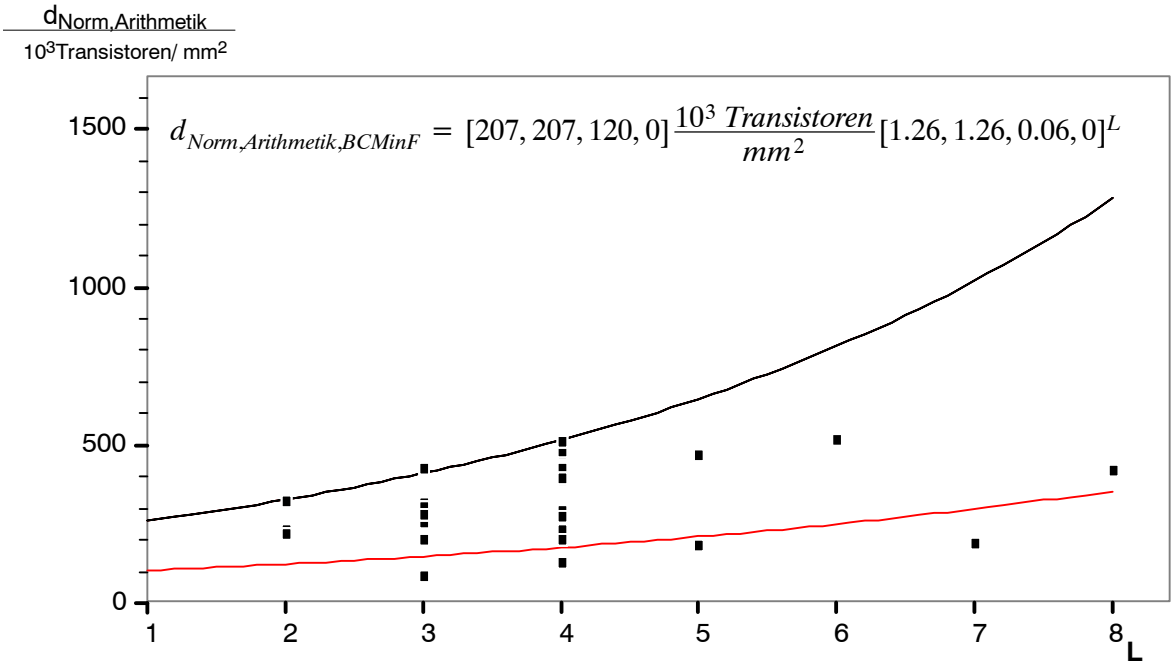


Bild 4. 24 . ***Bestcase Minimum Flächenmodell (BCMinF)*** normierte Transistordichte für Arithmetik.

Eine weitere Modellvariante soll sich am Mittelwert der normierten Transistordichten orientieren. Der Mittelwert wird unabhängig von der Anzahl der eingesetzten Metallisierungsebenen berechnet. Dieses Mittelwertflächenmodell *(MF)* wird in Bild 4. 25 Für arithmetische und logische Einheiten dargestellt. Weitere Details findet man im ***Anhang E***.

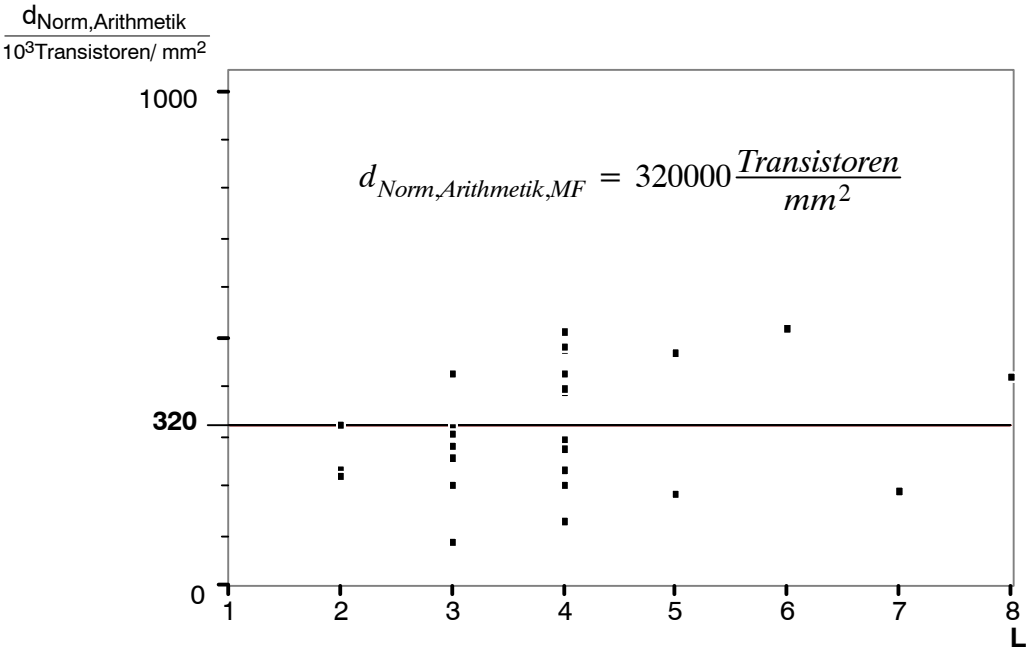


Bild 4. 25 . ***Mittelwert-Flächenmodell (MF)*** - normierte Transistordichte für Arithmetik.

Als letzte Variante soll noch eine Verallgemeinerung des *MF-Modells* eingeführt werden. Hier werden alle Stichprobenwerte auf ein *Fuzzy-Intervall* abgebildet (*Mittelwert-Intervall-Flächenmodell, MIF* Bild 4. 26). Mit diesem Modell werden alle möglichen Einzellösungen einheitlich erfaßt.

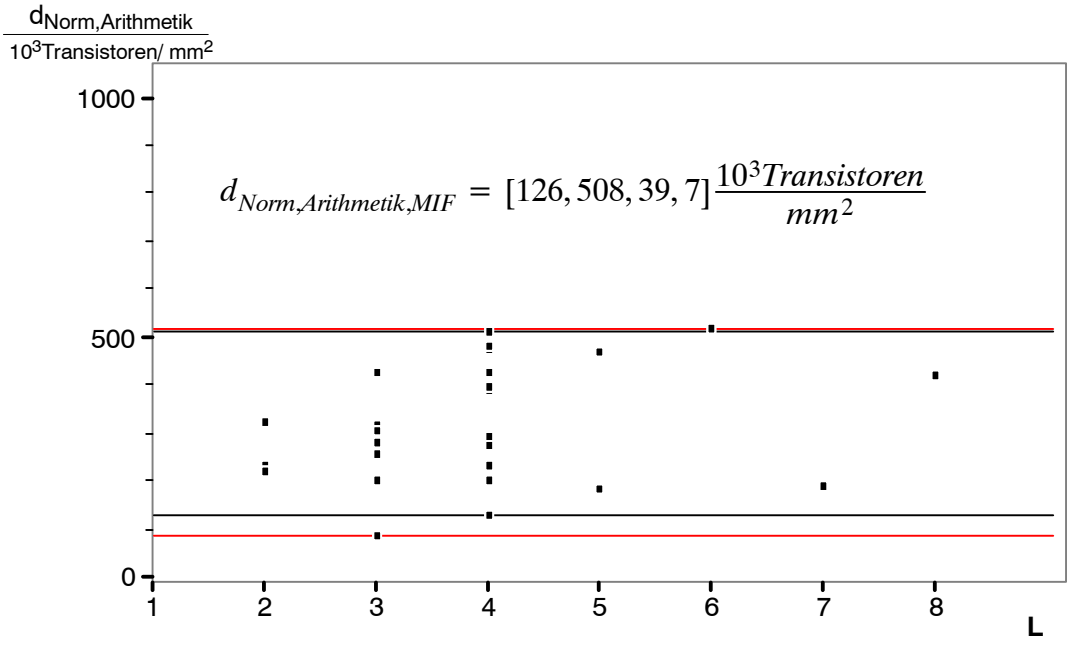


Bild 4. 26 . ***Mittelwert-Intervall-Flächenmodell (MIF)*** normierte Transistordichte (Arithmetik).

5 Anwendung des Modells auf Videosignalverarbeitungs-*ASICs*

In diesem Abschnitt der Arbeit soll der in Kapitel 4 entwickelte Modellansatz (Bild 4. 3) für die Kosten- und Performance-Modellierung applikationsspezifischer *VLSI-Architekturen (ASICs)* angewendet und über den Vergleich mit bekannten Referenzlösungen verifiziert werden. Wenn hier neben der Schaltkreisgröße auch bekannte Performance-Daten berücksichtigt werden sollen, reduziert sich die Stichprobenmenge gegenüber den in Kapitel 4 berücksichtigten Referenzbeispielen. Es ergibt sich für Videocodierungsanwendungen (Bild 4. 4)eine kleinere, analysierbare Stichprobenmenge aus [5], [21], [27], [29], [31], [36], [44], [47], [60], [61], [67], [84], [88]. Unter Berücksichtigung der Angaben für Codierung, Decodierung und vollständige Codec-Anwendungen ergeben sich dann etwa 20 nach Kosten (Schaltkreisgröße) und Performance (Datendurchsatz) analysierbare Varianten, die im Folgenden als Referenzrealisierungen zum Vergleich mit Modellierungsergebnissen dienen sollen. Entsprechend dem in Kapitel 4 entwickelten Modellansatz sollen Modellierungsergebnisse $y_{modelliert}$, aus eingehenden Modellparametern berechnet werden:

$$y_{modelliert} = f(Modellparameter) \tag{5.1}$$

Im Hinblick auf den einheitlichen Vergleich der Güte der Modellierung unterschiedlicher Kosten- und Performance-Kriterien sollen die Fehler modellbasierter Schätzungen relativ zu den bekannten Referenzdaten erfaßt werden. Nach [25] eignet sich als Bewertungskriterium für die Güte eines Schätzers besonders ein mittlerer quadratischer Fehler. Auf Grund der Quadrierung gehen im Vergleich zu einem *mittleren Fehler* oder zu einem *mittleren absoluten Fehler* große Modellabweichungen stärker in das Ergebnis ein als kleine Abweichungen. Das im Folgenden einheitlich eingesetzte Fehlermaß, der mittlere quadratische relative Fehler E_{mittel} ist definiert als:

$$E_{mittel} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_{Referenz,i} - y_{modelliert,i}}{y_{Referenz,i}}\right)^2 \tag{5.2}$$

Zusätzlich soll der Zusammenhang zwischen Referenzdaten und Modellierungsergebnissen an Hand des Korrelationskoeffizienten ϱ untersucht werden:

$$\rho_{Y_{Referenz},Y_{modelliert}} = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(y_{Referenz,i} - Y_{Referenz,M}\right)\left(y_{modelliert,i} - Y_{modelliert,M}\right)}{\sigma_{Y_{Referenz}} \cdot \sigma_{Y_{modelliert}}} \qquad ; M: Mittelwert$$

$$mit: \quad \sigma_X = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - X_M)^2} \qquad X \in \left\{Y_{Referenz}, Y_{modelliert}\right\} \qquad x_i \in \left\{y_{Referenz,i}, y_{modelliert,i}\right\} \tag{5.3}$$

Wenn ein Modell mit der Realität gut übereinstimmt, ergibt sich zwischen Referenz- und Modelldaten als Trend ein linearer Zusammenhang, der sich in einem Korrelationskoeffizienten

$\varrho \approx 1$ widerspiegelt. Wenn statistisch kein linearer Zusammenhang der analysierten Daten erkennbar ist, ergibt sich $\varrho = 0$. Für $\varrho = 0$ ist nicht auszuschließen, daß eine nichtlineare Beziehung zwischen den zu vergleichenden Datensätzen zu finden ist. Im hier betrachteten Modellansatz, der möglichst gut die Referenzdaten modellieren soll, würde ein Ergebnis mit einem kleinen Korrelationskoeffizienten ($\varrho \approx 0$) auf ein unzureichendes Modell hinweisen.

5 .1 Performance-Modellierung

Im Rahmen dieser Arbeit soll der für eine Echtzeitsignalverarbeitungsanwendung erzielbare Datendurchsatz R_S als Performance-Maß einer applikationsspezifischen VLSI-Architektur betrachtet werden. In anderen Zusammenhängen sind andere oder weiter gehende Performance-Kriterien häufig ebenfalls zu berücksichtigen.

Das generelle Ziel für ein Performance-Modell ist ein möglichst einfacher Ansätze mit gleichzeitig hoher Genauigkeit der Ergebnisse. Da sich beide Ziele widersprechen können, gilt es hier, den besten Kompromiß zwischen der Genauigkeit und dem Aufwand eines Modells zu finden. Neben dem aufwendigeren $R_{S,Modell}$, (Gleichung (4.6)-(4.12)) das den Datendurchsatz unter Berücksichtigung des Charakters der Verarbeitung und der Datenzugriffe bestimmt, soll daher auch die stark vereinfachte Performance-Abschätzung $R_{S,OpAS,max}$ (Gleichung (4.13)-(4.15)) angewendet werden. $R_{S,OpAS,max}$ entspricht abgesehen von dem für eine betrachtete Applikation konstanten Faktor dem direkten Vergleich der Operationsraten *(MOPS/ GOPS)* von *VLSI-Prozessoren* ohne Berücksichtigung der Art der Verarbeitung. Detailergebnisse der Modellierung werden in *Anhang H* dargestellt.

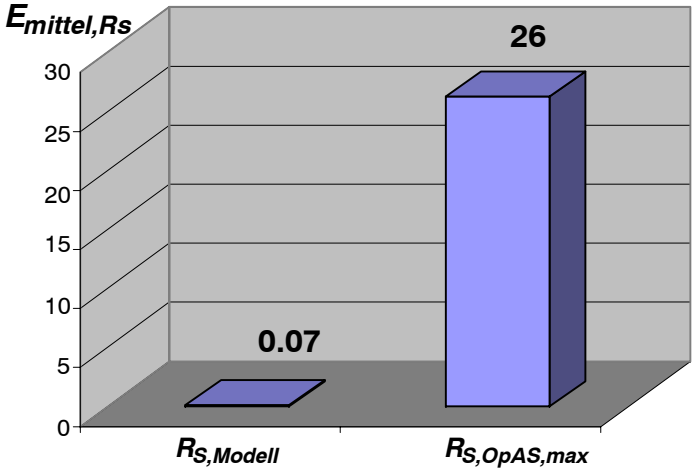


Bild 5. 1 . Mittlerer Fehler $E_{mittel,Rs}$ zu bekannten Performance-Daten (Datendurchsatz R_S).
$R_{S,Modell}$: Modellierung des Datendurchsatzes unter Berücksichtigung der Art der Verarbeitung und der Datenzugriffe (Gleichungen (4.6) - (4.12)).
$R_{S,OpAS,max}$: Vereinfachte, aus dem Verhältnis der *ASIC*-Operationsraten *(MOPS/ GOPS)* zu den Operationsanforderungen gewonnene Abschätzung des maximal erreichbaren Datendurchsatzes (Gleichung (4.15)).
Detaillierte Modellierungsergebnisse findet man in ***Anhang H***.

Bild 5. 1 zeigt den mittlerem Fehler E_{mittel} nach Gleichung (5.2). Es ist zu erkennen, daß der mittlere Fehler für $R_{S,Modell}$ sehr gering ist. Die Anwendung von $R_{S,OpAS,max}$ führt zu einem Fehler, der gegenüber $R_{S,Modell}$ fast um den Faktor 400 größer ist. Auf Grund dieses Faktors muß festgestellt werden, daß die Betrachtung der Operationsraten *(MOPS/ GOPS)* ungeeignet ist, um auf den applikationsspezifisch erzielbaren Datendurchsatz einer Architektur ($R_{S,OpAS,max}$) zu schließen.

Auch wenn es auf Grund des großen Fehlers nicht sinnvoll ist, aus den Operationsraten einer Architektur direkt auf ihre Performance zu schließen, stellt sich die Frage, ob es andere Gründe gibt, die eine Betrachtung von Operationsraten rechtfertigen. Hier sollen die Performance-Daten aus ***Anhang H*** gemäß Gleichung (5.3) mit Hilfe des Korrelationskoeffizienten zwischen bekannten Referenzdaten und modellierten Daten untersucht werden. Neben der direkten Verwendung der Daten soll auch die Korrelation der Rangfolge von Referenzdaten und Modelldaten berechnet werden. Für die Rangfolgebestimmung werden die bekannten und die modellierten Performance-Daten in *Anhang H* jeweils der Größe nach sortiert. Jedem Eintrag wird dann entsprechend seiner Position in der resultierenden Liste ein Rang zugeordnet.

In Bild 5. 2 werden die resultierenden Korrelationskoeffizienten dargestellt. Die aus Modellierungsergebnissen und Referenzdaten berechneten Korrelationskoeffizienten sind $R_{S,Modell}$ = 0.91 und $R_{S,OpAS,max}$ = 0.7. Für beide Modelle ergibt sich eine verbesserte Korrelation mit den Referenzdaten, wenn man die Rangfolgen betrachtet. Bild 5. 2 zeigt, daß das im mittleren Fehler wesentlich schlechtere $R_{S,OpAS,max}$-Modell (Bild 5. 1) in in der Korrelation zu den bekannten Referenzdaten bei dem Übergang zur Rangfolge stärker hinzugewinnt, indem der Korrelationskoeffizient von ϱ = 0.7 auf ϱ = 0.8 steigt.

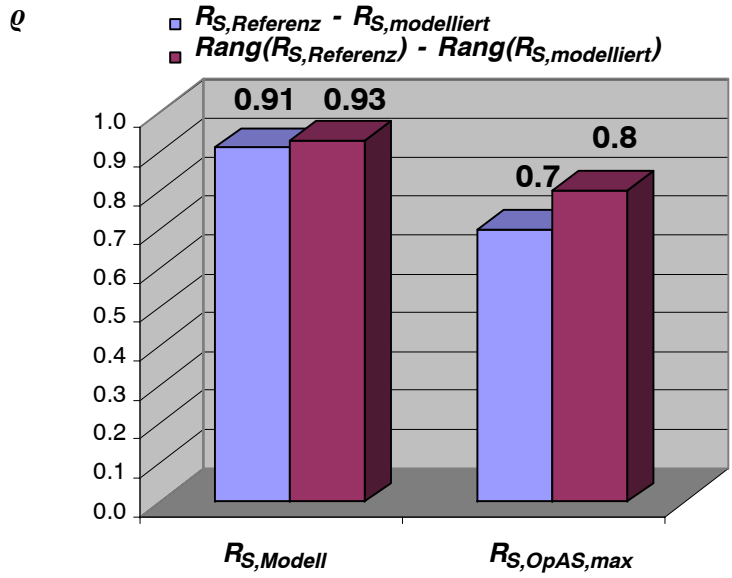


Bild 5. 2 . Korrelationskoeffizienten der Performance-Modellierungsergebnisse ($R_{S,Modell}$ und $R_{S,OpAS,max}$) mit den veröffentlichten Referenzdaten R_S ***(Anhang H)***.

Für die untersuchte Stichprobenmenge lassen sich aus den dargestellten Ergebnissen zwei Folgerungen ableiten. Für eine möglichst gute Performance-Abschätzung des erzielbaren Datendurchsatzes, müssen der Charakter der Verarbeitung und die Datenzugriffe berücksichtigt werden ($R_{S,Modell}$). Nur wenn es um den Vergleich von alternativen VLSI-Implementierungen geht, besteht die Chance mit einem stark vereinfachten, an Operationsraten *(MOPS/ GOPS)* orientierten Modell ($R_{S,OpAS,max}$) brauchbare Ergebnisse zu erzielen. An Hand des zugehörigen Korrelationskoeffizienten (ϱ = 0.8) läßt sich ableiten, daß hier die Abbildung auf eine Rangfolgeliste den Vergleich alternativer Lösungen am besten unterstützt.

5 .2 Modellierung der Schaltkreisfläche

In diesem Abschnitt sollen das in Kapitel 4 .4 .4 entwickelte Flächenmodell und seine alternativen Ansätze ***(Anhang E)*** diskutiert werden. Die einzelnen Modellierungsergebnisse für die untersuchte Stichprobe mit 27 Beispielrealisierungen werden in *Anhang G* tabellarisch dargestellt. Bild 5. 3 zeigt für die modellierten Schaltkreisgrößen den mittleren Fehler nach Gleichung (5.2). Der geringste Fehler ergibt sich für das *Bestcase-Flächenmodell (BCF)* mit auf äquivalente Technologien umgerechneten Parametern (Bild 4. 17 , Bild 4. 18 , Bild 4. 22 und *Anhang G*). Das *BCF*-Modell wird gefolgt vom *Bestcase-Minimum-Flächenmodell (BCMinF, ,* Bild 4. 24 *),* das die Streuung zu geringen (minimalen) Transistordichten über *Fuzzy-Intervalle* mit berücksichtigt. Fast gleich gut ist das *Mittelwert-Flächenmodell (MF)*. Deutlich schlechter wird das Modellierungsergebnis für das *Mittelwert-Intervall-Flächenmodell* (*MIF*, Bild 4. 26).

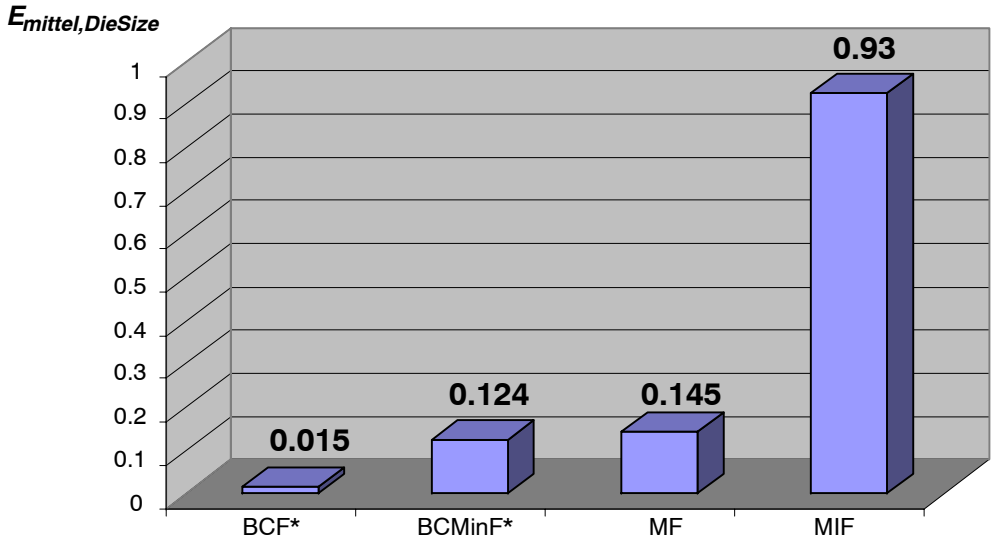


Bild 5. 3 . Mittlerer Fehler der modellierten Schaltkreisgröße *(DieSize, Anhang I)*.
* Anwendung äquivalenter Technologieparameter nach *Anhang G*.

