Zur systematischen Ermittlung Hardware-geeigneter Zahlendarstellungen für Algorithmen der digitalen Signalverarbeitung

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

von
Dipl.-Ing. Steffen Blume

geboren am 27.08.1981
in Stadthagen

2015

Berichte aus der Informationstechnik

Steffen Blume

Zur systematischen Ermittlung Hardware-geeigneter Zahlendarstellungen für Algorithmen der digitalen Signalverarbeitung

Shaker Verlag
Aachen 2015

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Hannover, Leibniz Univ., Diss., 2015

Printed in Germany.

ISBN 978-3-8440-3691-6
ISSN 1610-9406

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen
Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Danksagungen

Zuvorderst danke ich Herrn Prof. Dr.-Ing. Peter Pirsch für die Möglichkeit am Institut für Mikroelektronische Systeme der Gottfried Wilhelm Leibniz Universität arbeiten und in diesem Rahmen promovieren zu können. Darüber hinaus danke ich ihm für die Übernahme des ersten Referats sowie kritische, fordernde und fördernde Diskussionen über die vorliegende Arbeit. Weiterhin bedanke ich mich bei Herrn Prof. Dr.-Ing. Lars Hedrich für die Übernahme des zweiten Referats. Herrn Prof. Dr.-Ing. Holger Blume danke ich für die Übernahme des Vorsitz der Prüfungskommission, für die Beschäftigung am Institut und gewinnbringende Diskussionen.

Meinen Kollegen am Institut danke ich