Bild 5. 4 zeigt einen Vergleich der untersuchten Modelle über die Korrelationskoeffizienten zwischen originalen und modellierten Schaltkreisgrößen, bzw. zwischen der Rangfolge der

originalen Referenzdaten und der Rangfolge modellierter Werte. Die größte Korrelation mit den Originaldaten ergibt sich für das *BCF*-Modell bei Anwendung äquivalenter Technologieparameter. Die anderen Modelle liegen in der Korrelation mit den Referenzwerten relativ nahe beieinander. Insbesondere das bei der direkten Fehlerbetrachtung in Bild 5. 3 deutlich schlechtere *MIF*-Modell erweist sich bei der Korrelation der Rangfolgewerte als kaum schlechter als *BCMinF** und *MF.*

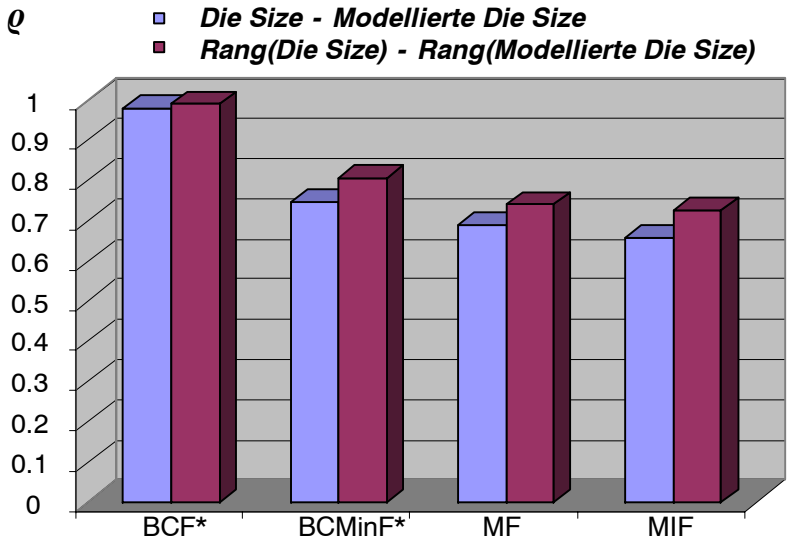


Bild 5. 4 . Korrelationskoeffizienten zwischen bekanntem Schaltkreisgrößen und modellierten Schaltkreisgrößen für verschiedene Flächenmodelle nach *Anhang E*
* Anwendung äquivalenter Technologieparameter nach *Anhang G*.

Die in den Bildern 5. 3 und 5. 4 dargestellten Ergebnisse können wie folgt interpretiert werden. Wenn die Schaltkreisfläche eines *ASICs* möglichst gut abgeschätzt werden soll, bietet sich das *BCF-Modell* an, das dann sehr genaue Ergebnisse berechnet, wenn spezifiziert ist, welcher Anteil der verfügbaren Metallisierungsebenen für Arithmetik/Logik und für Speicher jeweils genutzt wird. Sofern ein Entwurf durch einen Shrink der Gates von einer älteren Technologie in eine neue Technologie umgesetzt wird, sollte hier die Strukturgröße ***s*** aus dem ursprünglichen Prozeß angesetzt werden.

Im Hinblick auf einen geringen Modellfehler sind die Modelle *BcMinF** und *MF* annähernd gleich gut. Das *MF-Modell* ist etwas einfacher aus realen Daten abzuleiten und auf neue Modellierungen anzuwenden.

Der Einsatz des *MIF-Modelles* bringt keine erkennbaren Vorteile und läßt sich nur dann über die kaum schlechtere Korrelation in der Rangfolgebestimmung begründen, wenn es um den Vergleich alternativer *VLSI*-Architekturen geht.

5 .3 Effizienzbetrachtung und Multikriterienanalyse

In den Kapiteln 5 .1 und 5 .2 sind Ansätze für die Modellierung der Performance einer Echtzeitanwendung (*Beispiel:* Datendurchsatz) und für die Kosten (*Beispiel:* Schaltkreisgröße) diskutiert worden. In diesem Abschnitt der Arbeit sollen Kosten und Performance gemeinsam im Hinblick auf die Effizienz einer *ASIC*-Implementierung untersucht werden. In Kapitel 5 .3 .1 soll zunächst die direkt berechnete Effizienz gemäß Gleichung (2.2) betrachtet werden. In Kapitel 5 .3 .2 wird der in Kapitel 3 .3 eingeführte verallgemeinerte Effizienzbegriff nach Gleichung (3.3) näher untersucht *(Fuzzy-Multikriterienanalyse)*.

5 .3 .1 Effizienz als Quotient aus Datendurchsatz und Schaltkreisgröße

In diesem Abschnitt der Arbeit soll nun zunächst an Hand der modellierten Datendurchsatzrate $R_{S,Modell}$ *(Anhang H)* und der nach dem ***BCF*-Ansatz*** modellierten Schaltkreisfläche (Anhang I) die Effizienz gemäß Gleichung (2.2) bestimmt werden und mit der Effizienz bekannter Modellierungsdaten verglichen werden. Bild 5. 5 zeigt das Ergebnis mit einer für die betrachtete Stichprobenmenge recht großen Übereinstimmung zwischen Referenzdaten und Modellierungsergebnissen, die durch einen nahe bei 1 liegenden Korrelationskoeffizienten **ϱ = 0.93** formal bestätigt wird.

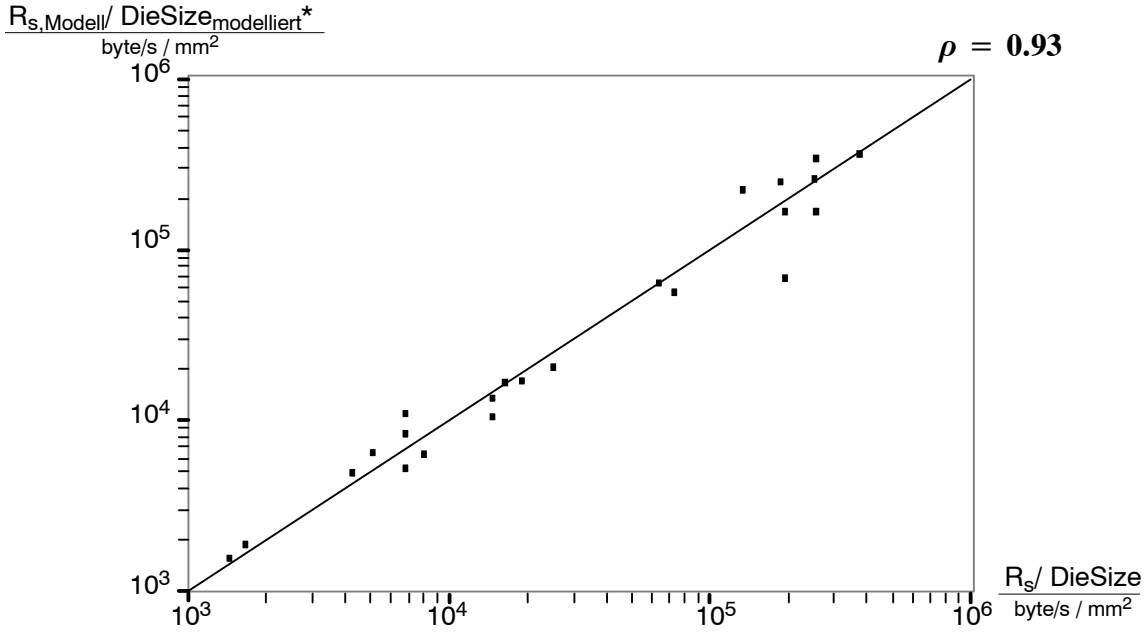


Bild 5. 5 . Modellierte Effizienz im Vergleich zu bekannten Daten ***(nach Anhang J)***.
* Flächen in mit *BCF-Modell* berechnet mit an äquivalente *BCF*-Technologie angepaßten Parametern *(Anhänge E,G).*

5 .3 .2 Fuzzy-Multikriterienanalyse und Zielerfüllung

Die Anwendung des verallgemeinerten Effizienzbegriffes als Erfüllungsgrad μ_f der *Fuzzy Multikriterienanalyse* nach Gleichung (3.3) setzt die Spezifikation sinnvoller Ziele (Bild 3. 8) voraus. Im Hinblick auf einen Vergleich mit der in Kapitel 5 .3 .1 diskutierten Effizienz wird das Ziel für die Schaltkreisgröße so definiert, daß diese so klein wie möglich sein soll. Da bei sehr kleinen Schaltkreisen ein über die Schaltkreisgröße definiertes Kostenmodell den relativen Anteil für das Gehäuse zu sehr vernachlässigt, soll eine untere Grenze von 20 mm^2 definiert werden. Die in dieser Arbeit analysierten *ASICs* haben eine maximale Größe von 240 mm^2. Für größere Schaltkreise werden geringe Ausbeuten nach Gleichung (4.17) zu hohen Herstellungskosten führen. Es wird daher beispielhaft für die folgende Untersuchung eine Zielsetzung definiert, die eine Mindestgröße der Schaltkreisgröße und die Maximalgröße der untersuchten Schaltkreise erfaßt :

$$DieSize_{Ziel} = [20, 20, 0, 240]\ mm^2 \tag{5.4}$$

Im Folgenden soll das Ziel für die Performance einer Anwendung der Videosignalverarbeitung so definiert werden, daß bis zu TV-Anwendungen (*CCIR* Format, Tabelle 4.1) der erzielbare Datendurchsatz so groß wie möglich sein soll. Eine darüber hinausgehende Performance soll dann als gleichwertig zum CCIR-Format bewertet werden :

$$R_{S,Ziel} = [21, 200, 21, 0]\ Mbyte/s \tag{5.5}$$

Bild 5. 6 stellt die in den Gleichungen (5.4) und (5.5) spezifizierten Ziele grafisch dar.

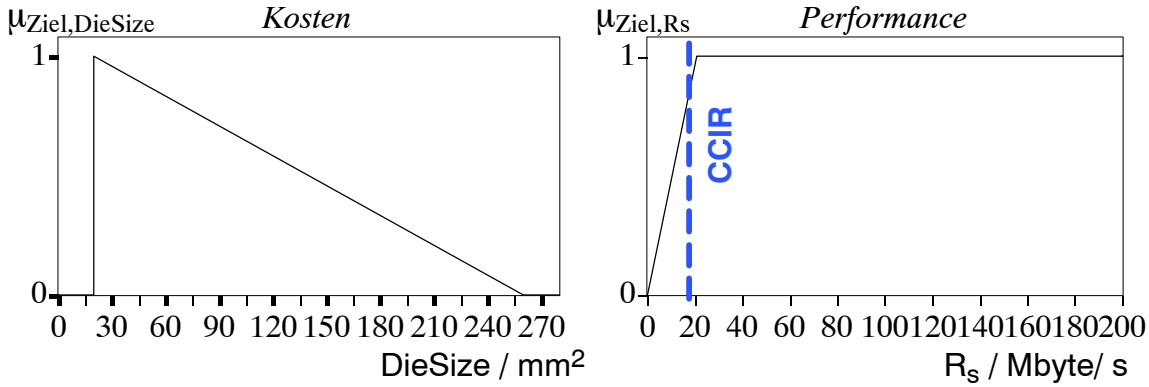


Bild 5. 6 . Beispiele für Zieldefinitionen der Multikriterienanalyse nach Gleichung (3.3).

Bild 5. 7 vergleicht die direkt modellierte Effizienz mit dem modellierten Erfüllungsgrad μ_f zu den in Bild 5. 6 spezifizierten Zielen. Der Korrelationskoeffizient $\varrho = 0.97$ deutet darauf hin, daß bei den hier spezifizierten Zielen ein Zusammenhang zwischen der Fuzzy-Multikriterienanalyse und der direkten Effizienzbestimmung besteht.

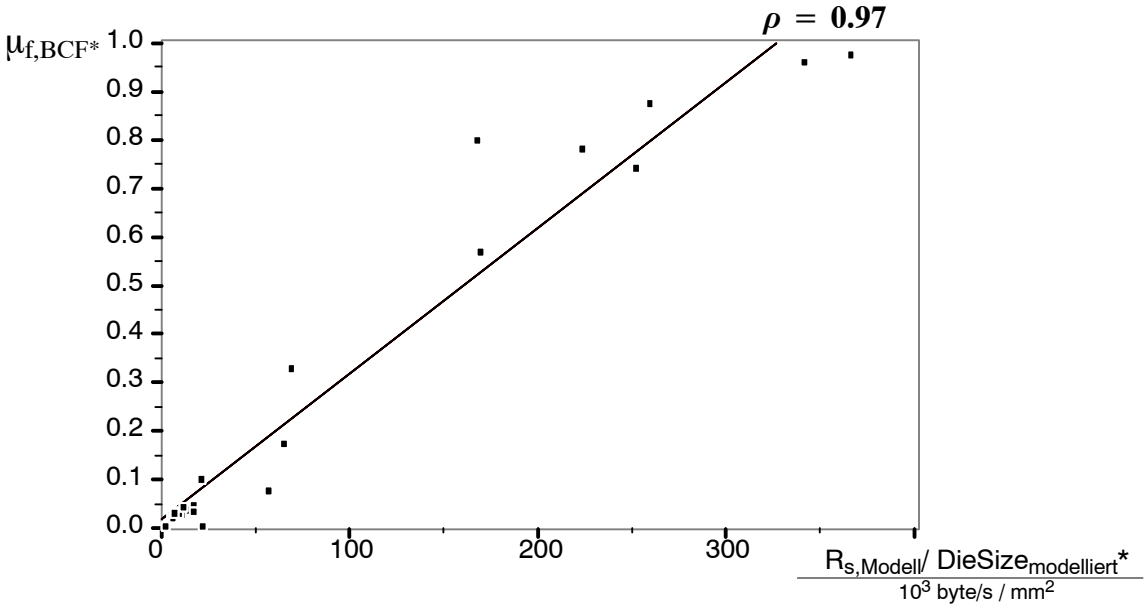


Bild 5. 7 . Vergleich der modellierten Effizienz und dem modellierten Erfüllungsgrad $\mu_{f,BCF}$ ***(nach Anhang J), $w_{Rs}=1$, $w_{DieSize}=1$,*** * Anwendung äquivalenter Technologieparameter nach *Anhang G*.

Das Ergebnis in Bild 5. 7 bestätigt experimentell die an Hand einer analytischen Modellierung entwickelte Hypothese, daß eine Fuzzy Multikriterienanalyse als Verallgemeinerung des Effizienzbegriffes betrachtet werden kann [39]. Die Interpretation der Fuzzy-Multikriterienanalyse als verallgemeinerte Effizienz wird auch über ihre vielfältigen Erweiterungsmöglichkeiten zusätzlich unterstützt. So können hier sogar linguistische Variablen zur Charakterisierung von Architekturansätzen eingesetzt werden [22].

Ein weiterer Ansatz zur Bewertung der Multikriterienanalyse ist der Grad der Übereinstimmung von Modellierungsergebnissen mit dem Erfüllungsgrad für die veröffentlichten Referenzdaten. Bild 5. 8 zeigt den mittleren Fehler des Erfüllungsrades μ_f nach Gleichung (5.2) für verschiedene eingesetzte Flächenmodelle. Grundlage sind die aus bekannten und modellierten Performance- und Kostenwerten *(Anhänge H, I)* gewonnenen Erfüllungsgrade für die in Bild 5. 6 spezifizierten Ziele. Der geringste Fehler entsteht für eine Multikriterienanalyse, bei der die Schaltkreisgröße mit dem *BCF**-Modell modelliert wird. Wenn es nicht möglich ist, das auf eine äquivalente Technologie umgerechnete *Bestcase*-Flächenmodell *(BCF)* anzuwenden, bietet sich das im mittleren Fehler nachfolgende *Mittelwert*-Flächenmodell *(MF)* zur Modellierung der Schaltkreisgröße an.

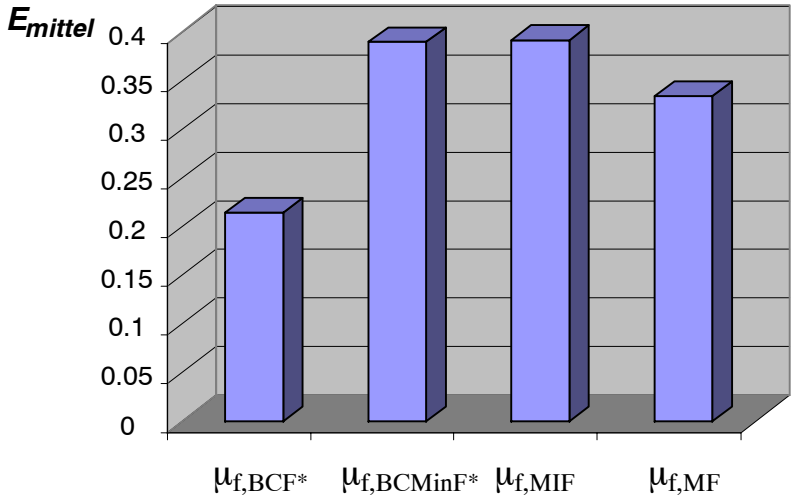


Bild 5. 8 . Mittlerer Fehler des Erfüllungsgrades für verschiedene Flächenmodelle
$w_{Rs}=1$, $w_{DieSize}=1$
* Anwendung äquivalenter Technologieparameter nach *Anhang G*.

Betrachtet man an Stelle des mittleren Fehlers die Korrelation zwischen den bekannten und den modellierten Werten für den Erfüllungsgrad, zeigt sich erneut, daß der Einsatz des *BCF**-Modells mit äquivalenten Technologieparametern zum besten Ergebnis führt. Geht es um die Korrelation zwischen Referenz- und Modellwerten folgt dann eine Multikriterienanalyse, die auf dem *MF*-Modell basiert. Wenn eine möglichst gute Erfassung der Rangfolge gewünscht wird, sollte die Schaltkreisfläche mit dem *MIF*-Modell bestimmt werden, sofern das *BCF**-Modell nicht anwendbar ist.

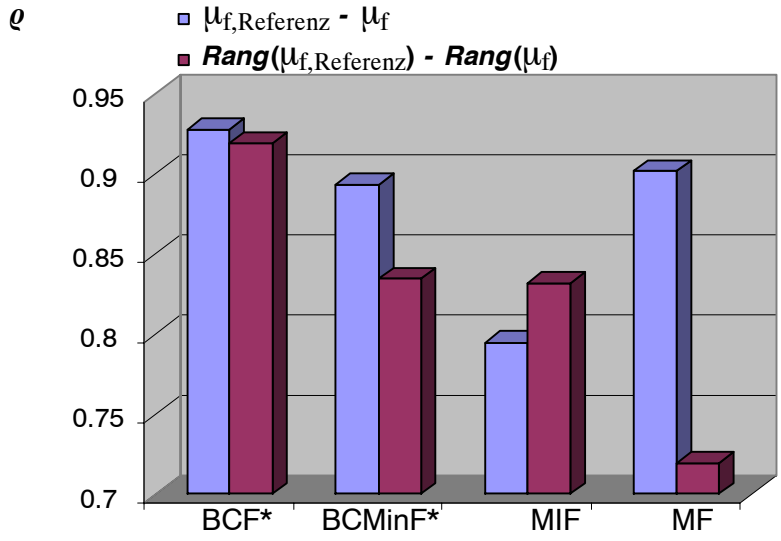


Bild 5. 9 . Korrelation zwischen bekanntem Erfüllungsgrad und modelliertem Erfüllungsgrad für verschiedene Modelle für die Schaltkreisfläche nach *Anhang E*, $w_{Rs}=1$, $w_{DieSize}=1$
* Anwendung äquivalenter Technologieparameter nach *Anhang G*.

Die in den Bildern 5. 7 - 5. 9 dargestellten Ergebnisse zeigen für die untersuchte Stichprobenmenge, daß bei einer geeigneten Zielsetzung (Bild 5. 6) vergleichbare Ergebnisse zwischen

einer direkt berechneten Effizienz nach Gleichung (2.2) und einer Multikriterienanalyse nach Gleichung (3.3) ermittelt werden. Der besondere Vorteil der Multikriterienanalyse liegt nun darin, daß neben der Einbeziehung bekannter Lösungen zur Effizienz, vielfältige neue Anforderungen sowohl für einzelne Kriterien, als auch in der Zahl der betrachteten Kriterien berücksichtigt werden können.

5 .4 Optimierung der Zielerfüllung über Zahl der Datenpfade und Speichergrößen

Der in Kapitel 4 entwickelte Ansatz zur Kosten- und Performance-Modellierung von VLSI-Architekturen ist in den Kapiteln 5 .1 bis 5 .3 für eine ausgewählte Stichprobenmenge für die Bewertung von VLSI-Implementierungen angewendet und untersucht worden. Hier hat sich der unter Beachtung des Charakters der Verarbeitung und der Datenzugriffe der modellierte Datendurchsatz $R_{S,Modell}$ als geeignetes Performance-Kriterium erwiesen. Für die Modellierung der Schaltkreisfläche sollte möglichst das entwickelte *Bestcase**-Flächenmodell ***(BCF*)*** angewendet werden. Hier wird jeweils für Speicher und für Arithmetik/ Logik berücksichtigt, welcher Anteil der verfügbaren Metallisierungsebenen für ein kompaktes Layout nutzbar ist und in welcher Strukturgröße das Modell am besten mit realen Daten übereinstimmt (Umrechnung auf eine äquivalente Technologie, *Anhang G)*. Beide Modellansätze haben sich auch für eine direkt aus dem Quotienten aus Performance und Kosten berechnete Effizienz und für eine erweiterte Effizienz (*Fuzzy-Multikriterienanalyse*) als am besten geeignet erwiesen und sollen im Folgenden verwendet werden, sofern keine abweichenden Kriterien angegeben werden.

Neben der bisher diskutierten Modellierung und Bewertung diskreter Lösungen liegt es für die Entwicklung neuer Konzepte nahe, Architekturparameter unter Berücksichtigung der applikationsspezifischen Anforderungen und der jeweiligen technologischen Randbedingungen zu optimieren. Für das in dieser Arbeit für die Modellierung eingesetzte *VSP Decision Program (Anhang A)* ist in einer Studienarbeit [23] ein Optimierungsansatz entwickelt worden, der auf genetischen Algorithmen basiert und der die Optimierung nichtlinearer Kostenfunktionen unterstützt.

Nachdem in den Kapiteln 5 .1 bis 5 .3 an Hand realisierter Beispiellösungen gezeigt werden konnte, daß der Einsatz der F*uzzy Multikriterienanalyse* zu sinnvollen Ergebnissen führt, sollen auch Optimierungsansätze auf Basis der veröffentlichten Lösungen untersucht werden. Von Bedeutung ist hier, ob eine Optimierung des Erfüllungsgrades μ_f für die in Bild 5. 6 dargestellten Ziele überhaupt möglich und auch sinnvoll ist. Als Beispiel, das aus veröffentlichten Daten ableitbar ist, soll jeweils die Zahl der Datenpfade (1...32) und die Größe der lokalen Speicher (512 byte...32 kbyte) für Videocodierungsanwendungen nach Kapitel 4 .3 .1 und *Anhang C* optimiert werden.

Bild 5. 10 zeigt die Ergebnisse für Optimierungsläufe der in [5], [21], [27], [29], [31], [36], [44], [47], [60], [61], [67], [84], [88] veröffentlichten Prozessoren. Für a) wird dargestellt, wie sich im Mittel jeweils für veröffentlichte Decoder, Encoder und Codecs der Erfüllungsrad μ_f durch eine modellgestützte Optimierung erhöhen läßt. Links wird jeweils die Verbesserung des Erfüllungsgrades dargestellt, der mit ***BCF**** und $R_{S,Modell}$ berechnet wird. Rechts wird jeweils der Erfüllungsgrad aus ***BCF**** und $R_{S,OpAS,max}$ (aus den Operationsraten *MOPS*/ *GOPS* berechneter maximal Datendurchsatz) gewonnen. Bild 5. 10 a zeigt, daß ein größeres Optimierungspotential für Encoder- und Codec-Anwendungen besteht. Wird die Performance über $R_{S,OpAS,max}$ bestimmt, bietet das angewendete Modell nur ein geringes Optimierungspotential für den Erfüllungsgrad μ_f. Bild 5. 10 b zeigt, wie sich der Erfüllungsgrad über das Optimierungsverfahren erhöht. Hier wird ein Optimierungsgewinn definiert, der sich aus dem Verhältnis des Erfüllungsgrades nach der Optimierung zu dem Erfüllungsgrad vor Beginn der Optimierung (*für alle PE's :* Speichergröße = [512,32768,0,0], Datenpfadzahl = [1,32,0,0]). Bild 5. 10 b könnte einer frühen Konzeptphase entsprechen, in der zunächst mit einer größeren Lösungsmenge begonnen wird und dann ein Lösungsraum mittels Optimierung eingeschränkt wird. Der Optimierungsgewinn ist ein Indikator dafür, ob das gewählte Modell bei der Konzeptgewinnung sinnvoll anwendbar ist. Ist der Optimierungsgewinn sehr klein, würde das einen Hinweis darauf geben, daß das bisher entwickelte Modell für die Konzeption neuer, applikationsspezifischer Architekturen ungeeignet ist. Hier deutet ein Optimierungsgewinn zwischen 4 und 12 darauf hin, daß Verbesserungen in frühen Konzeptphasen ableitbar sind.

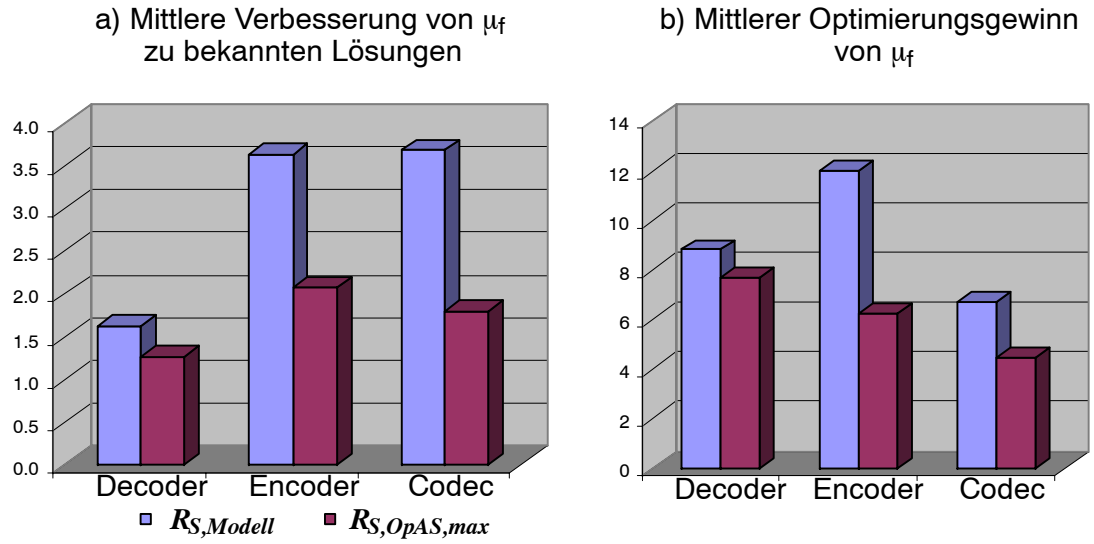


Bild 5. 10 . Mittlere Verbesserung und Optimierungsgewinn zu μ_f, jeweils gemittelt über Lösungen zu Decodern, Encodern und Codecs.

Für die in Bild 5. 10 a) untersuchte Stichprobenmenge zeigt sich, daß das entwickelte Modell ein Potential zur Verbesserung des Erfüllungsgrades μ_f für bereits realisierter Lösungen bietet. Hier gibt es Unterschiede, wenn der Erfüllungsgrad μ_f aus dem Datendurchsatz mit $R_{S,Mo}$-

$R_{S,Modell}$ (Datendurchsatz unter Einbeziehung des Charakters der Verarbeitung und der Datenzugriffe) oder mit $R_{S,OpAS,max}$ (Operationsratenmodell in *MOPS*/ *GOPS*) bestimmt wird.

Mit Bild 5. 11 in dem die Erhöhung der Datenpfadzahlen für beide Ansätze aufgetragen ist, sollen hier die Unterschiede zwischen beiden Lösungen etwas genauer analysiert werden. Nur in einem Fall wird bei Einsatz des $R_{S,OpAS,max}$-Modells die Zahl der Datenpfade weiter erhöht als bei $R_{S,Modell}$. In allen anderen Fällen ergeben sich geringere oder keine Erhöhungen der Zahl der Datenpfade für das $R_{S,OpAS,max}$-Modell.

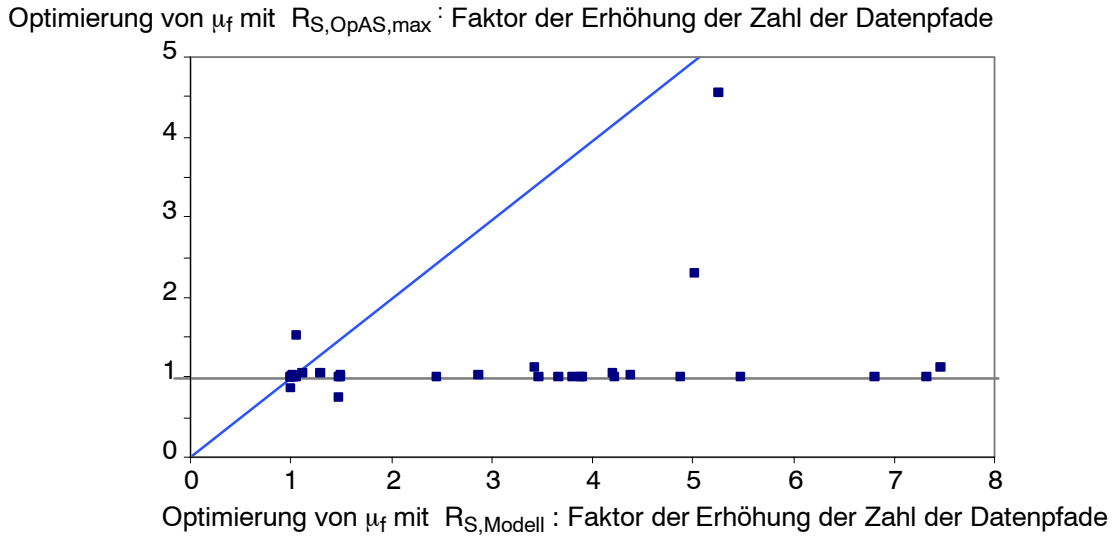


Bild 5. 11 . Erhöhung der Zahl der Datenpfade durch Optimierung des Erfüllungsgrades μ_f für verschiedene Modelle zur Berechnung des Datendurchsatzes R_S.

Das in Bild 5. 11 dargestellte Ergebnis ist erklärbar, weil das $R_{S,OpAS,max}$-Modell auf den maximalen Operationsraten eines applikationsspezifischen Schaltkreises basiert, ohne den Charakter der Verarbeitung und ohne die Datenzugriffe zu berücksichtigen. In der Praxis zeigt das $R_{S,Modell}$, daß sich unter Einbeziehung des Charakters der Verarbeitung und der Datenzugriffe eine geringere Auslastung von Funktionseinheiten ergibt. Wendet man das $R_{S,OpAS,max}$ - Modell als hinreichendes Kriterium zur Performance-Modellierung an, verspricht es einen höheren Datendurchsatz, als er tatsächlich realisierbar ist. Als Folge zeigen die Optimierungsergebnisse, daß entweder schon ein einzelner Datenpfad ausreicht oder nur wenige Datenpfade zu einer optimalen Lösung führen.

Neben einer Erhöhung der Zahl der Datenpfade zeigt sich, daß für die untersuchte Stichprobenmenge auch eine Reduktion der Speichergrößen zu einer besseren Zielerfüllung beitragen kann. Bild 5. 12 zeigt die aus Optimierungen resultierenden Speicherreduktionen im Vergleich für μ_f basierend auf $R_{S,Modell}$ und μ_f basierend auf $R_{S,OpAS,max}$. Tendenziell werden bei Verwendung von $R_{S,OpAS,max}$ Speichergrößen stärker reduziert. Bei näherer Betrachtung der Ergebnisse kann man hier feststellen, daß sich die Speichergrößen an der unteren für die Optimierung vorgegebenen Grenze orientieren. Das ist darüber erklärbar, daß bei Einsatz von

$R_{S,OpAS,max}$ der Datendurchsatz unabhängig von Datenzugriffsanforderungen berechnet wird. Kleinere Speichergrößen tragen daher direkt zu einer Erhöhung der Effizienz bei.

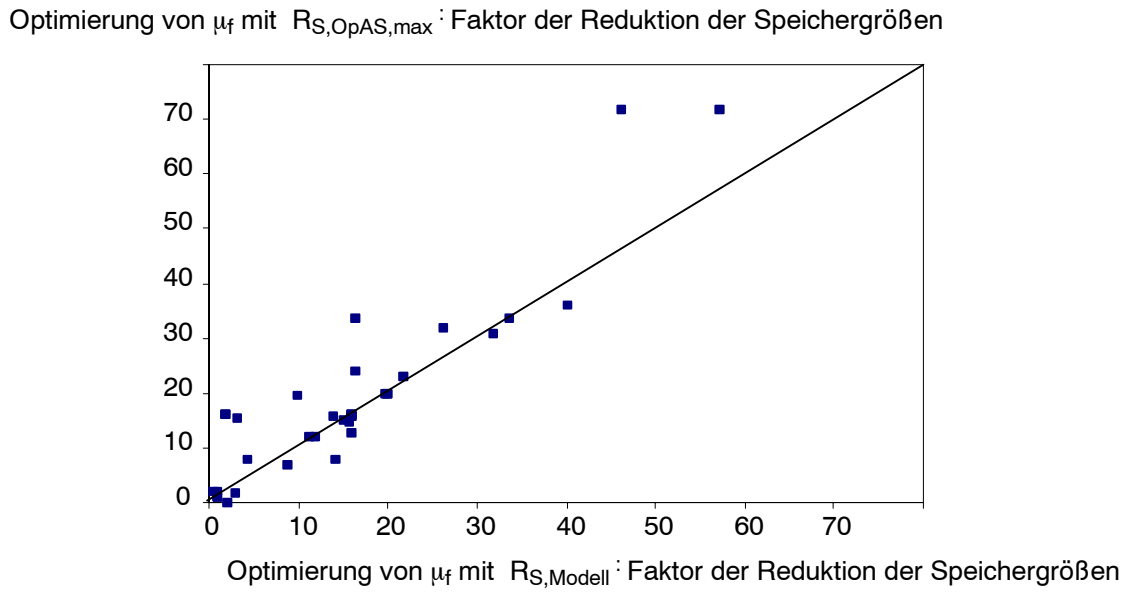


Bild 5. 12 . Reduktion der lokalen Speichergrößen durch Optimierung des Erfüllungsgrades μ_f für verschiedene Modelle zur Berechnung des Datendurchsatzes R_S.

Es kommt generell bei beiden Ansätzen durch die Optimierung zu weitgehenden Speicherreduktionen. Das hat zwei mögliche Gründe. Einerseits kann der in dieser Arbeit vorausgesetzte Zusammenhang zwischen Speichergrößen und Datenzugriffsanforderungen eine stärkere Reduktion von externen Datenzugriffen modellieren, als es in der Realität, z. B. auf Grund von Einschränkungen im Kontrollfluß eines Prozessors, möglich ist. Andererseits erfaßt das in dieser Arbeit angesetzte Modell nicht Effekte, die sich dadurch ergeben, daß für kleine Videoformate komplette Bildspeicher oder generell zumindest Blockzeilenspeicher monolithisch integrieren lassen. Hier ergeben sich für ein auf einer Platine realisiertes Gesamtsystem mögliche Einsparungen, die eine Systemeffizienz erhöhen, obwohl nach dem hier untersuchten Modell die Effizienz eines betrachteten *ASICs* durch die größeren Speicher sinkt. Um derartige Effekte besser zu erfassen, müßte das Kosten- und Performance-Modell einer Anwendung von den Grenzen eines *ASICs* auf das umgebende Board oder sogar das Gesamtsystem erweitert werden.

5 .5 Anwendung von *Fuzzy-Arithmetik* in der Konzeptphase

Zunächst wurden in dieser Arbeit Methoden der *Fuzzy Set Theorie* eingesetzt, um Transistordichten konsistent zu modellieren (*BCMinF*- und *MIF*-Flächenmodell nach Kapitel 4 .4 und *Anhang E*). Als nächster Schritt folgte die Spezifikation von Implementierungszielen für Kosten und Performance mit *Fuzzy-Intervallen* in Trapezform.

Nach Kapitel 3 liegt ein wichtiger Grund für die Anwendung der *Fuzzy Set Theorie* in der Erfassung möglicher Lösungsmengen. Insbesondere bei der Konzeption einer neuen, applika-

tionsspezifischen *VLSI*-Architektur bietet es sich hier an, die Unschärfe von Modellparametern mittels *Fuzzy-Zahlen* zu erfassen und Kosten- und Performance-Modelle um eine korrespondierende *Fuzzy-Arithmetik* zu erweitern.

Im Folgenden soll an Hand eines Beispieles, das aus den vorliegenden Daten abzuleiten ist, untersucht werden, ob eine stark vereinfachte Modellierung, wie sie in frühen Konzeptphasen auf Grund fehlender Detailinformationen häufig erforderlich ist, überhaupt zu sinnvollen Ergebnissen führen kann.

5 .5 .1 Vereinfachte Modellierung der Transistorzahlen von Verarbeitungseinheiten

Während der Konzeptphase der Entwicklung eines neuen Prozessors lassen sich Flächen von Speichermodulen recht einfach aus den Transistorzahlen nach Gleichung (4.5) ermitteln. Schwieriger wird es auf Grund zahlreicher Implementierungsalternativen, Transistorzahlen für *Arithmetische* und *Logische Einheiten* und für *Kontrolleinheiten* in frühen Konzeptphasen zu ermitteln. Ein Lösungsansatz ist die Verwendung von parametrisierten Architekturbeschreibungen nach Kapitel 4 .3 .2 (Bild 4. 7). Im Einzelfall können auch diese Modelle noch eine zu große Komplexität aufweisen und so kann eine weitergehende Vereinfachung gewünscht sein. Eine wichtige zu untersuchende Frage ist, wie weit eine vereinfachte Modellierung der Transistorzahlen der Verarbeitungseinheiten sinnvoll ist.

Hier soll aus den veröffentlichten Referenzdaten eine *Fuzzy-Modellierung* der Transistorzahlen gewonnen werden. Grundidee ist eine in mehreren Stufen angewendete starke Vereinfachung, bei der jeweils allen Datenpfaden, skalaren Verarbeitungseinheiten oder sonstigen arithmetisch logischen Einheiten jeweils unabhängig von ihrer spezifischen Architektur genau eine als *Fuzzy-Zahl* beschriebene Zahl der Transistoren zugeordnet wird. Einerseits wird es auf Grund der kleinen Stichprobenmenge an Referenzdaten nicht möglich sein, einen vorgeschlagenen Ansatz mit einer statistischen Basis zu verifizieren. Wenn andererseits ein Modell auch bei starken Vereinfachungen sinnvolle und plausible Ergebnisse erzielt werden, deutet das auf die Anwendbarkeit der *Fuzzy-Arithmetik* hin.

In *Anhang K* werden für die untersuchten Referenzlösungen angegebene oder aus den Layouts über Flächenverhältnisse herausgerechnete Transistorzahlen für Module tabellarisch aufgelistet. Als starke Vereinfachung sollen alle erfaßten Realisierungsdaten jeweils auf eine Transistorzahl für Datenpfade, eine für skalare Einheiten und eine für sonstige Einheiten abgebildet werden. Es gibt keine feste Regel, wie reale Daten auf *Fuzzy-Zahlen* abzubilden sind. Das erschwert einerseits die Modellbildung, bietet aber andererseits auch die Chance, Sichtweisen eines Entwicklers in ein Modell aufzunehmen. Für den hier untersuchten Ansatz soll die hinter der Modellierung stehende Sichtweise in die Richtung gehen, daß die Realisierung des Mittelwertes der jeweils erfaßten Transistorzahlen sicher möglich sein soll ($\mu_{NTrans} = 1$). Für Werte, die außerhalb der minimalen und maximalen Beispielwerte liegen, soll angenommen werden, daß sie mit Sicherheit keine möglichen Realisierungswerte für ein untersuchtes Architektur-

konzept darstellen ($\mu_{NTrans} = 0$). Bild 5. 13 zeigt, daß die Abbildung der mittleren, der minimalen und der maximalen Werte auf *Fuzzy-Zahlen* in Dreiecksform führt.

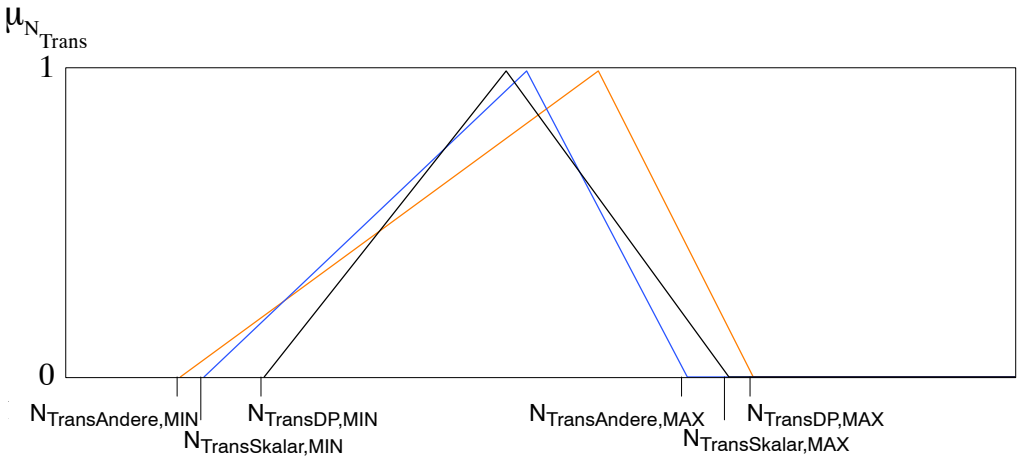


Bild 5. 13 . Mögliche Lösungsmenge für die Transistorzahl (N_{Trans}) einer Verarbeitungseinheit als *Fuzzy-Zahl*, nach *Anhang K*.

Zum Vergleich mit Methoden der klassischen Intervallarithmetik sollen die Daten aus *Anhang K* jeweils über ihre minimalen und ihre maximalen Werte auf Intervalle in Rechteckform abgebildet werden. Bild 5. 14 zeigt die resultierende Intervalldarstellung.

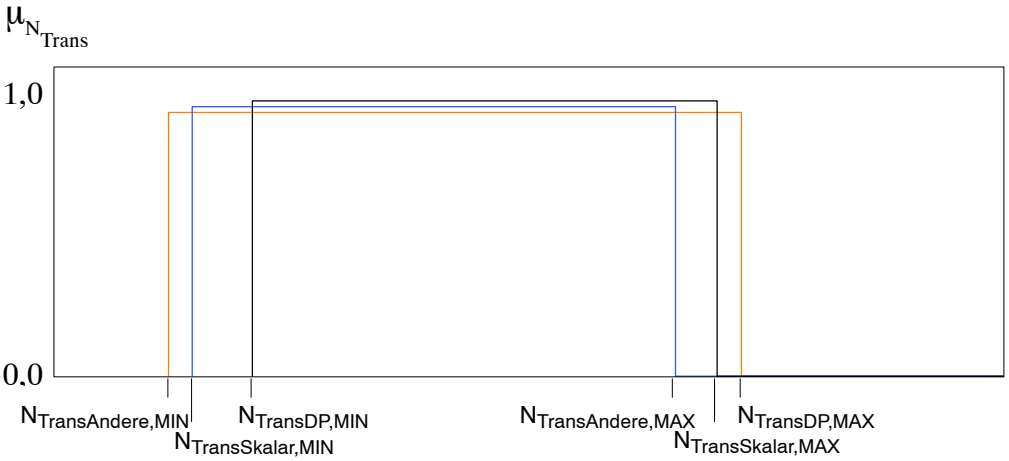


Bild 5. 14 . Mögliche Lösungsmenge für die Transistorzahl (N_{Trans}) einer Verarbeitungseinheit als *Intervall*, nach *Anhang K*.

Das aus den Daten in *Anhang K* abgeleitete vereinfachte Modell wird in Tabelle 5.1 dargestellt. Im Hinblick auf eine Analyse des Einflusses von *Fuzzy-Zahlen* auf die Qualität eines Modellierungsergebnisses sollen die modellierten Referenzlösungen schrittweise in mehreren Varianten (V1-V3) angewendet werden. Bei V1 wird allen Datenpfaden eines *ASICs* eine Transistorzahl [246000,246000,204000,1000000] zugewiesen. Skalare Einheiten und andere Einheiten werden mit ihren jeweils bekannten Transistorzahlen modelliert. Bei den Varianten V2 und V3 kommen die Fuzzy-Zahlenwerte aus Tabelle 5.1 für skalare und andere Einheiten

jeweils mit dazu. Die Variante V3 soll dann zusätzlich noch mit der klassischen Intervallösung nach Bild 5. 14 angewendet werden.

		Zahl der Transistoren / 10^3		**Variante**		
		Fuzzy-Zahl	Intervall	V1	V2	V3
$N_{TransDP}$	(Datenpfad)	[246,246,204,1000]	[42,1246,0,0]	x	x	x
$N_{TransSkalar}$	(skalare Einheiten)	[286,286,258,635]	[28,921,0,0]	o	x	x
$N_{TransAndere}$	(andere Logik)	[483,483,460,1007]	[23,1490,0,0]	o	o	x

Tabelle 5.1 . Stark vereinfachte Modellierung der Transistoranzahlen von Schaltkreismodulen (Datenpfade, skalare Einheiten, andere Logik).
x: Ersetzen mit vereinfachtenWerten, o: Verwendung originaler Werte, aus *Anhang K*.

Die Gesamtzahl der Transistoren für arithmetische und logische Einheiten ergibt sich dann mit dem vereinfachten Modell als :

$$N_{Trans,Arithmetik} = N_{TransDP} \cdot Anzahl_{Datenpfade} + N_{TransSkalar} \cdot Anzahl_{Skalareinheiten} + N_{TransAndere} \cdot Anzahl_{AndereEinheiten} \tag{5.6}$$

5 .5 .2 Modellierung mit dem vereinfachten Modell

Bild 5. 15 zeigt für einen älteren Prozessor, der in einer 1.0 µm-Technologie realisiert wurde, die resultierende Flächenabschätzungen für die einzelnen Varianten. Die Originalgröße des Schaltkreises beträgt 214 mm^2. Die Modellierung mit bekannten Transistorzahlen mit dem *BCF**-Modell wird durch ein dreieckförmiges Intervall spezifiziert, das sich eng um den bekannten Referenzwert anordnet. Über $V1_{Fuzzy}$ und $V2_{Fuzzy}$,$V3_{Fuzzy}$ und V3 weiten sich die Intervalle zunehmend auf. Ihre maximalen Schaltkreisflächen sind auf Grund der resultierenden geringen Ausbeute für eine *VLSI*-Realisierung unrealistisch groß. Die Intervallbreiten begründen sich über die Spannweite der Strukturgrößen der analysierten Beispiele, die von 1 µm bis zu 0.1 µm reichen, und der resultierenden Transistorzahlen der Schaltkreismodule.

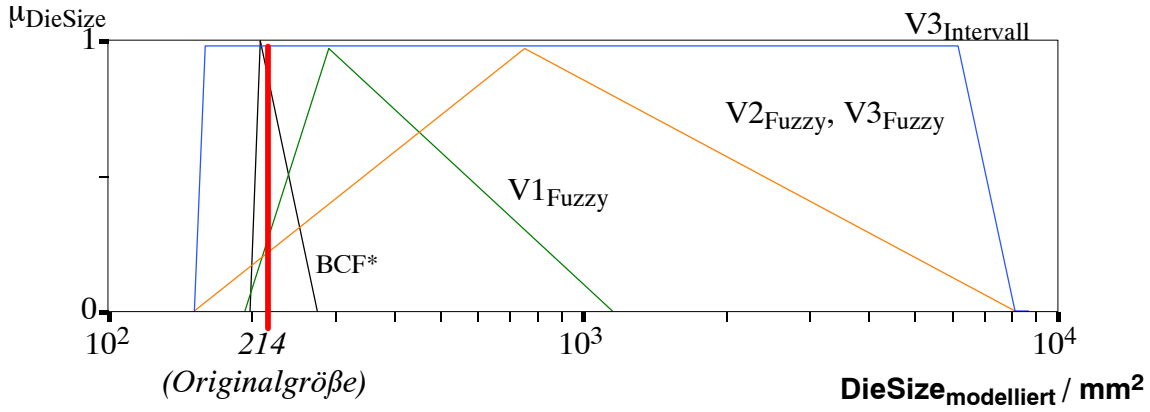


Bild 5. 15 . Beispiel für die modellierte Schaltkreisgröße (DieSize) [67] für verschiedene Varianten (V1-V3) zu Transistorzahlen pro Schaltkreismodul nach Tabelle 5.1 .
* Anwendung äquivalenter Technologieparameter nach *Anhang G*.

Mit dem entwickelten Ansatz zur Modellierung der Transistorzahl von Verarbeitungseinheiten sollen für die Multikriterienanalyse der mittlere Fehler des Erfüllungsgrades μ_f nach Gleichung (5.2) und die Korrelation mit bekannten Referenzwerten untersucht werden. Wenn trotz der sehr starken Vereinfachung, plausible Ergebnisse der Multikriterienanalyse heraus kommen, ist das ein Hinweis auf die Anwendbarkeit des vorgeschlagenen Ansatzes mit *Fuzzy-Zahlen*.

Bild 5. 16 zeigt, daß der mittlere Fehler des Erfüllungsgrades von der Modellierung mit bekannten Transistorzahlen zu den vereinfachten Lösungen nach Tabelle 5.1 etwa um den Faktor 3 zunimmt. Der mittlere Fehler für die unterschiedlichen Varianten liegt annähernd in der gleichen Größenordnung. Diese Ergebnis ist ein Hinweis darauf, daß die Multikriterienanalyse für die hier untersuchte Stichprobenmenge und die gesetzten Ziele (Bild 5. 6) recht unempfindlich gegenüber unscharfen Modellparametern für Transistorzahlen ist. Diese Tendenz wird bestätigt von dem gleichmäßig großen Korrelationskoeffizienten zwischen bekanntem Erfüllungsgrad und den modellierten Werten für alle untersuchten Varianten.

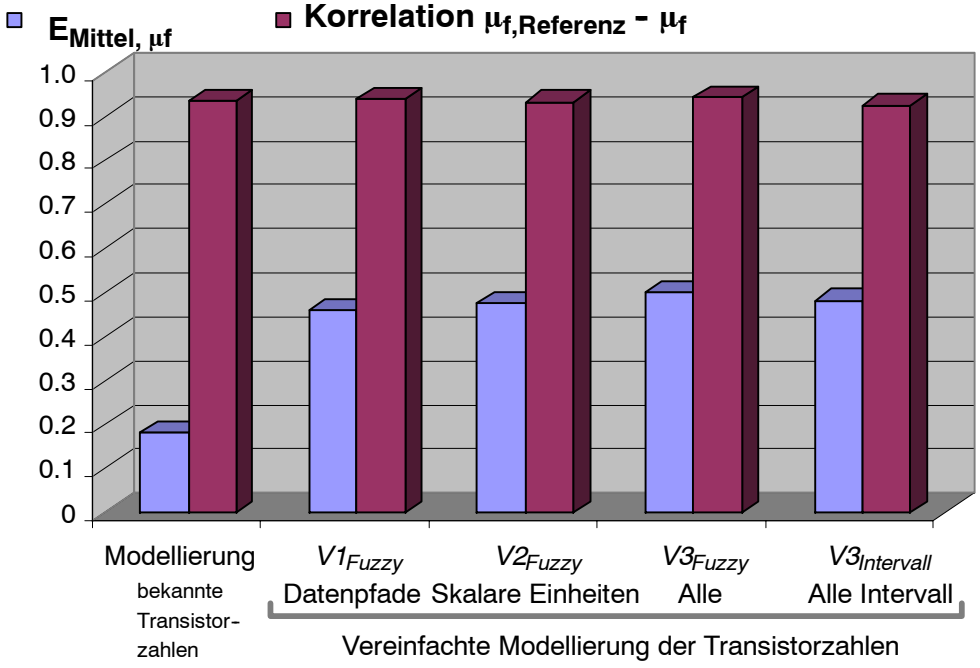


Bild 5. 16 . Mittlerer Fehler E_{Mittel} und Korrelation mit Referenzdaten für die untersuchten vereinfachten Modellierungen der Transistorzahlen von Verarbeitungseinheiten

Bild 5. 17 zeigt an Hand des mittleren Fehlers für die Effizienz (direkt berechneter Quotient aus Datendurchsatz und Schaltkreisgröße), daß mit jeder zusätzlichen Vereinfachung des Modelles (Aufweitung der zugehörigen Fuzzy-Intervalle für die modellierte Schaltkreisgröße) der mittlere Fehler zu den Referenzdaten zunimmt.

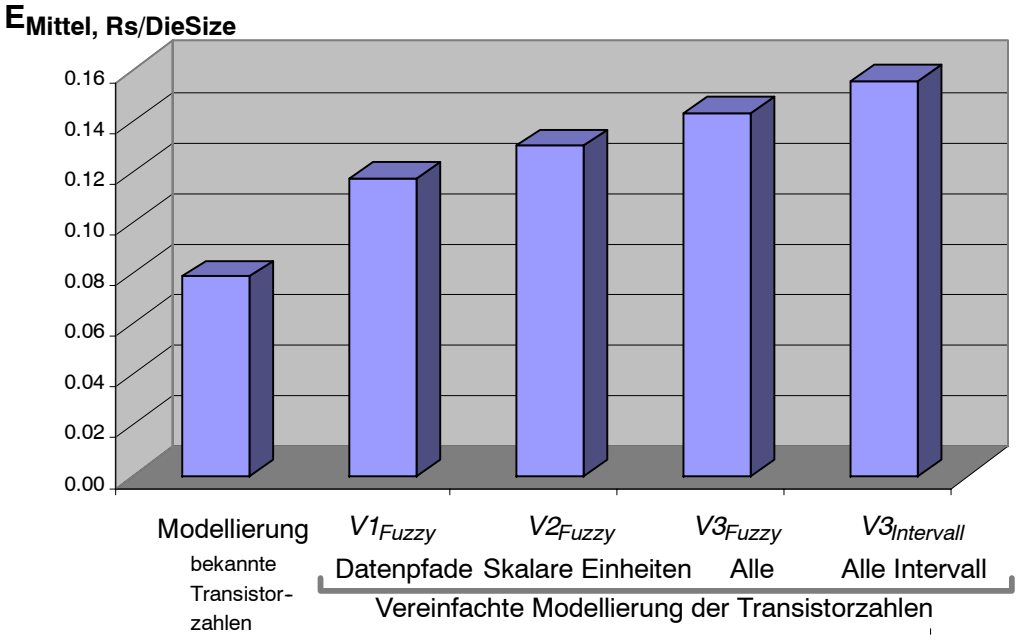


Bild 5. 17 . Mittlerer Fehler $E_{Mittel,Rs/DieSize}$ für die untersuchten vereinfachten Modellierungen der Transistorzahlen von Verarbeitungseinheiten. $R_S/DieSize$ ist vor der Fehlerberechnung defuzzyfiziert [100]worden.

Der in Bild 5. 16 dargestellte, ab der ersten Vereinfachung etwa konstante Fehler deutet zumindest für die untersuchte Stichprobenmenge und die spezifizierten Entwurfsziele darauf hin, daß eine ***Fuzzy-Multikriterienanalyse*** hier unempfindlicher gegenüber einer Erhöhung der Unschärfe der Modellparameter ist als die direkt berechnete Effizienz $R_S/DieSize$.

5 .6 Optimierung unter unscharfen Randbedingungen

Ein wichtiges Ergebnis aus Kapitel 5 .5 ist die Unempfindlichkeit der *Fuzzy-Multikriterienanalyse* gegenüber einer Erhöhung des Grades der Unschärfe über die in Tabelle 5.1 eingeführten Varianten. Sollte diese Unempfindlichkeit über die untersuchte Stichprobe hinausgehen und unabhängig von den hier untersuchten Implementierungszielen (Bild 5. 6) bleiben, müßte sich die *Fuzzy-Multikriterienanalyse* besonders gut für die Architekturoptimierung unter unscharfen Randbedingungen eignen.

Daher soll der modellierte Erfüllungsgrad μ_f gemäß der Variante *Fuzzy Zahl/ V3* aus Tabelle 5.1 spezifiziert und vergleichbar zur Untersuchung in Kapitel 5 .4 mit dem in [23] entwickelten, genetischen Ansatz optimiert werden. Bild 5. 18 zeigt im Vergleich den optimierten Erfüllungsgrad für bekannte Transistorzahlen und den optimierten Erfüllungsgrad für die Variante *Fuzzy Zahl V3* nach Tabelle 5.1 . Zwischen beiden Erfüllungsgraden gibt es einen engen Zusammenhang, der auch durch den großen Korrelationskoeffizienten $\varrho = 0.997$ rechnerisch bestätigt wird.

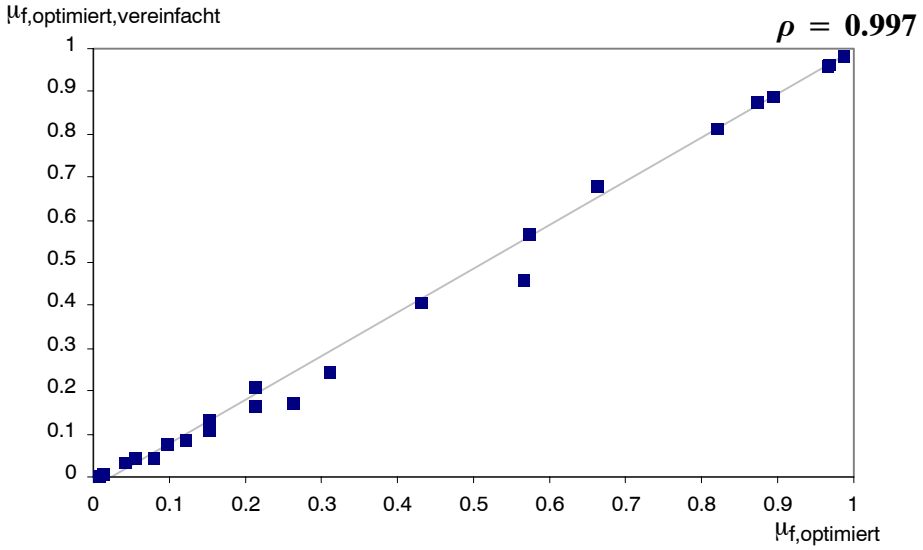


Bild 5. 18 . Vergleich der Optimierung der Effizienz μ_f für bekannte Transistorzahlen der Module ($\mu_{f,optimiert}$) und vereinfachend modellierten Transistorzahlen nach Tabelle 5.1 (*Fuzzy Zahl* V3:$\mu_{f,optimiert,vereinfacht}$).

In Bild 5. 19 werden vergleichbar zur Untersuchung in Kapitel 5 .4 die Optimierungsergebnisse mit bekannten Transistorzahlen und mit vereinfacht modellierten Transistorzahlen nach Tabelle 5.1 (Variante *Fuzzy Zahl V3*) verglichen. Hier werden erneut die mittlere Verbesserung zur bekannten Referenzlösung und der mittlere Optimierungsgewinn untersucht. Trotz der im Einzelfall großen Unschärfe der Daten ergeben sich für die vereinfachte Modellierung der Transistorzahlen jeweils nur geringe Reduktionen in der Verbesserung des Erfüllungsgrades oder des Optimierungsgewinns.

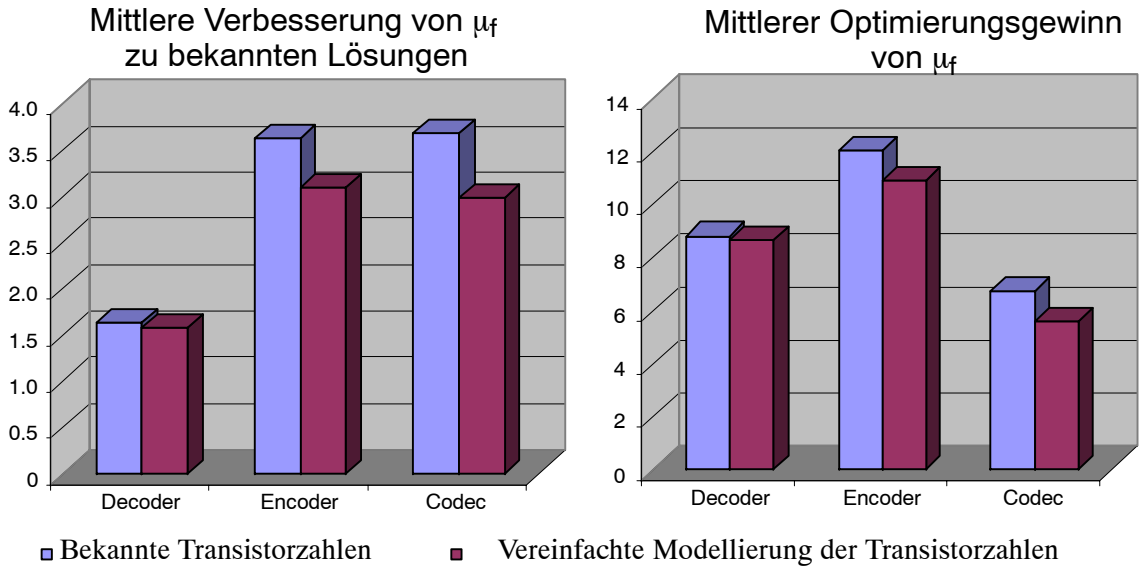


Bild 5. 19 . Mittlere Verbesserung und Optimierungsgewinn zu μ_f, jeweils gemittelt über Lösungen zu Decodern, Encodern und Codecs. Vereinfachte Modellierung: Fuzzy Zahl V3 (Tabelle 5.1).

Die Ergebnisse in Bild 5. 19 passen zum Korrelationskoeffizienten $\varrho = 0.997$ (Bild 5. 18) und deuten darauf hin, daß die Optimierungen mit bekannten Transistorzahlen der Verarbeitungseinheiten und die Optimierung mit unscharf beschriebenen Transistorzahlen der Verarbeitungseinheiten zu vergleichbar guten Ergebnissen führen können.

Nach der Fragestellung, ob bei unscharfen und bei bekannten Modellparametern vergleichbare optimierte Erfüllungsgrade erzielt werden können, stellt sich die Frage, wie sich die Unschärfe der betrachteten Transistorzahlen auf die optimalen Lösungen auswirkt. Bild 5. 20 zeigt hier den Zusammenhang für den Faktor der Reduktion der Speichergrößen. Der Korrelationskoeffizient ist $\varrho = 0.90$. Der Zusammenhang ist tendenziell linear, weist aber eine größere Streuung auf.

Bild 5. 21 vergleicht die aus der Optimierung gewonnene Zahl der Datenpfade für das Referenzmodell mit dem unscharfen Modell nach Tabelle 5.1 (Variante *Fuzzy Zahl V3*). Unter Einbeziehung aller Werte ergibt sich hier zwischen beiden Optimierungsergebnissen ein Korrelationskoeffizient $\varrho = 0.78$. Alternativ kann man von Hand die Ergebnisse in zwei Klassen aufteilen. Eine erste Klasse ordnet sich um eine Gerade mit der *Steigung 1/ 1* an. Eine zweite Klasse ergibt sich für eine *Steigung 1 / 2*. Die zugehörigen Korrelationskoeffizienten erhöhen sich für die Gerade mit der *Steigung 1/1* auf $\varrho_{1/1} = 0.99$ und bei der Geraden mit der *Steigung 1/2* auf $\varrho_{1/2} = 0.90$.

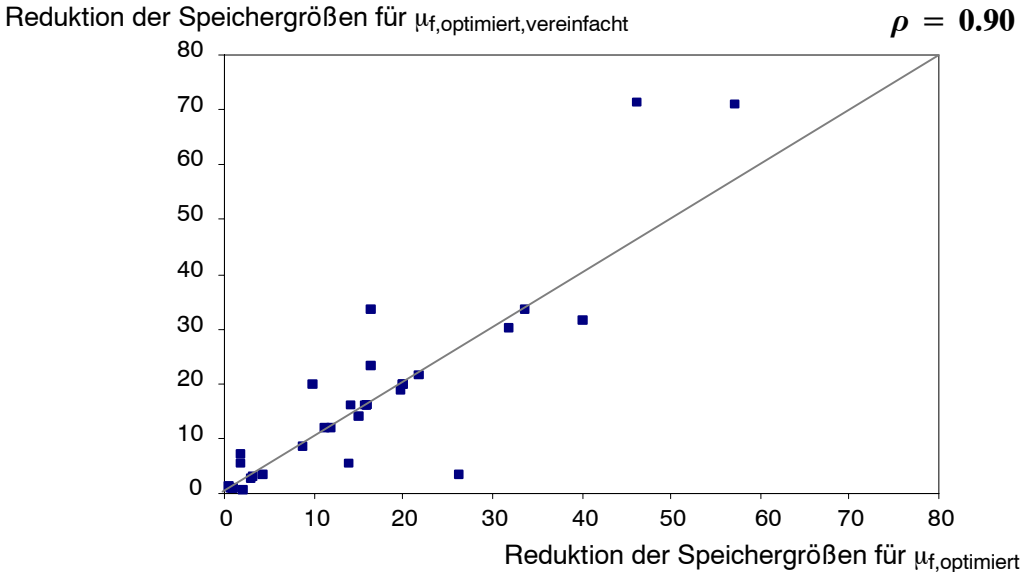


Bild 5. 20 . Reduktion der Speichergrößen gegenüber veröffentlichten Daten durch Optimierung: $\mu_{f,optimiert}$: für bekannte Transistorzahlen; $\mu_{f,optimiert,\ vereinfacht}$: für vereinfacht modellierte Transistorzahlen der Verarbeitungseinheiten nach Tabelle 5.1 (*Fuzzy Zahl* V3).

Bei der Suche nach möglichen Zusammenhängen mit dem Ergebnis in Bild 5. 21 kann man die in Kapitel 2 .3 .4 eingeführte Granularität nach Stone [86] näher betrachten. Die Granularität R/C einer Task ergibt sich aus dem Verhältnis zwischen Rechenzeit *(**Run Time:** $T_{Proc,PE}$)* und der Kommunikationszeit *(**Communication Time:** $T_{IO,PE}$)* bei Abbildung auf ein Prozessorelement.

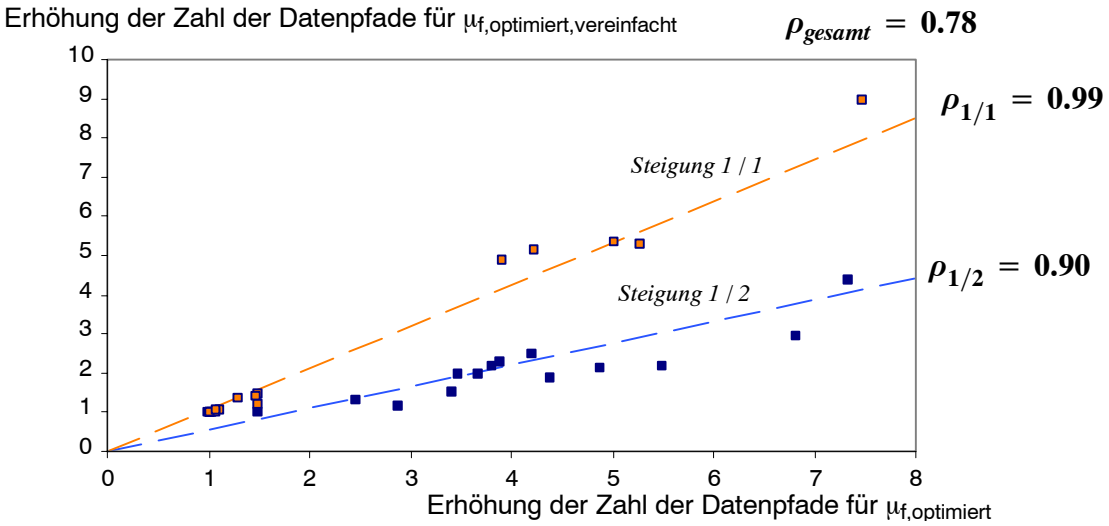


Bild 5. 21 . Erhöhung der Zahl der Datenpfade gegenüber veröffentlichten Daten durch Optimierung: $\mu_{f,optimiert}$: für bekannte Transistorzahlen; $\mu_{f,optimiert,\ vereinfacht}$: für vereinfacht modellierte Transistorzahlen der Verarbeitungseinheiten nach Tabelle 5.1 (*Fuzzy Zahl* V3).

Bild 5. 22 zeigt die für die untersuchten Architekturen berechneten Granularitäten in Verbindung mit den Steigungen der zugeordneten Trendlinien. Der Mittelwert der Granularität für *Steigung 1/1* ist 17. Für *Steigung 1/2* liegt der Mittelwert der Granularität bei 49.

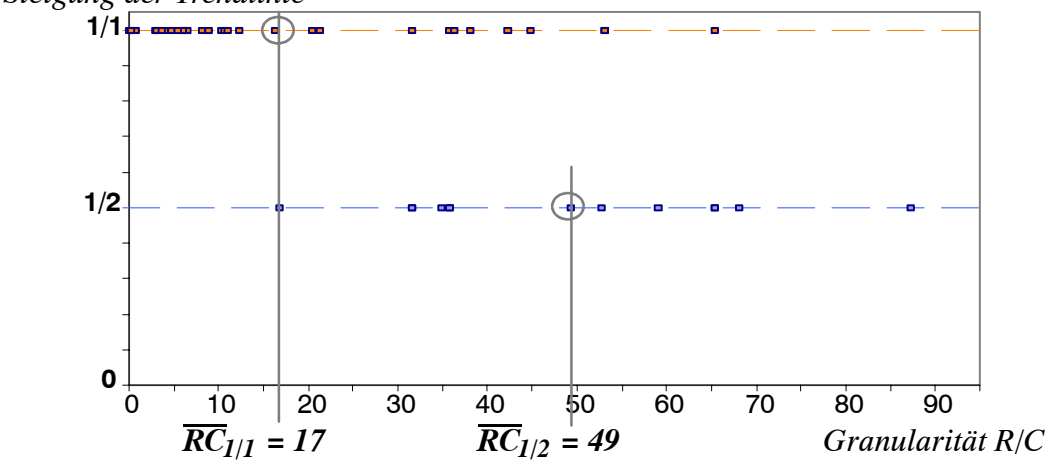


Bild 5. 22 . Granularitäten der optimierten Einzellösungen und Zuordnung zu den Trendlinien in Bild 5. 21 .

Das Ergebnis in Bild 5. 22 zeigt für die untersuchte Stichprobenmenge, daß bei unscharfen Transistorzahlen für die Verarbeitungseinheiten nach Tabelle 5.1 (Variante *Fuzzy Zahl V3*) bei kleineren Granularitäten der Prozessorelemente vergleichbare Ergebnisse erzielt werden, wie bei den bekannten Referenzdaten. Bei höheren Granularitäten zeigt sich, daß zwar einerseits die Optimierungsergebnisse als Erfüllungsgrad μ_f vergleichbar sind (Bild 5. 18), hier aber die resultierenden optimalen Datenpfadzahlen beim Übergang auf unscharfe Transistorzahlen für die Verarbeitungseinheiten um den Faktor 2 sinken.

6 Diskussion des entwickelten Modelles

6 .1 Diskussion der Ergebnisse

In Kapitel 4 ist ein neuer Modellansatz für die Kosten- und Performance-Modellierung von applikationsspezifischen VLSI-Architekturen entwickelt worden (Bild 4. 3). Im Kapitel 5 der Arbeit ist dieser Modellansatz auf veröffentlichte Beispielarchitekturen für programmierbare Videosignalprozessoren angewendet worden. Die geringe Größe der untersuchten Stichprobenmenge mit weniger als 30 Elementen führt dazu, daß eine statistisch abgesicherte Verifikation des vorgeschlagenen Ansatzes nicht möglich ist. Neben der untersuchten Stichprobenmenge gibt es noch etwa 40 weitere Veröffentlichungen zu Videosignalprozessoren, die aber nicht direkt auswertbar sind. Es fehlen jeweils Details, die für die Auswertung in dieser Arbeit benötigt werden. Selbst wenn es gelingen könnte, hier noch über Nachfragen bei den Autoren fehlende Informationen zu erhalten, bleibt das Problem, daß sich die Stichprobenmenge kaum mehr als verdoppeln kann und eine bessere statistische Basis dadurch nicht erwartbar ist. Die direkte Übertragung von Kosten- und Performance-Modellen für Mikroprozessoren [18] auf applikationsspezifische VLSI-Architekturen ist nicht sinnvoll. Es gibt hier signifikante Unterschiede der eingesetzten VLSI-Technologien. Auf der Anwendungsseite lassen sich Mikroprozessoren über die Konzentration auf *Floatingpoint*-Einheiten und *Caches* und ihre Performance für *SPEC-Benchmarks* untersuchen. Die Betrachtung applikationsspezifischer VLSI-Architekturen erfordert einen höheren Abstraktionsgrad, der die Charakterisierung und Modellierung vielfältiger Anwendungs- und Architekturvarianten unterstützt.

Sofern eine modellgestützte Untersuchung von VLSI-Architekturen auf Konzeptebene gewünscht ist, bleiben hier zwei grundlegende Möglichkeiten. Einmal können empirisch ermittelte oder angenommene Kennwerte in einer Modellierung verwendet werden, was in der Praxis häufig so durchgeführt wird. Eine hinreichende Genauigkeit der Ergebnisse ist hier in Frage zu stellen. Als Alternative bietet die in dieser Arbeit entwickelte Modellierung die Beschreibung möglicher Lösungsmengen, die nach [100] erwartete Lösungen einschließen können. In dieser Arbeit werden mögliche Lösungsmengen mit *Fuzzy Sets* beschrieben.

Performance-Modellierung

Im ersten Schritt ist im Hinblick auf die anwendungsabhängige Performance einer VLSI-Architektur der für eine Echtzeitverarbeitung erzielbare Datendurchsatz modelliert worden. Hier hat sich gezeigt, daß ein modellierter Datendurchsatz $R_{S,Modell}$, der sich an den Anforderungen der Algorithmen und am Charakter der Verarbeitung (skalare arithmetisch/ logische Operationen, Datenpfadoperationen) und an den Datenzugriffsanforderungen orientiert, einen etwa um den Faktor 400 geringeren Fehler nach Gleichung (5.2) aufweist als der einfachere Ansatz mit $R_{S,OpAS,max}$ (Bild 5. 1). $R_{S,OpAS,max}$ setzt Operationsraten einer Architektur

(häufig bezeichnet als *MOPS* oder *GOPS*) ohne Berücksichtigung der Operationen oder der Operationsfolgen mit den Anforderungen der Algorithmen in Beziehung. Trotz des viel größeren Fehlers ist die Anwendung von $R_{S,OpAS,max}$ dann noch vertretbar, wenn es um den Performance-Vergleich alternativer Lösungen geht. Hier zeigt das Ergebnis in Bild 5. 2 an der Korrelation der Werte und der Korrelation der Rangfolgen zwischen Referenz- und Modelldaten, daß beide Performance-Kriterien zu ähnlichen Vergleichsergebnissen für die untersuchten Architekturen führen. Wenn es um eine möglichst genaue Vorhersage des Datendurchsatzes geht, sollte man $R_{S,Modell}$ anwenden. Geht es darum, alternative Architekturen entsprechend ihren Performance-Unterschieden einzuordnen, genügt auch nach der untersuchten Stichprobenmenge das $R_{S,OpAS,max}$ -Modell.

Modellierung der Schaltkreisfläche als Kostenkriterium

Im Hinblick auf eine herstellerunabhängige Modellierung der Schaltkreisfläche ist nach Kapitel 5 .2 das in dieser Arbeit entwickelte *Bestcase*-Flächenmodell für Semicustom-Entwürfe *(Anhang E)* zu empfehlen.

Effizienz und Zielerfüllung bei einer Fuzzy Multikriterienanalyse

In Kapitel 5 .3 .2 konnte gezeigt werden, daß direkt aus Performance- und Kostenkriterien berechnete Effizienzen auf eine Fuzzy-Multikritereinanalyse abbildbar sind, die zu vergleichbaren Ergebnissen führen kann. Eine Fuzzy -Multikriterienanalyse kann daher als verallgemeinerte Effizienz betrachtet werden. Neben der Einbeziehung bekannter Effizienzbetrachtungen bietet die Multikriterienanalyse eine Vielzahl von Erweiterungsmöglichkeiten, z. B. in der Erhöhung der Zahl der berücksichtigten Kriterien.

Optimierung der Zielerfüllung einer Fuzzy Multikriterienanalyse

In Kapitel 5 .4 ist beispielhaft für bekannte Lösungen die Zielerfüllung in Abhängigkeit von der Zahl der parallelen Verarbeitungseinheiten und der Größen der lokalen On-Chip-Speicher mit einem genetischen Ansatz optimiert worden. Neben einem Datendurchsatz $R_{S,Modell}$ wurde auch das $R_{S,OpAS,max}$-Modell als alternative Performance Modellierung verwendet. Erwartungsgemäß zeigt sich hier, daß der Erfüllungsgrad der optimierten Lösungen für das $R_{S,OpAS,max}$-Modell durchgängig geringer ausfällt. Während nach $R_{S,Modell}$ sowohl die Zahl der Datenpfade moderat erhöht und die Größen lokaler Speicher moderat reduziert werden, weisen die unter Einbeziehung von $R_{S,OpAS,max}$ gewonnenen Ergebnisse kleinere Datenpfadzahlen und kleinere Speichergrößen auf.

Modellierung und Optimierung unter unscharfen Randbedingungen

In den Kapiteln 5 .5 und 5 .6 wurde in mehreren Varianten für die diskutierten Referenzlösungen die Zahl der Transistoren der Verarbeitungseinheiten vereinfacht modelliert. Diese Vereinfachungen wurden auf *Fuzzy Intervalle* abgebildet. Im Hinblick auf die Fehler der berechneten Erfüllungsgrade ergeben sich zwischen den verschiedenen Varianten kaum Unterschiede. Die Korrelationskoeffizienten zu den Referenzlösungen sind nahezu konstant.

Die Optimierung unter unscharfen Randbedingungen hat gezeigt, daß für die untersuchte Stichprobenmenge die optimierten Erfüllungsgrade in der gleichen Größenordnung liegen, wie bei bekannten Referenzdaten. In beiden Fällen werden Speichergrößen ähnlich reduziert. Ein signifikanter Unterschied ergibt sich bei der Zahl der Datenpfade. Feiner granulare Prozessoreinheiten werden in beiden Fällen gleich optimiert, bei steigender Granularität weisen die Lösungen bei Unschärfe eine geringere Zahl der optimierten Datenpfade auf.

6 .2 Ausblick für die Anwendung des entwickelten Modelles

Aus Sicht des Anwenders, bietet das entwickelte Modell eine Vielzahl von Möglichkeiten für die Auswahl und die Spezifikation von zu verarbeitenden Algorithmen, Architekturen und *VLSI-Technologien* bei gleichzeitig unscharf als *Fuzzy Sets* (Trapezform) spezifizierbaren Modellparametern (Bild 6. 1). Als Ergebnis bietet der Modellansatz einen aus der *Fuzzy-Multikriterienanalyse* resultierender Erfüllungsgrad μ_f, der mit einer Zahl zwischen 0 und 1 kennzeichnet, mit welchem Grad vom Anwender gestellte Ziele von einer untersuchten Architektur erfüllt werden. Alternativ können *Performance-* und *Kosten-Kriterien* und eine aus *Performance* und *Kosten* gebildete Effizienz η betrachtet werden. Sofern einzelne Architekturparameter mit Intervallen beschrieben sind, wird über Parametervariationen innerhalb der Intervallgrenzen eine Optimierung unterstützt. Ein zunächst über Parameterintervalle charakterisierter möglicher Lösungsraum, kann durch den Einsatz der Optimierung zielgerichtet eingegrenzt werden. Kapitel 5 hat gezeigt, daß bei unscharfen Lösungsmengen der Modellparameter, die mit *Fuzzy Sets* (Trapezform) berücksichtigt werden, eine realistische Chance besteht, brauchbare Modellierungsergebnisse zu erzielen.

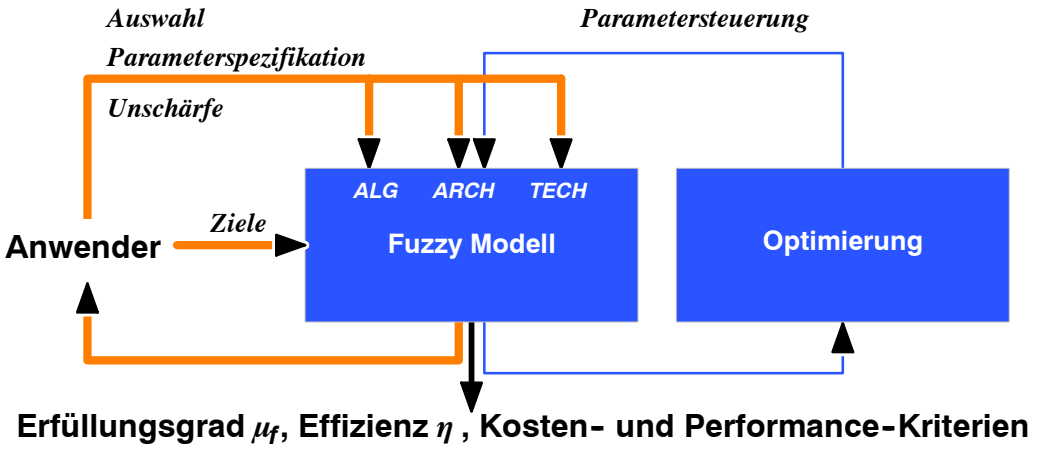


Bild 6. 1 . Der in dieser Arbeit entwickelte Gesamtansatz aus Sicht des Anwenders.
(*ALG*: Algorithmus, *ARCH*: Architektur, *TECH*: VLSI-Technologie)

Die Untersuchungen in Kapitel 5 haben sich primär auf die Verifikation des vorgeschlagenen Modellansatzes konzentriert. Es wurden verschiedene Videocodierungsverfahren und Architekturen modelliert und verglichen, die in unterschiedlichen *VLSI-Technologien* realisiert

wurden. Es hat sich gezeigt, daß die Unterschiede zwischen den betrachteten Referenzlösungen auch von den Modellierungsergebnissen vergleichbar gut dargestellt werden.

Die Entwicklung einer neuen applikationsspezifischen VLSI-Architektur und die damit verbundene Suche nach der besten Lösung erfordert die Betrachtung eines möglichen Entwurfsraumes aus verschiedenen Blickwinkeln. Hier hat Kapitel 5 gezeigt, daß der entwickelte Modellansatz eine besonders hohe Flexibilität bietet, alternative Sichtweisen eines Entwicklers in die Modellierung mit einzubeziehen. Hierzu gehören unterschiedliche Kosten- und Performance-Funktionen, abgestufte Unschärfen der angewendeten Modellparameter und die zielgerichtete Eingrenzung eines Lösungsraumes durch Optimierung.

In Kapitel 5 sind zu Verifikationszwecken alternative Lösungen zu Algorithmen, Architekturen und Technologien gemeinsam untersucht worden. Dabei sind Videocodecs die in 1 µm-Technologien hergestellt wurden mit aktuellen Videodecodern in 0.1 µm-Technologien verglichen worden. Sofern es nicht um die Verifikation eines Modelles sondern um der Entwicklung einer neuen *VLSI*-Architektur geht, wird es in der Praxis sinnvoller sein, die Zahl der gemeinsam betrachten Alternativen einzuschränken. Eine resultierende Aufschlüsselung und zugehörige Interpretationen möglicher Anwendungen des Modellansatzes zeigt Tabelle 6.1 (**X** : Festlegung auf eine Lösung, die parametrisierbar sein kann, a: Festlegung auf eine Menge mit alternativen Lösungen). In Zeile 1 findet sich die in Kapitel 5 diskutierte Modellverifikation wieder. In den Zeilen 2-8 folgen alternative Sichtweisen. Sofern die in Kapitel 5 diskutierte Verifikation gültig bleibt, ist zu erwarten, daß das entwickelte Modell zu den in Tabelle 6.1 angegebenen Anwendungen realistische Ergebnisse berechnet.

	ALG	**ARCH**	**TECH**	**Anwendung**
1	a	a	a	Modellverifikation
2	a	a	**X**	Abbildung einer Klasse von Algorithmen auf eine Menge von Prozessoren unter einheitlichen technologischen Randbedingungen (Referenzprozeß)
3	a	**X**	a	Abbildung einer Algorithmenklasse auf eine Prozessorarchitektur und Analyse der Skalierung für unterschiedliche Technologien
4	a	**X**	**X**	Abbildung einer Algorithmenklasse auf eine Prozessorarchitektur, die in einer vorgegebenen Technologie realisiert werden soll
5	**X**	a	a	Bewertung von verschiedenen, in unterschiedlichen Technologien realisierten Prozessoren für eine Applikation
6	**X**	a	**X**	Bewertung von verschiedenen, in einer Technologie realisierten Prozessoren für eine Applikation
7	**X**	**X**	a	Skalierung von Kosten und Performance einer Architektur für genau eine Anwendung für unterschiedliche Technologien
8	**X**	**X**	**X**	Konzeption und Optimierung einer Architektur für eine vorgegeben Applikation und eine vorgegebene Technologie

Tabelle 6.1 . Modellbasierte Erkundung möglicher Lösungsräume für den Architekturentwurf.
(X: Festlegung auf genau eine Lösung,
a : alternative Lösungen werden in der Modellierung gemeinsam betrachtet,
ALG: Algorithmus, *ARCH*: Architektur, *TECH*: VLSI-Technologie)

7 Zusammenfassung

In dieser Arbeit ist ein neuer Ansatz zur Kosten- und Performance-Modellierung von applikationsspezifischen VLSI-Schaltkreisen *(ASICs)* entwickelt und verifiziert worden. Im Gegensatz zu bekannten Mikroprozessormodellen, die sich auf wenige Kernmerkmale einer Architektur beschränken (*Floatingpoint-Einheiten* und *Caches*) und diese mit Anforderungen aus bekannten *SPEC-Benchmarks* in Beziehung setzen, zeigt diese Arbeit, daß für *ASICs* vielfältigere Alternativen zu Applikationen, Architekturen und VLSI-Technologien berücksichtigt werden müssen. Als Folge ergibt sich im Vergleich zu Mikroprozessormodellen eine deutlich höhere Komplexität mit zahlreichen analytischen Modellfunktionen.

Zusätzlich müssen in frühen Phasen der Entwicklung neuer Architekturkonzepte Unsicherheiten bei der Spezifikation von Modellparametern berücksichtigt werden. Auf Grund einer fehlenden statistischen Basis sind alternative Methoden zur Berücksichtigung dieser Unsicherheiten erforderlich. Daher werden in dieser Arbeit analytische Kosten- und Performance-Modelle auf *Fuzzy-Arithmetik* erweitert. Hier sind Zahlen als *Fuzzy Sets* spezifiziert, mit deren Hilfe eine abgestufte Zugehörigkeit von Modellparametern zur möglichen Lösungsmenge einer VLSI-Implementierung festgelegt wird.

Neben der Ermittlung von Kosten- und Performance-Kriterien unter unscharfen Randbedingungen wird in dieser Arbeit untersucht, wie im Hinblick auf nachfolgende VLSI-Implementierungen die beste Balance aus Kosten und Performance gefunden werden kann. Für in der Literatur veröffentlichte Effizienzmaße, wie eine gewichtete Summe aus Kosten und Verarbeitungszeit, Produktdarstellungen oder Quotienten aus Performance und Kosten, wird gezeigt, daß Äquivalenzen zwischen diesen Ansätzen herstellbar sind. Voraussetzungen sind geeignete Gewichte der Einzelsummanden. Unabhängig von der Wahl des spezifischen Effizienzmaßes liegt ein Problem in der Erweiterbarkeit um zusätzliche Kriterien. Hier bietet die *Fuzzy Set Theorie* mit einer *Fuzzy-Multikriterienanalyse* eine in der Zahl der berücksichtigten Kriterien nahezu unbegrenzt erweiterbare Lösung. Im Gegensatz zu direkt berechneten Effizienzmaßen werden hier Ziele zu einzelnen Kriterien mit Modellierungsergebnissen in Beziehung gesetzt und über alle Kriterien zu einem Erfüllungsgrad zusammengefaßt. Hier hat die vorliegende Arbeit am Beispiel realisierter Videosignalprozessoren gezeigt, daß eine *Fuzzy-Multikriterienanalyse* bei geeignet spezifizierten Zielen zu den gleichen Ergebnissen führt wie ein direkt bestimmtes Effizienzmaß. Auf Grund dieser Übereinstimmung, größeren Freiheitsgraden bei der Spezifikation von Entwurfszielen und der gleichzeitig besseren Erweiterbarkeit mit neuen Kriterien, kann eine *Fuzzy-Multikriterienanalyse* als verallgemeinerte Lösung zu bekannten Effizienzbegriffen aus der Literatur angesehen werden.

Im Rahmen der Anwendung des entwickelten analytischen, um Methoden der *Fuzzy Set Theorie* erweiterten Ansatzes, läßt sich eine Vielzahl an neuen Ergebnissen ableiten, die über einen Vergleich mit den Realisierungsdaten bekannter Lösungen verifiziert sind.

So ist ein neues, herstellerunabhängiges Bestcase-Modell zur Abschätzung der Schaltkreisgröße entwickelt worden, das den Einfluß der im Schaltkreis-Layout genutzten Metallisierungsebenen auf die erzielbaren Transistordichten erfaßt. Bereits realisierte *ASICs* können mit dem Flächenmodell klassifiziert werden, ob sie nahe bei den besten erzielbaren Transistordichten liegen oder weiter davon entfernt sind. In der Konzeptphase von Neuentwicklungen erlaubt der neue Flächenmodellierungsansatz eine einfach und schnell durchführbare Flächenabschätzung.

Im Hinblick auf Performance-Untersuchungen wird in dieser Arbeit der Datendurchsatz modelliert. Es wird berücksichtigt, welche arithmetischen Grundoperationen und welche Datenzugriffsmuster von den zu verarbeitenden Algorithmen gefordert sind.

Neben der Entwicklung eines geeigneten Modellansatzes liegt ein sehr hoher Aufwand in der Verifikation eines Kosten- und Performance-Modells. Daher wird der entwickelte Modellansatz auf der Basis veröffentlichter Daten von Videosignalprozessoren *(VSPs)* analysiert. Hier sind mit einem im Rahmen dieser Arbeit entwickelten Modellierungsprogramm *(VSP Decision Program)* unterschiedliche Modellsichten mit abgestuften Modellgenauigkeiten implementiert und verglichen worden.

Hierzu gehört der Vergleich des entwickelten Ansatzes zur Bestimmung des Datendurchsatzes mit einer vereinfachten, nur an den Operationsraten *(MOPS/ GOPS)* orientierten Lösung. Weitere Vergleiche zu alternativen Modellierungen beinhalten schrittweise vereinfachte Flächenmodellierungen und zugehörige Untersuchungen zur Multikriterienanalyse. In allen Fällen zeigt sich, daß Vereinfachungen schnell zu sehr großen Fehlern in den Modellierungsergebnissen führen. Gleichzeitig belegen aber berechnete Korrelationskoeffizienten zwischen bekannten und modellierten Daten, daß auch stark vereinfachte Modellierungen noch einen erkennbaren Zusammenhang mit bekannten Referenzdaten aufweisen.

In der Konzeptphase einer konkreten Architekturentwicklung steht häufig die Optimierung von Architekturparametern im Vordergrund. Eine Anwendung des Modells ergibt, daß sowohl bei Optimierungen mit bekannten Referenzdaten als auch bei stark vereinfacht modellierten Transistorzahlen für Verarbeitungseinheiten *(Fuzzy-Zahlen)* vergleichbare Ergebnisse erzielbar sind.

Diese Arbeit hat gezeigt, daß der Einsatz stark vereinfachter Modelle bestenfalls für den Vergleich realisierter Lösungen begründbar ist. Wenn es bei der Entwicklung neuer Architekturkonzepte um eine genauere Abschätzung erzielbarer Realisierungsdaten geht, ist die Anwendung der hier entwickelten, detaillierteren Kosten- und Performance-Modelle zu empfehlen. In Verbindung mit dem entwickelten Fuzzy-Multikriterienansatz und einer realisierten Optimierungslösung ist das wesentliches Ergebnis dieser Arbeit ein sehr leistungsfähiger Modellierungsansatz. Er unterstützt eine zielorientierte, an den Sichtweisen eines Entwicklers orientierte Bewertung und Optimierung von applikationsspezifischen VLSI-Architekturen.

8 Literatur

[1] S. Abeling, „Entwicklung einer generischen Modellbibliothek für die analytische Modellierung von applikationsspezifischen VLSI-Prozessoren", Studienarbeit, Institut für Mikroelektronische Systeme, April, 2004.

[2] B. Ackland et. al., „A Single-Chip, 1.6 Billion, 16-b MAC/s Multiprocessor DSP", *IEEE Journal of Solid-State Circuits*, No. 35, Vol. 3, S. 412-424, 2000.

[3] S. Agarwala et. al., „A 600 MHz VLIW DSP, *IEEE Journal of Solid-State Circuits*, No. 37, Vol. 11, S. 1532-1544, 2002.

[4] G. M. Amdahl, „Validity of the single-processor approach to achieving large scale computing capabilities", In AFIPS Conference Proceedings vol. 30 (Atlantic City, N.J., Apr. 18-20). AFIPS Press, Reston, Va., S. 483-485, 1967.

[5] K. Aono et. al., „A Video Digital Signal Processor with a Vector-Pipeline Architecture", *IEEE Journal of Solid-State Circuits*, No. 27, Vol. 12, S. 1886-1894, 1992.

[6] H. Arakida et. al., „A 160mW, 80nA Standby, MPEG-4 Audiovisual LSI with 16Mb Embedded DRAM and a 5GOPS Adaptive Post Filter, *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, 2003.

[7] H. B. Bakoglu, „Circuits, Interconnections and Packaging for VLSI", Reading: Addison-Wesley, 1990.

[8] E. Barke, „Brauchen wir eine europäische EDA Industrie?", GMM-Studie, VDI/ VDE Gesellschaft Mikroelektronik, Mikro- und Feinwerktechnik (GMM), Frankfurt am Main, Juni 1997.

[9] D. Bhandarkar, „Billion Transistor Processor Chips in Mainstream Enterprise Servers of the Future", Vortrag am Laboratorium für Informationstechnologie, Universität Hannover, 5. Mai, 2004.

[10] D. Brinthaupt et. al., „A Video Decoder for H.261 Video Teleconferencing and MPEG Stored Interactive Video Applications", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, S. 34-35, 1993.

[11] D. A. Carlson, R. W. Castelino, R. O. Mueller, „Multimedia Extensions for a 550-MHz RISC Microprocessor, *IEEE Journal of Solid-State Circuits*, No. 32, Vol. 11, S. 1618-1624, 1997.

[12] S. Dutta, R. Jensen, A. Rieckmann, „Viper: A multiprocessor SOC for advanced set-top box and digital TV systems", *IEEE Design & Test of Computers*, Vol. 18, No. 5, S.21 -31, 2001.

[13] J. C. Eble, „A Generic System Simulator With Novel On-Chip Cache and Throughput Models For Gigascale Integration", Dissertation Georgia Institute of Technology, 1998.

[14] M. Fares, B. Kaminska, „Exploring test space with fuzzy decision making",*IEEE Design & Test of Computers*, Vol. 11, No. 3(Fall), S. 17-27, 1994.

[15] M. J. Flynn, „Some Computer Organizations and their Effectiveness", IEEE Transactions on Computers, Vol. 21, No. 9, S. 948-960, September 1972.

[16] M. J. Flynn, „Computer Architecture - Pipelined And Parallel Processor Design". Jones And Bartlett Publishers, 1995.

[17] M. J. Flynn, P. Hung, A. Peymandoust, „Using Simple Tools To Evaluate Complex Architectural Trade-Offs", *IEEE Micro*, Vol. 20, No 4, S. 67-75, Juli/ August 2000.

[18] S. Fu, „Cost Performance Optimizations of Microprocessors", Dissertation, Stanford University, 2001.

[19] J. Geerlings, et. al., „A Single Chip MPEG2 Codec fo DVD+RW", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, 2003.

[20] B. Geuskens, K. Rose, „Modeling Microprocessor Performance", Kluwer Academic Publishers, Boston/ Dordrecht/ London, 1998.

[21] J. Goto et. al., „250-MHz BiCMOS Super-High-Speed Video Signal Processor (S-VSP)", *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 12, S.1876-1974, 1991.

[22] A. Götz, „Erstellung von Modellen zur Performance-Analyse von Videosignalprozessoren", Studienarbeit am Institut für Theoretische Nachrichtentechnik, 1998.

[23] O. Gottschalk, „Implementierung und Untersuchung von Genetischen Algorithmen für die Optimierung mit einem Programm zur Performancemodellierung", Laboratorium für Informatinstechnologie, Universität Hannover, 1999.

[24] E. R. Hansen, „Global Optimization Using Interval Analysis", Marcel Dekker, New York, 1992.

[25] J. Hartung, „Lehr- und Handbuch der angewandten Statistik", 12. Auflage, R. Oldenbourg Verlag, München, S. 125 ff., 1999.

[26] T. Hashimoto et. al., „A 27-MHz/54 MHz 11-mw MPEG-4 Video Decoder LSI for Mobile Applications", *IEEE Journal of Solid-State Circuits*, No. 37, Vol. 11, S. 1574-1581, 2002.

[27] K. Herrmann, J. Otterstedt, H. Jeschke, M. Kuboschek, „A MIMD-Based Video Signal Processing Architecture Suitable for Large Area Integration and a 16.6cm^2 Monolithic Implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 6, No. 2, S. 284-291, Juni 1998.

[28] I.-J. Huang et. al., „A cost effective Multimedia Extension to ARM7 Microprocessors", *IEEE Proceedings of the Symposium on Circuits And Systems (ISCAS)*, Vol 2, S. 360-363, 2002.

[29] H. Igura et. al., „An 800 MOPS 110 mW 1.5 V parallel DSP for mobile multimedia processing", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, S. 292-293, 1998.

[30] M. Ikeda et. al., „SuperENC: MPEG-2 video encoder chip", *IEEE Micro*, Vol. 19, No. 4, S. 56-65, Juli/ August, 1999.

[31] T. Inoue et. al., „A 300 MHz 16 b BiCMOS video signal processor", *IEEE Journal of Solid-State Circuits,* Vol. 28, No. 12, pp.1321-1330, 1993.

[32] International Technology Roadmap For Semiconductors 1999 - Design, http://www.itrs.org/ , 1999.

[33] International Technology Roadmap For Semiconductors 2003 - Design, http://www.itrs.org/ , 2003.

[34] S. Ishiwata et. al, „A Single-Chip MPEG-2 Codec Based on Customizable Media Embedded Processor", *IEEE Journal of Solid-State Circuits*, Vol. 38, No. 3, S. 530-540, 2003.

[35] H. Iwasaki et. al., „Single-chip MPEG-2 422P@HL Codec LSI with Multi-chip Configuration for Large Scale Processing beyond HDTV Level", Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'03), 2003.

[36] E. Iwata, „A 2.2 GOPS video DSP with 2-RISC MIMD, 6-PE SIMD architecture for real-time MPEG2 video coding/decoding", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, S. 258-259 und 469, 1997.

[37] H. Jeschke, K. Gaedke, P. Pirsch, „A VLSI based multiprocessor architecture for video signal processing ", *Proceedings of the 92' IEEE Symposium on Circuits and Systems,* Vol. 4, S. 1685-1688, Mai 1992.

[38] H. Jeschke, K. Gaedke, P. Pirsch, „Multiprocessor Performance for Real-Time Processing of Video Coding Applications", *IEEE Transactions on Circuits and Systems for Video Technology,* Vol. 2, No. 2, S. 221 - 230, Juni 1992.

[39] H. Jeschke, „Fuzzy Multiobjective Decision Making On Modeled VLSI Architecture Concepts," *Proceedings of the International Symposium on Circuits And Systems (ISCAS)*, Juni 1998.

[40] K. A. Jung, Y. S. Lee et. al., „An integrated H.263 video CODEC with protocol processor", *Proceedings of the IEEE International Symposium on Circuits and Systems*, Vol. 5, S. 283-286, 2001.

[41] J. M. Kheirandish, „Untersuchung von Approximationsverfahren für die Rechenzeitbestimmung von VLSI-Architekturen für die Echtzeitvideosignalverarbeitung", Diplomarbeit am Laboratorium für Informationstechnologie, Universität Hannover, 1995.

[42] B.-W., Kim, C.-M. Kyung, „Exploiting Intellectual Properties With ImpreciseDesign Costs for System-on-Chip Synthesis", IEEE Transactions On very Large Scale Integration (VLSI) Systems, Vol. 10, No. 3, S. 240-252, Juni 2002.

[43] G. J. Klir, U. H. St. Clair, B. Yuan „Fuzzy Set Theory", Prentice Hall PTR, Upper Saddle River, New Jersey, 1997.

[44] T. Koyama et. al., „A 250-MHz Single-Chip Multiprocessor for Audio and Video Signal Processing", *IEEE Journal of Solid State Circuits*, Vol. 36, No. 11, S. 1768-1774, 2001.

[45] C. M. Krishna (Ed.), „Performance Modeling for Computer Architects". *IEEE Computer Society*, Oct. 1995.

[46] H. Kropp, H. Jeschke, P. Pirsch, „Performancemodellierung von Multiprozessoranordnungen für die Echtzeitvideosignalverarbeitung". DFG-Abschlußbericht Pi 169/ 4, Hannover, August 1997.

[47] H. Kubosawa et. al., „A 1.2-W, 2.16-GOPS/720-MFLOPS embedded superscalar microprocessor for multimedia applications", *IEEE Journal of Solid State Circuits*, Vol. 33, No. 11, S. 1640-1648, 1998.

[48] H. Kubosawa, persönliche Mitteilungen, 2003.

[49] J. Küter, „Entwicklung und Anwendung eines objektorientierten Ansatzes zur Analyse von Algorithmen der Videosignalverarbeitung", Studienarbeit, Laboratorium für Informationstechnologie, Universität Hannover, 1996.

[50] S. Kumaki et. al., „A 99-mm^2 0.7-W single-chip MPEG-2 422P@ML video, audio, and system encoder with a 64-Mb embedded DRAM for portable 422P@HL encoder system", *IEEE Journal of Solid-State Circuits*, Vol. 37, No. 3, S. 450-454, 2002.

[51] K. Kutaragi et. al., „A microprocessor with a 128 b CPU, 10 floating-point MACs, 4 floating-point dividers, and an MPEG2 decoder", *IEEE Journal of Solid State Circuits*, Vol. 34, No. 11, S. 1608-1618, 1999.

[52] S. Kyo et. al., „A 51 GOPS Scalable Video Recognition Processor for INtelligent Cruis Control Based on a Linear Array of 128 4-Way VLIW Processing Elements", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, 2003.

[53] J. Labrousse, G. A. Slavenburg, „CREATE LIFE: A modular design approach for high performance ASICs", 35th IEEE Computer Society International Conference Compcon Spring '90, 'Intellectual Leverage', Spring 1990, S. 427-433, 1990.

[54] R. B. Lee, „Multimedia Extensions for General-Purpose Processors", *Proceedings of the IEEE Workshop on Signal Processing*, S. 9-23, November, 1997.

[55] L. A. Lev et. al., „A 64-b Microprocessor with Multimedia Support", *IEEE Journal of Solid State Circuits*, Vol. 30, No. 11, S. 1227-1238, 1995.

[56] David B. Lidsky, „The Conceptual-Level Design Approach to Complex Systems", PhD Thesis in Electrical Engineering and Computer Sciences, Universtiy of California, Berkeley, 1998.

[57] K. Lomtakul, „Using Simple Tools to Evaluate Complex Architectural Trade-Offs", Masters Thesis, Institut für Mikroelektronische Systeme, Universität Hannover, 2004.

[58] W. Maly, „IC Design in High-Cost Nanometer-Technologies Era", *Proceedings of the IEEE Design Automation Conference (DAC)*, 2001.

[59] C. Mead, L. Conway, „Introduction to VLSI Systems", Addison-Wesley, Reading, MA, 1980.

[60] T. Minami, et. al., „A 300-MOPS Video Signal Processor With A Parallel Architecture", *IEEE Journal of Solid State Circuits*, Vol. 26, No. 12, S. 1868-1887, 1991.

[61] A. Mohri et. al., „A Real-time Digital VCR Encode/ Decode and MPEG-2 Decode LSI Implemented on a Dual-Issue RISC Processor", *IEEE Journal of Solid State Circuits*, Vol. 34, No. 7, S. 992-1000, 1999.

[62] F. Mombers et. al., „IMAGE: A Low Cost, Low Power, Video Processor For High Quality Motion Estimation in MPEG-2 Encoding", *IEEE Transactions on Consumer Electronics,* Vol. 44, No. 3, S. 774-783, August 1998.

[63] G. E. Moore, „Cramming More Components onto Integrated Circuits, *Electronics*, April 1965, S. 114, 1965.

[64] G. E. Moore, „No Exponential is Forever: But "Forever" Can Be Delayed !", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, 2003.

[65] R. E. Moore, „Interval Analysis, Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[66] http://www.mosys.com, „1 T-SRAM Macro Cell", 2000.

[67] S.-I. Nakagawa et. al., „A 24-b 50-ns digital image signal processor", *IEEE Journal of Solid State Circuits*, Vol. 25, No. 6, S. 1484-1493, 1990.

[68] E. Ogura et. al., „A 1.2-W single-chip MPEG2 MP@ML video encoder LSI including wide search range motion estimation and 81-MOPS controller", *IEEE Journal of Solid State Circuits*, Vol. 33, No. 11, S. 1765-1771, 1998.

[69] M. Ohashi, „A 27MHz 11.1mW MPEG-4 video decoder LSI for mobile application", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, 2002.

[70] Y. Okada et. al., „An 80 mm^2 MPEG2 audio/video decode LSI", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, S. 264 -265 und 470, 1997.

[71] H. Okano et. al., „An 8-way VLIW embedded multimedia processor built in 7-layer metal 0.11µm CMOS technology", *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC),* Digest of Technical Papers, S. 374 -375, 2002.

[72] J. B. G. Roberts at. al., „Highly parallel processors in military systems", IEE Proceedings on Computer and Digital Techniques, Vol. 135, No. 4, S. 202-207, Juli 1988.

[73] A. Peleg, U. Weiser, „MMX Technology Extension to the Intel Architecture", IEEE Micro, Vol. 16, No. 4, S. 10-20, Juli/ August 1996.

[74] P. Pirsch, N. Demassieux, W. Gehrke, „VLSI Architectures for Video Compression - A Survey," Proceedings of the IEEE, Vol. 83, No. 2, S. 220-246, 1995.

[75] P. Pirsch, „Architekturen der digitalen Signalverarbeitung," Teubner, 1996.

[76] P. Pirsch et. al., „VLSI-Architectures for MPEG-4", *Proceedings of the 2003 International Symposium On VLSI Technology Systems And Applications*, S. 208A-208E, 2003.

[77] K. Roy, R. Roy, T.-L. Chou, „Design of Low Power Digital systems". In „Emerging Technologies", Tutorial ISCAS 1996, Editoren R. Cavin, W. Liu, S. 137-204, 1996.

[78] W. Rosenstiel, „Platform based design of embedded systems in SystemC", Vortrag zu *Medea+ conference*, 22.-25. Oktober, 2002.

[79] T. L. Saaty, „Exploring the Interface Betweeen Hierarchies, Multiple Objectives and Fuzzy Sets", Fuzzy Sets and Systems 1, Amsterdam: North-Holland, S. 57-68, 1978.

[80] M. Schwiegershausen, „Ein Verfahren zur Optimierung heterogener Multiprozessorsysteme mittels Linearer Programmierung", Dissertation, Universität Hannover, 1997.

[81] U. Sigmund, M. Steinhaus, T. Ungerer, „On Performance, Transistor count and Chip Space Assessment of Multimedia enhanced Simultaneous Multithreaded Processors", Workshop on Multi-Threaded Execution, Architecture and Compilation (MTEAC-4), Monterrey, Ca., Dec. 10, 2000.

[82] L. Sombé, „Schließen bei unsicherem Wissen in der Künstlichen Intelligenz", Friedr. Vieweg Verlagsgesellschaft mbH, Braunschweig/ Wiesbaden, 1992.

[83] ST Microelectronics, „Asic Solutions 90 nm CMOS 090 Design Platform", http://us.st.com/stonline/prodpres/dedicate/soc/asic/evol.htm, 2002.

[84] H.-J. Stolberg et. al., „HiBRID-SOC: A Multi-Core System-on-Chip Architecture for Multimedia Signal Processing Applications", Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'03), S. 8-13 suppl., 2003.

[85] H.-J. Stolberg et. al., „A Platform-Independent Methodology for Performance Estimation of Multimedia Signal Processing Applications", zur Veröffentlichung akzeptiert in *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, Kluwer Academic Publishers, 2004.

[86] H. S. Stone, „High-Performance Computer Architecture", Reading: *Addison-Weseley*, S. 309 - 325, 1992.

[87] A. Suga et. al., „A 4-way VLIW embedded multimedia processor", *IEEE International Solid-State Circuits Conference*, Digest of Technical Papers, S- 240-241 und 461, 2000.

[88] K. Suzuki et. al., „A 2000-MOPS Embedded RISC Processor with a Rambus DRAM Controller", *IEEE Journal of Solid State Circuits*, Vol. 34, No. 7, S. 1010-1021, 1999.

[89] M. Takahashi et. al., „A 60-mW MPEG4 Video Codec Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme", *IEEE Journal of Solid State Circuits*, Vol. 33, No. 11, S. 1772-1781 , 1998.

[90] H. Takata et. al., „The D30V/MPEG multimedia processor", *IEEE Micro,* Vol. 19, No. 4, S. 38-47, 1999.

[91] http://www.umc.com, „Chipsizer Online Design Tool", 2003.

[92] F. Vahid, T. Givargis, „Platform Tuning for Embedded Systems Design", *IEEE Compute*r, Vol, 34, No.3, S. 112-114, 2001.

[93] U. Vehlies, „Ein Verfahren zur schrittweisen Umsetzung hochsprachlich beschriebene Algorithmen in Array-Prozessorstrukturen", Dissertation, Universität Hannover, 1991.

[94] M. Wahle, „Entwurf einer Graphikoberfläche für ein Programm zur Performancemodellierung von VLSI-Architekturen", Studienarbeit, Laboratorium für Informationstechnologie, Universität Hannover, 1995.

[95] T. Wiegand, et. al., „Overview of the H.264/AVC Video Coding Standard", IEEE Transactions On Circuits And Systems For Video Technology, Vol.13, No.7, S. 560-576, 2003.

[96] J. P. Wittenburg, W. Hinrichs, J. Kneip, M. Ohmacht, M. Berekovic, H. Lieske, H. Kloos, P. Pirsch, „Realization of a Programmable Parallel DSP for High Performance Image Processing Applications," *Design Automation Conference (DAC) 1998*, S. 56-61, Juni 1998.

[97] J.-P. Wittenburg, „Rechenzeiten für den Entwurf des HiPar DSP4-Prozessors", persönliche Mitteilungen, 2000.

[98] M. Wosnitza, „Modellgestützte Architekturbewertung unter Berücksichtigung unscharfer Kosten- und Performance-Parameter", Diplomarbeit, Universität Hannover, 1994.

[99] R. R. Yager, „Fuzzy Decision Making Including Unequal Objectives," Fuzzy Sets and Systems 1, North-Holland, S. 87 - 95, 1978.

[100] H.-J. Zimmermann, „Fuzzy Set Theory and its Application". 2nd Edition, Kluwer Academic Publishers, Boston, 1991.

[101] L. A. Zadeh, „Fuzzy Sets”, Information and Control, Vol. 8, No. 3, 338-353, Juni 1965.

[102] M. Zeller, „Entwurf und Anwendung von objektorientierten Methoden zur Bewertung von Videosignalprozessoren”, Studienarbeit, Laboratorium für Informationstechnologie, Universität Hannover, 1996.

Anhang A: Software-Implementierung des entwickelten Modellansatzes

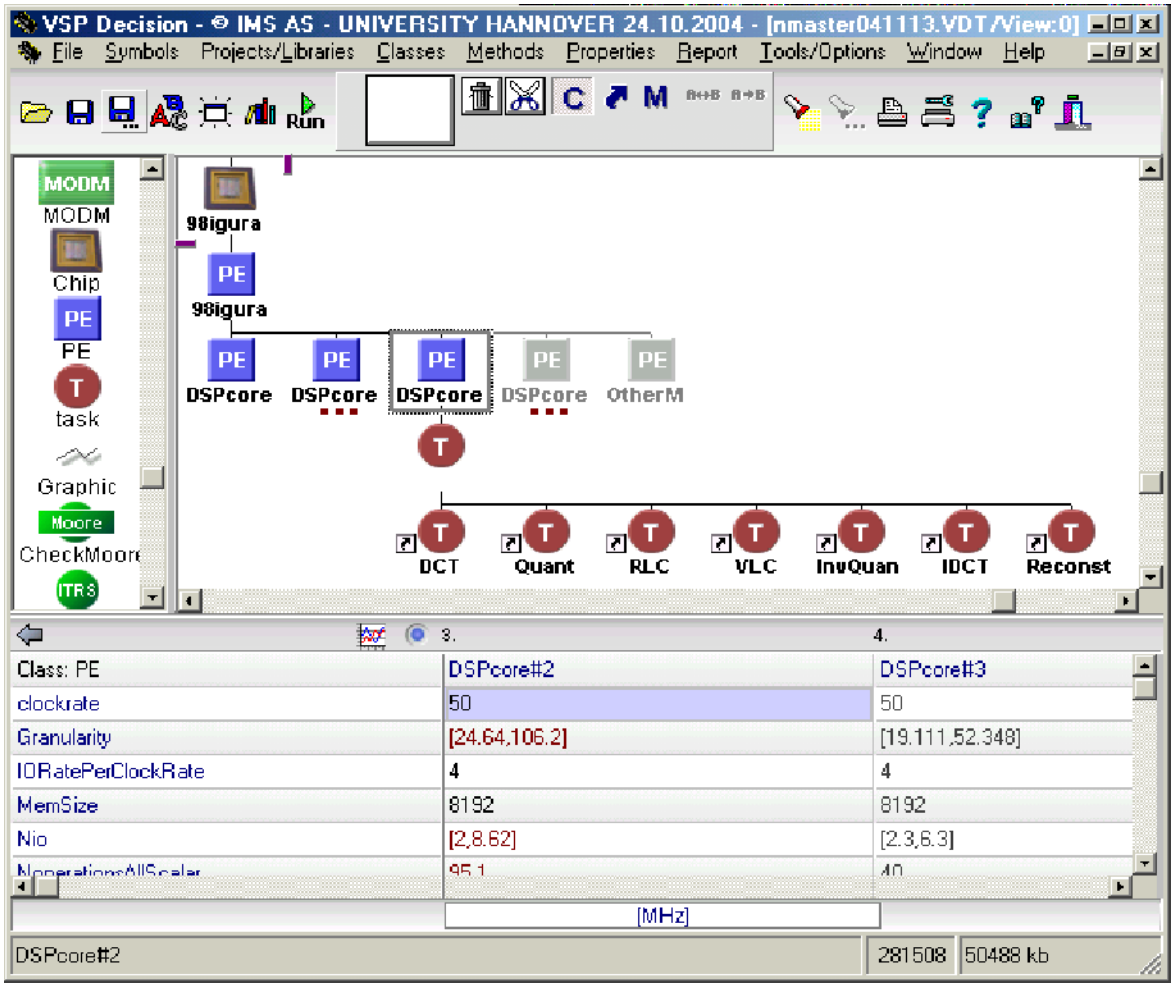


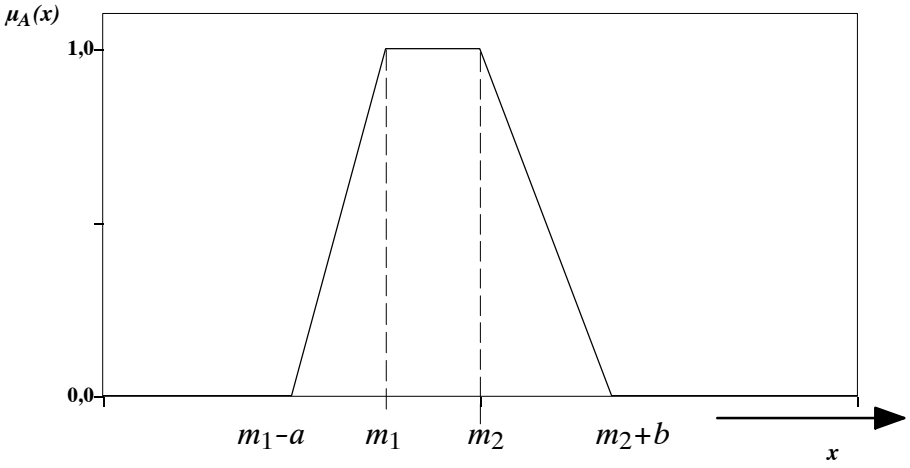
Bild A.1: Screenshot des entwickelten Programmes zur Architekturmodellierung mit *Fuzzy- Arithmetik* und *Fuzzy-Multikriterienanalyse* (VSP Decision Program).

Im Hinblick auf die hierarchische Struktur des generischen Architekturmodells (Bild. 2. 1), den Einsatz von *Fuzzy- Arithmetik* und *Fuzzy- Multikriterienanalyse* wurde im Rahmen dieser Arbeit und unter Beteiligung mehrerer Studienarbeiten [39], [94], [49], [102] ein spezielles Programm (*VSP Decision Program*) entwickelt, das die Modellierung von applikationsspezifischen Videosignalprozessoren unterstützt.

Wie in Bild A.1 dargestellt, können Architekturelemente mit einer Grafikoberfläche zueinander hierarchisch angeordnet werden. Den einzelnen Modellobjekten können spezifische Kennwerte (Eigenschaften) und Auswertemethoden zugewiesen werden. Sowohl die Objektstrukturen als auch die Methoden sind frei definierbar. Die Programmierung der Methoden erfolgt in einer Pascal-ähnlichen Hochsprache (*VPL*).

Die mit den Objekten verknüpften Methoden werden *bottom-up* in der Baumhierarchie eines Modells aufgerufen. So können Daten systematisch über die Hierarchie eines Modells zusammengefaßt werden. Die weiteren Untersuchungen in dieser Arbeit werden mit dem *VSP Decision Program* durchgeführt. Das *VSP Decision Program* besitzt zahlreiche Schnittstellen zu anderen Programmen, z. B. Tabellenkalkulationsprogrammen, die generell im Rahmen dieser Arbeit auch zum Aufbereiten und Nachbearbeiten von Modellierungsergebnissen genutzt wurden, ohne jeweils im Einzelnen angegeben zu sein.

Neben der Verarbeitung bekannter Datentypen, wie *Real, Integer, Boolean,* erlaubt das VSP Decision Program die Bearbeitung des Datentyps *Fuzzy*, der die nachfolgende Beschreibung unscharfer Zahlen als *Fuzzy Sets* in Trapezform unterstützt:



$$A = \left[m_1, m_2, a, b\right]$$

mit:

$$\mu_A(x) = \begin{cases} 0 & \text{, für } x \le m_1 - a \\ \frac{1}{a}(x - m_1 + a) & \text{, für } m_1 - a < x < m_1 \\ 1 & \text{, für } m_1 \le x \le m_2 \\ 1 - \frac{1}{b}(x - m_2) & \text{, für } x \ge m_2 + b \\ 0 & \text{, für } m_2 < x < m_2 + b \end{cases}$$

Beispiel für ein VPL-Modell einer Processing Unit (PU) mit N parallelen Submodulen:

```
{METHOD PU.EXE();}
//-------------------------------------------------------------------------------
// Beispiel:      Berechnung von Kostenfunktion:      Fläche A
//                Performance :                       Datendurchsatz Rs
//                Anzahl paralleler SubModule:        N
//-------------------------------------------------------------------------------

VarTree A,  Rs, A0, A1, P0, P1, N   : Fuzzy; // Objektparameter als Fuzzy- Zahlen
      Aziel, Rsziel                 : Fuzzy;
      Effizienz                     : Fuzzy;
      Erfuellungsgrad               : Fuzzy;

Var   ErfuellungsgradA,               // interne Variablen
      ErfuellungsgradRs             : Real;

{BEGIN}

      // Berechnung der Fläche und des Datendurchsatzes
      // Kostenfunktion C1, Performance P1 aus Tabelle 2.1

      A           := A0 + N * A1;
      Rs          := 1/ (1 /P0 + 1/ (N * P1));

      // Bestimmung der Effizienz

      Effizienz   := Rs/ A;

    // Multikriterienanalyse (kompensatorisch)

    ErfuellungsgradA:= MembAinB(A, Aziel);
    ErfuellungsgradRs:= MembAinB(Rs, Rsziel);

    Erfuellungsgrad := RealToFuzzy(ErfuellungsgradA*ErfuellungsgradRs);
{END;  //PU.EXE}
```

Anhang B: Beispiel zur Performance-Skalierung von Mikroprozessoren

Die Leistunsgfähigkeit von Mikroprozessoren wird oft mit bekannten Benchmarks analysiert ([18]). Benchmarks bestehen aus einer Mischung unterschiedlicher Teilaufgaben, die bis zur Performance-Analyse in Abhängigkeit des eingesetzten Festplattensystemes reichen können.

Begleitend zur Entwicklung des in Anhang A beschriebenen *VSP Decision Program* dieser Arbeit wurde an Hand eines einfachen Beispieles für verschiedene x86 Mikroprozessorgenerationen verfolgt, wie sich die steigenden Prozessortaktraten in der Praxis für rechenintensive Anwendungen auswirken, deren Daten während der Verarbeitung lokal im Hauptspeicher gespeichert werden können.

In diesem Beispiel wird mit dem in Anhang A beschriebenen *VSP Decision Program* eine nichtlineare Kostenfunktion mit 16 Parametern (x_i) minimiert (Gleichung B.1), deren Optimum von vornherein bekannt ist und somit eine einfache Überprüfung des Optimierungsergebnisses erlaubt :

$$Y = \sum_{i=0}^{16} (x_i - 15)^2 \ , Y \rightarrow Min! \tag{B.1}$$

Die Minimierung wird mit einem rechenintensiven Parametersuchverfahren nach Hansen durchgeführt [24]. Das Ergebnis zeigt für verschiedene Prozessorgenerationen (Bild B.1), daß sich die Verarbeitungsgeschwindigkeit etwa schritthaltend mit der Taktrate erhöht.

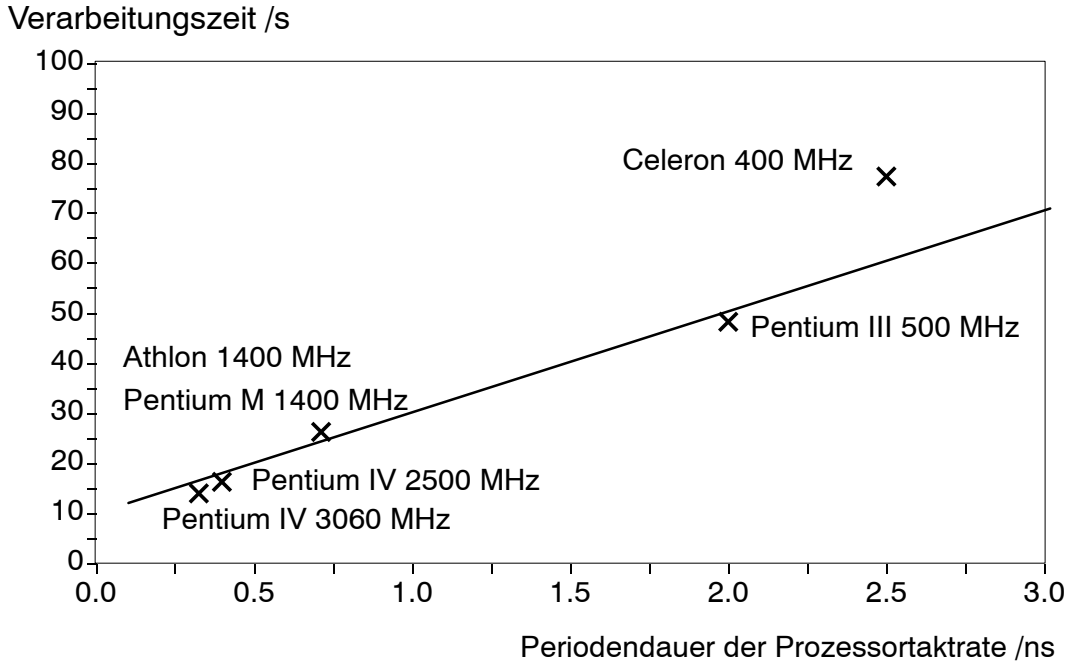


Bild B.1: Geschwindigkeitsteigerung schritthaltend mit der Prozessortaktrate.

Anhang C: Charakterisierung von Hybrid-Videocodierungsalgorithmen

Die nachfolgende Darstellung orientiert sich an einer Charakterisierung, die im Rahmen eines DFG-Projektes *Performancemodellierung* verwendet wurde [46] und die im Rahmen dieser Arbeit beispielhaft verwendet werden. Für den Entwurf neuer Prozessoren sind weitere Details zur Charakterisierung der Algorithmen erforderlich, zum Beispiel die Einbeziehung dynamischer, inhaltsabhängiger Häufigkeiten der Operationen [85].

In dieser Arbeit festgelegte Klassifikation der Verarbeitung :

Skalare Operation : Arithmetische Einzeloperationen, wie Addition (ADD), Subtraktion (SUB), Multiplikation (MUL), Akkumulation (ACC), SHIFT, LIMIT, ALU.

Datenpfad Operation : Feste Folge von arithmetischen Einzeloperationen, z. B. MUL-ACC-SHIFT-LIMIT

Encoder	N_{OP}, Zahl der ***Operationen pro byte***		
	$N_{OP,S}$: Skalar	**$N_{OP,DP}$: Datenpfad**	**$N_{OP,AS}$: Alle Skalar**
ME +/- 7	0.1	18	54.1
FILT/PRED	0	9	20
Transf. (DCT)	0	16	34
Q	6	1	10
IQ	0	1	4
Inv.Transf.(IDCT)	0	16	34
RLC	4	0	4
VLC	8.1	0	8.1
Reconstruction	1	0	1
Summe	**19.2**	**61**	**169.2**

Decoder	N_{OP}, Zahl der ***Operationen pro byte***		
	$N_{OP,S}$: Skalar	**$N_{OP,DP}$: Datenpfad**	**$N_{OP,AS}$: Alle Skalar**
VLD	8.1	0	8.1
RLD	4	0	4
IQ	0	1	4
Inv.Transf.(IDCT)	0	16	34
Reconstruction	1	0	1
FILT/PRED	0	9	20
Summe	**13.1**	**26**	**71.1**

Anhang C: Charakterisierung von Videocodierungsalgorithmen

Encoder	***(Speichergröße [byte];N_{IO})***		
ME +/- 7	(0 ; 32)	(256; 16)	(1156;2.995)
FILT/PRED	(0; 36)	(36; 2)	
Transf. (DCT)	(0;32)	(256;1)	
Q	(0;1)		
IQ	(0;2)		
Inv.Transf.(IDCT)	(0;32)	(256;1)	
RLC	(0;8)	(128;1)	
VLC	(0;54)	(128;1)	
Reconstruction	(0;1)		

Decoder	***(Speichergröße [byte];N_{IO})***		
VLD	(0;54)	(128;1)	
RLD	(0;8)	(128;1)	
IQ	(0;2)	1	4
Inv.Transf.(IDCT)	(0;32)	(256;2)	
Reconstruction	(0;2)		
FILT/PRED	(0;36)	(36;2)	

Tabelle C.1 :Für die Modellierung verwendete Zugriffsanforderungen eines Hybrid-Videocodierungsverfahrens. Für Speichergrößen, die die angegebenen Werte überschreiten, soll der letzte (kleinste) Wert für N_{IO} konstant gelten.

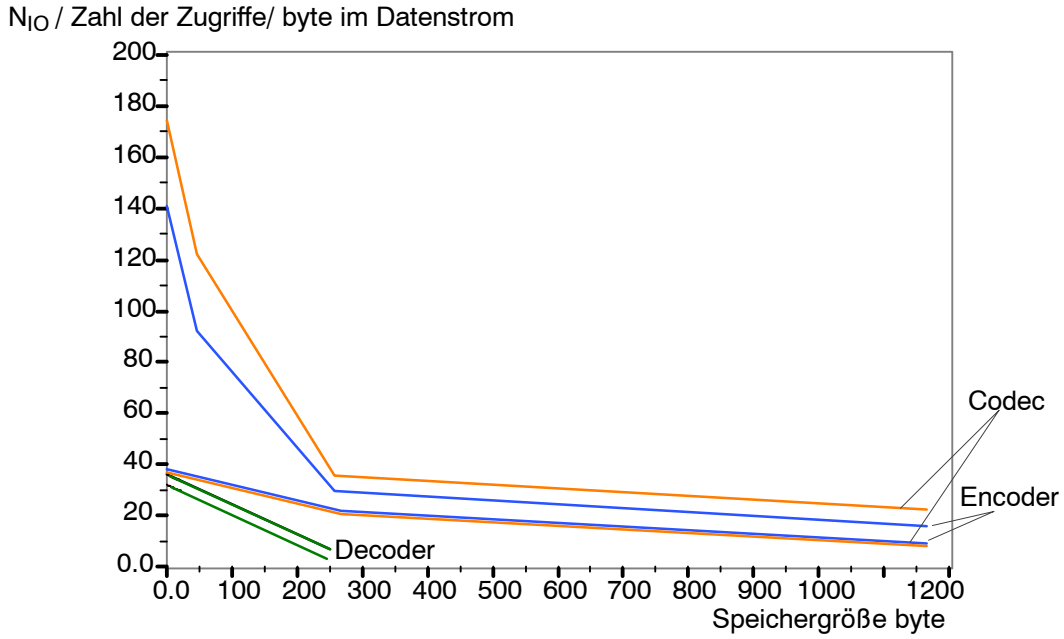


Bild C.1: Mit dem *VSP Decision Program* berechnete Gesamtzugriffsanforderungen.

Anhang D: Transistorzahlen für On-Chip SRAM-Speicher

In Gleichung (4.5) ist ein einfacher Ansatz für die Berechnung der Transistorzahlen für Speichermodule definiert worden. Nach [18] muß für lokale als Caches organisierte Speicher bei Mikroprozessoren ein zusätzlicher Aufwand für Steuerungslogik angerechnet werden. Daher ist für die Anwendung des vereinfachten Modelles nach Gleichung (4.5), das die Aufwendungen für die Ansteuerungslogik vernachlässigt, zu untersuchen, ob die getroffenen Vereinfachungen zulässig sind.

Daher wurde aus [2], [3], [5], [10], [12], [21], [27], [29], [30], [34], [36], [40], [44],[47] ,[50], [51], [60], [61], [62], [67], [68], [69], [70], [71], [87], [88], [89], [90], [35] analysiert, wieviele Transistoren pro gespeicherten bit anzurechnen sind.

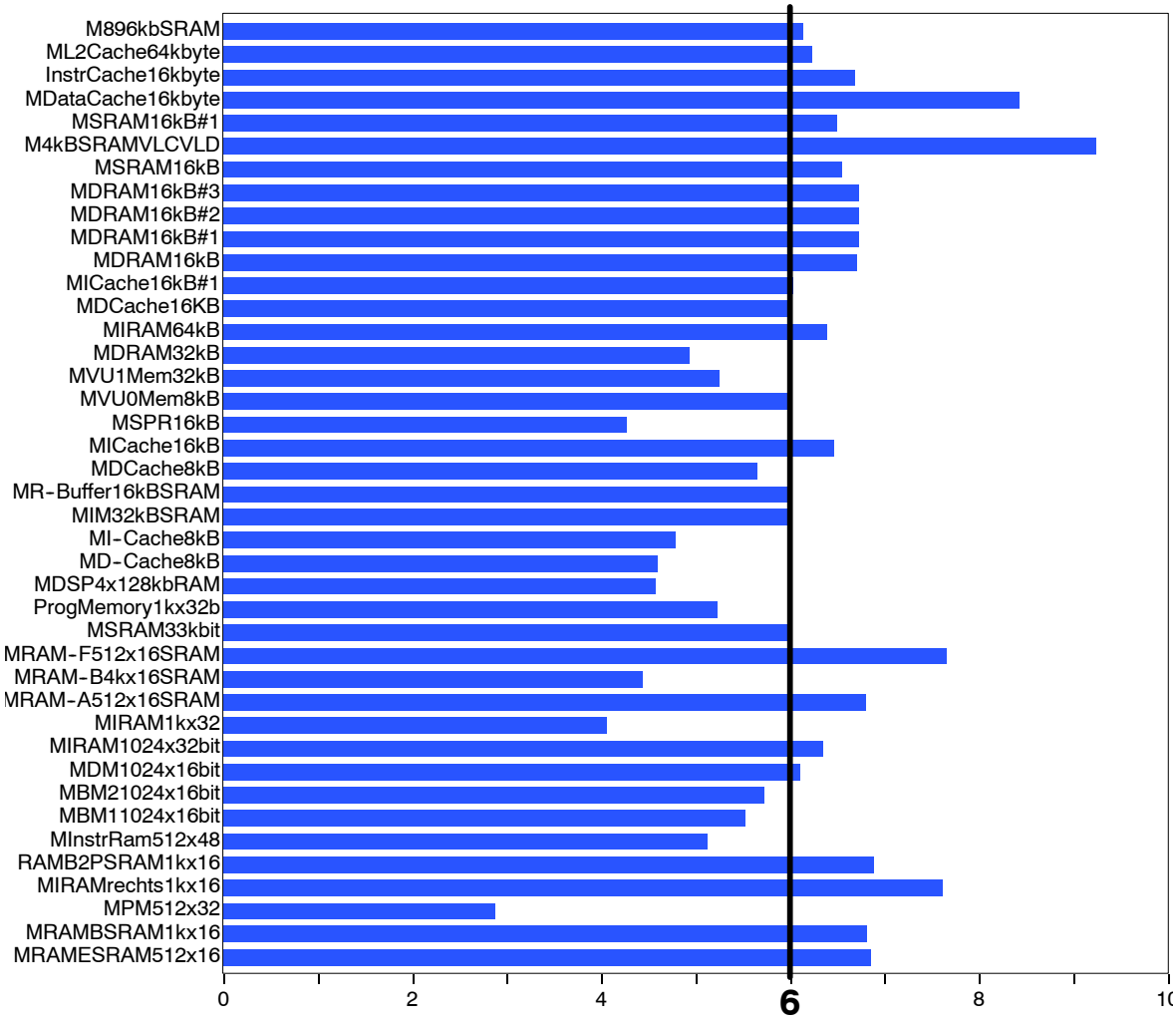


Bild D.1: Aus Layout-Fotos ausgemessene Zahl der Transistoren für Single-Port Speicher. Der Mittelwert ist 6.03 Transistoren pro bit.

Die Analyse wurde in mehreren Schritten durchgeführt. Zunächst sind im Layout eines Prozessors jeweils alle Flächen für Speicher ausgemessen und addiert worden. Im zweiten Schritt

wurde aus der Zahl der veröffentlichten Transistoren und den ausgemessenen Speicherflächen die Transistordichten für Speicher berechnet. Unter der vereinfachenden Annahme einer konstanten Transistordichte über alle Speichermodule ist dann für im Layout identifizierbare Speichermodule, für die Angaben zu Speichergrößen veröffentlicht wurden, eine geschätzte Zahl der Transistoren pro Speicherzelle berechnet worden. Bild D.1 zeigt das Ergebnis für Single Port Speicher. Bild D.2 stellt die Ergebnisse für Dual-Port Speicher dar. Der Mittelwert über alle Lösungen entspricht dem Modellansatz nach Gleichung (4.5). Abweichungen können über zu wenig berücksichtigte, zusätzliche Steuerungslogiken, 4-Transistorzellen, 1-Transistorzellen ([66]) oder über Ungenauigkeiten der Meßmethode erklärt werden.

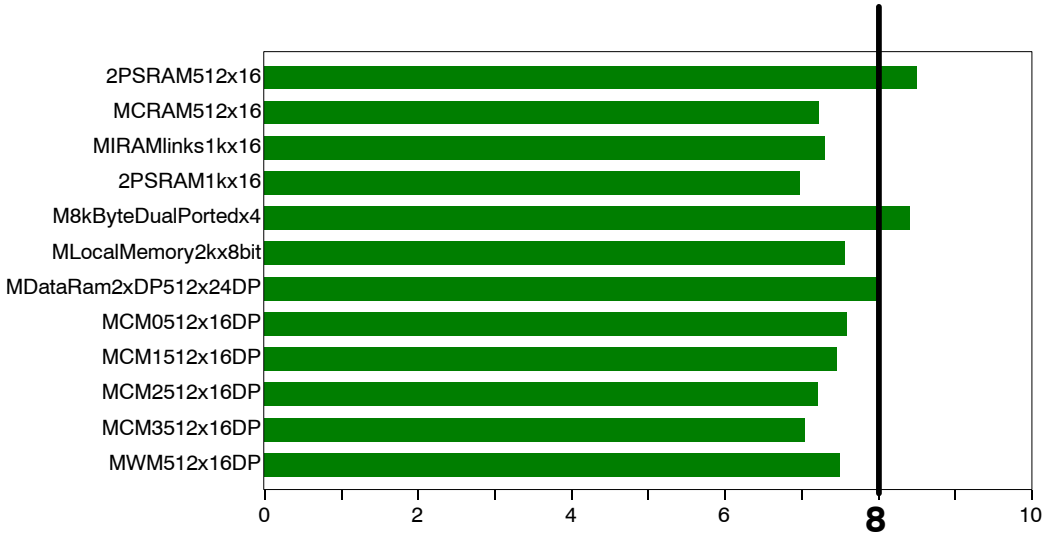


Bild D.2: Aus Layout-Fotos ausgemessene Zahl der Transistoren für Dual-Port Speicher. Der Mittelwert ist 7.56 Transistoren pro bit.

Anhang E: Flächenmodellierung im Überblick

Das Flächenmodell in dieser Arbeit basiert auf einer Analyse von *ASICs* für die Echtzeitvideosignalverarbeitung ([2], [3], [5], [10], [12], [21], [27], [29], [30], [34], [36], [40], [44],[47] ,[50], [51], [60], [61], [62], [67], [68], [69], [70], [71], [87], [88], [89], [90], [35]).
Die Schaltkreisgröße *(Die Size)* ergibt sich aus folgenden Parametern:

$$DieSize = \left(A_{PAD,0} + A_{PAD,1} \cdot s\right) \cdot N_{Pins} + GVF \cdot \left(\frac{N_{Trans,Arithmetik}}{d_{Norm,Arithmetik}} + \frac{N_{Trans,Speicher}}{d_{Norm,Speicher}}\right) \cdot \left(\frac{s}{0.1\mu m}\right)^2$$

s:	Strukturgröße (gezeichnete Gatterlänge) eines Halbleiterprozesses in µm
$A_{PAD,0}$	= 0.00112 mm^2
$A_{PAD,1}$	= 0.233 mm^2/ µm
N_{Pins}	Zahl der Pin-Anschlüsse oder PAD-Zellen eines Schaltkreises
GVF	= 1.27 (globaler Verdrahtungsfaktor für *BCMinF, MF, MIF*)
GVF_{BCF}	= [1.27, 1.27, 0.0635, 0.4064]
$N_{Trans,Arithmetik}$	Zahl der Transistoren für Arithmetik oder Logik
$N_{Trans,Speicher}$	Zahl der Transistoren für Speicher
$d_{Norm,Arithmetik}$	auf 0.1µm quadratisch normierte Transistordichte für Arithmetik/ Logik
$d_{Norm,Speicher}$	auf 0.1µm quadratisch normierte Transistordichte für Speicher
L	Zahl der jeweils für Arithmetik oder Speicher genutzten Metallisierungsebenen

Transistordichtenmodelle

Bestcase-Flächenmodell (BCF)

$$d_{Norm,Arithmetik,BCF} = 207400 \frac{Transistoren}{mm^2} 1.26^L$$

$$d_{Norm,Speicher,BCF} = 703120 \frac{Transistoren}{mm^2} 1.32^L$$

Bestcase-Minimum-Flächenmodell (BCMinF)

$$d_{Norm,Arithmetik,BCMinF} = [207400, 207400, 120000, 0] \frac{Transistoren}{mm^2} [1.26, 1.26, 0.06, 0]^L$$

$$d_{Norm,Speicher,BCMinF} = [774180, 774180, 508134, 0] \frac{Transistoren}{mm^2} [1.27, 1.27, 0.04, 0]^L$$

Mittelwert-Flächenmodell (MF)

$$d_{Norm,Arithmetik,MF} = 320000 \frac{Transistoren}{mm^2}$$

$$d_{Norm,Speicher,MF} = 990000 \frac{Transistoren}{mm^2}$$

Mittelwert-Intervall-Flächenmodell (MIF)

$$d_{Norm,Arithmetik,MIF} = [126, 508, 39, 7] \frac{10^3\ Transistoren}{mm^2}$$

$$d_{Norm,Speicher,MIF} = [442, 18733, 86, 101] \frac{10^3\ \ Transistoren}{mm^2}$$

Anhang F: Daten zur Analyse des *IMAP CE* Prozessors von *NEC*

	Veröffentlichte Werte [52]	Normierung auf 0.1μm	**Geschätzte Werte aus *BCF*-Modell (Anhang D)**
s (Strukturgröße)	0.18 μm		
L (Zahl der Metallisierungsebenen)	7		
Transistorzahl Arithmetik	$21.4 \cdot 10^6$		
Fläche Arithmetik	* ***61.9 mm²***		**66.3 mm²**
Transistordichte Arithmetik	345.8·103 Tr./ mm²	$1.120 \cdot 10^6$ Tr./ mm² ⇒ 7 Layer in *BCF*	
Transistorzahl Speicher	$11.6 \cdot 10^6$		
Fläche Speicher	* ***19.03 mm²***		** **17.6 mm²**
Transistordichte Speicher	$578.9 \cdot 10^3$ Tr./ mm²	$1.876 \cdot 10^6$ Tr./ mm² ⇒ 4 Layer in *BCF*	
GVF	* ***1.27***		
Fläche ohne PAD-Zellen	* ***102.79***		**106.6 mm²**
Pincount	500.0		
Fläche PAD Zellen	* ***18.21 mm²***		**21.5 mm²**
Gesamtfläche	* **121.0 mm²**		**128.1 mm²**
	Modellabweichung für Gesamtfläche : 5.87 %		

* Im Chip-Foto ausgemessene Layout-Daten.

** ***Annahme:*** Bei Speichermodulen nur Nutzung von 4 Metallisierungsebenen möglich

Anhang G: Tabelle zu äquivalenten Technologien nach dem *BCF-Modell*

Es werden mit dem in Anhang D spezifizierten ***BCF-Modell*** Schaltkreisflächen unter Variation der Prozeßparameter ***s*** und ***L*** berechnet.
Die Parameter ***s'***, $\mathbf{L'_{Arithmetik}}$ und $\mathbf{L'_{Speicher}}$, die zu der geringsten Abweichung zwischen bekannter und modellierter Schaltkreisfläche führen, charakterisieren eine nach dem ***BCF-Modell*** in der Fläche äquivalente Technologie.

		Veröffentlichte Werte		Äquivalente Technologie nach dem *BCF-Modell*		
	Referenz	***s*** **[µm]**	**L**	***s'*****[µm]**	$\mathbf{L'_{Arithmetik}}$	$\mathbf{L'_{Speicher}}$
1	[67]	1.0	2	1.2	2	2
2	[21]	0.8	3	1.0	3	2
3	[60]	0.8	2	0.8	1	1
4	[5]	0.8	2	0.8	2	2
5	[31]	0.5	3	0.8	2	1
6	[36]	0.4	3	0.5	3	1
7	[70]	0.5	3	0.5	2	2
8	[27]	0.8	4	0.8	4	3
9	[29]	0.25	3	0.3	3	2
10	[47]	0.21	4	0.35	3	2
11	[62]	0.35	5	0.4	4	3
12	[68]	0.4	3	0.4	3	3
13	[89]	0.3	3	0.35	2	2
14	[30]	0.25	4	0.3	4	2
15	[51]	0.25	4	0.3	3	3
16	[61]	0.25	4	0.25	3	3
17	[88]	0.25	4	0.3	2	2
18	[90]	0.25	4	0.25	4	2
19	[2]	0.25	4	0.35	3	3
20	[87]	0.18	5	0.18	3	3
21	[44]	0.25	4	0.25	3	2
22	[40]	0.35	4	0.35	4	3
23	[50]	0.13	4	0.13	2	2
24	[69]	0.18	4	0.18	4	4
25	[71]	0.11	7	0.18	4	4
26	[34]	0.18	6	0.21	6	5
27	[35]	0.13	8	0.13	5	3

Anhang H: Daten zur modellbasierten Performance-Analyse von *ASICs*

Die nachfolgenden Daten ergeben sich aus einer Modellierung nach Gleichung (4.6)-(4.15), Tabelle 4.2 , Bild 4. 6 . Es werden jeweils die bekannten, veröffentlichten Taktraten der Prozessoren für die Modellierung angewendet.

	Video Codecs	**$R_{S,bekannt}$ [Mbyte/s]**	**$R_{S,Modell}$ [Mbyte/s]**	**$R_{S,OpAS,max}$ [Mbyte/s]**
1	[67]	0.304	[0.338,0.358]	0.3044
2	[21]	2.92	2.90	2.54
3	[21] mit VLC VLD*	2.92	2.27	2.54
4	[60]	0.381	[0.417,0.469]	1.522
5	[5]	0.672	0.853	10.03
6	[27]	1.141	[0.931,0.977	3.25
7	[29]	0.570	[0.435, 0.933]	3.84

	Video Decoder	**$R_{S,bekannt}$ [Mbyte/s]**	**$R_{S,Modell}$ [Mbyte/s]**	**$R_{S,OpAS,max}$ [Mbyte/s]**
8	[21]	12.7	13.89	12.5
9	[5]	4	3.58	40.54
10	[47]	24.9	27.48	54
11	[61]	20.7	22.09	13.69
12	[88]	21	18.52	50
13	[44]	20.7	[27.74,30.86]	37.5
14	[84]mit VLC VLD*	20.7	11.82	52.4
15	[84]	20.7	23.83	52.4

	Video Encoder	**$R_{S,bekannt}$ [Mbyte/s]**	**$R_{S,Modell}$ [Mbyte/s]**	**$R_{S,OpAS,max}$ [Mbyte/s]**
16	[21]	3.8	3.67	3.18
17	[5]	0.81	1.12	11.32
18	[31]	4.56	3.74	9.55
19	[36]	15.55	13.39	16.42
20	[61]	10.14	15.19	10.23
21	[88]	21	7.59	10.69

*alternative Berechnung mit *VLC*/ *VLD*-Implementierung auf dem untersuchten ASIC, nicht berücksichtigt für die Berechnungen zu den Bildern 5. 1 und 5. 2 .

Tabelle H.1: Mit dem im Anhang A beschriebenen Modellierungsprogramm bestimmte Datendurchsätze für Videocodierungsanwendungen.

Anhang I : Modellierungsergebnisse zur Schaltkreisgröße

Referenz		Zahl der Transistoren / 10^3		*Die Size* Schaltkreisgröße in mm^2				
		Arithmetik und Logik	Speicher	veröffentlicht	BCF*	BCMinF*	MF	MIF
1	[67]	194	344	214	[208,208,10,66]	[151,151,0,196]	162	[113,334,2,112]
2	[21]	223	907	202	[191,191,10,61]	[115,115,0,180]	155	[99,333,2,105]
3	[60]	410	500	231	[221,221,11,71]	[186,186,0,234]	198	[139,407,2,141]
4	[5]	461	492	160	[176,176,9,56]	[176,176,0,254]	187	[125,415,2,156]
5	[31]	322	946	281	[210,210,10,67]	[77,77,0,86]	95	[69,181,1,53]
6	[36]	1310	2480	213	[210,210,11,67]	[120,120,0,184]	158	[103,347,2,123]
7	[70]	553	203	80	[83,83,4,27]	[71,71,0,85]	86	[63,177,0,66]
8	[27]	625	295	144	[143,143,7,46]	[140,140,0,216]	213	[142,484,2,195]
9	[29]	1268	3932	85	[86,86,4,27]	[56,56,0,86]	75	[49,162,1,53]
10	[47]	1664	836	100	[99,99,5,31]	[35,35,0,40]	49	[36,99,0,36]
11	[62]	500	1300	56	[56,56,3,18]	[37,37,0,44]	62	[43,124,1,39]
12	[68]	2856	1644	170	[181,181,9,58]	[181,181,0,296]	235	[151,552,2,226]
13	[89]	641	2359	81	[80,80,4,26]	[53,53,0,69]	68	[47,136,1,41]
14	[30]	2600	2400	96	[94,94,5,30]	[62,62,0,98]	96	[63,218,1,84]
15	[51]	6568	3932	240	[248,248,12,79]	[148,148,0,229]	226	[151,513,2,203]
16	[61]	881	5019	55	[57,57,3,18]	[48,48,0,71]	77	[50,160,1,47]
17	[88]	2327	1573	110	[110,110,5,35]	[54,54,0,83]	83	[55,186,1,73]
18	[90]	1232	5528	72	[69,69,3,22]	[56,56,0,86]	90	[58,191,1,59]
19	[2]	3985	2097	200	[206,206,10,66]	[95,95,0,137]	141	[96,312,1,122]
20	[87]	3200	3500	56	[56,56,3,18]	[41,41,0,56]	71	[49,151,1,55]
21	[44]	4018	6623	155	[149,149,7,48]	[117,117,0179]	182	[120,399,2,142]
22	[40]	475	770	42	[42,42,2,13]	[40,40,0,41]	55	[41,105,1,33]
23	[50]	7281	4719	57	[64,64,3,20]	[43,43,0,70]	67	[44,154,1,61]
24	[69]	2199	8801	37	[44,44,2,14]	[44,44,0,75]	74	[46,162,1,52]
25	[71]	3985	6921	61	[61,61,3,19]	[18,18,0,21]	39	[27,82,0,28]
26	[34]	4688	11250	72	[74,74,4,24]	[53,53,0,89]	122	[77,271,2,94]
27	[35]	35358	26042	196	[183,183,9,59]	[101,101,0,183]	325	[210,755,3,301]

Tabelle I.1: Mit dem im Anhang A beschriebenen Modellierungsprogramm berechnete Schaltkreisgrößen nach Kapitel 4 und nach Anhang E.

BCF: Bestcase Flächenmodell

BCMinF: Bestcase und Minimum Flächenmodell

MF: Mittelwertflächenmodell

MIF: Mittelwert-Intervall-Flächenmodell.

*: Einsatz äquivalenter Technologien nach *Anhang G*

Anhang J: Bekannte und modellierte Effizienz veröffentlichter *ASICs*

Die nachfolgenden Daten ergeben sich aus einer Modellierung nach Gleichung (4.6)-(4.15), Tabelle 4.2 , Bild 4. 6 , dem im ***Anhang E*** dargestellten ***BCF-Modell*** mit äquivalenten Technologiedaten und den Daten in den ***Anhängen H*** und ***I***, sowie den Daten der referenzierten Veröffentlichungen. Der Erfüllungsgrad μ_f ergibt sich durch die zusätzliche Verknüpfung der Modellierungsergebnisse nach Gleichung (3.3) mit den in Bild 5. 6 dargestellten Entwurfszielen (Gewichte $w_i = 1$). Sofern Ergebnisse als Fuzzy-Zahlen vorlagen, sind sie mit einer Flächenschwerpunktbildung des zugehörigen Trapezes auf reelle Zahlen abgebildet worden [100].

	Video Codecs	**R_S / Die_{Size}** [byte/s / mm²]	**$R_{S,Modell}$/ $Die_{Size,BCF}$** [byte/s / mm²]	**$\mu_f = f(R_S, Die_{Size})$**	**μ_f ($R_{S,Modell}$, $Die_{Size,BCF*}$)**
1	[67]	1423	1568	0.0029	0.0034
2	[21]	14487	13477	0.0356	0.0362
3	[21] mit VLC VLD**	14487	10520	0.0356	0.0283
4	[60]	1645	1880	0.0023	0.0024
5	[5]	4201	4886	0.0140	0.0330
6	[27]	7920	6343	0.0276	0.0319
7	[29]	6737	8412	0.0208	0.0205

	Video Decoder	**R_S / Die_{Size}** [byte/s / mm²]	**$R_{S,Modell}$/ $Die_{Size,BCF}$** [byte/s / mm²]	**$\mu_f = f(R_S, Die_{Size})$**	**μ_f ($R_{S,Modell}$, $Die_{Size,BCF}$)**
8	[21]	63208	64467	0.1552	0.1734
10	[5]	25008	20489	0.0834	0.0997
11	[47]	249880	259346	0.6684	0.8768
12	[61]	374026	365951	0.8523	0.9752
13	[88]	191237	167697	0.6240	0.7988
14	[44]	133486	223366	0.4361	0.7811
15	[84]mit VLC VLD**	252878	169068	0.7417	0.5680
16	[84]	252878	340954	0.7417	0.9612

	Video Encoder	**R_S / Die_{Size}** [byte/s / mm²]	**$R_{S,Modell}$/ $Die_{Size,BCF}$** [byte/s / mm²]	**$\mu_f = f(R_S, Die_{Size})$**	**μ_f ($R_{S,Modell}$, $Die_{Size,BCF}$)**
17	[21]	18794	17039	0.0462	0.0458
18	[5]	5049	6416	0.0168	0.0312
19	[31]	16264	16758	0.0000	0.0335
20	[36]	73043	56401	0.1527	0.0770
21	[61]	183578	251591	0.4320	0.7406
22	[88]	191237	68727	0.6420	0.3274

**alternative Berechnung mit *VLC* und *VLD* Implementierung auf dem untersuchten ASIC.

Anhang K: Realisierte Transistorzahlen für Arithmetik/ Logik

	Referenz	Datenpfad*	Skalare Einheit**	Sonstige Logik
1	[67]	51897		142038
2	[21]	42323	27350	153063
4	[60]	73300		116800
5	[5] Core Processor	49590		685156
	[5] DCT Core			23028
6	[31]	66968	29560	226655
7	[36] PE	100000		63750
	[36] VLCVLD	164739		
	[36] Control RISC			163750
	[36] Andere			240704
8	[27]	137005		488880
9	[29]			1000000
10	[47] Control Unit			1000000
	[47] Floating Point Unit	441843		
	[47] SIMD Datapath	124000		
10	[61] dRISC			330000
	[61] VLCVLD	330000	165495	
11	[88]	276000		1450000
12	[44]	355215		330000
13	[84] Stream Processor		920533	
	[84] MacroBlock Engine	1246000		300000
	[84] HiPar	233163		1489362

	Minimum	42323	27350	23028
	Mittelwert	246136	285735	482540
	Maximum	1246000	920533	1489362

In dieser Arbeit festgelegte Klassifikation von Verarbeitungseinheiten:

* Datenpfad: Einheit für eine Pipeline-Verarbeitung einer Folge von arithmetischen Einzeloperationen, z. B. MUL-ACC-SHIFT-LIMIT

** Skalare Einheit: Einheit für arithmetische Einzeloperationen, wie Addition (ADD), Subtraktion (SUB), Multiplikation (MUL), Akkumulation (ACC), SHIFT, LIMIT, ALU.

Lebenslauf

Hartwig Jeschke
geboren am 22.12.1960 in Kellinghusen/ Kreis Steinburg

Schulausbildung:

1966 – 1971	Besuch der Grund– und Hauptschule in Burg auf Fehmarn
1971 – 1980	Besuch des Insel–Gymnasiums in Burg auf Fehmarn

Berufsausbildung:

1982 – 1987	Studium der Elektrotechnik an der Universität Hannover
1987	Abschluß des Studiums mit dem Diplom

Berufstätigkeit:

1980 – 1982	Bundeswehr: Ausbildung und Tätigkeit als Beobachtungsfunkmechaniker
1988 – 2000	Wissenschaftlicher Mitarbeiter am Institut für Theoretische Nachrichtentechnik der Universität Hannover
Ab 1992	Akademischer Rat
Seit 1994	Akademischer Oberrat
Seit 2000	Akademischer Oberrat und wissenschaftlicher Mitarbeiter am Institut für Mikroelektronische Systeme der Universität Hannover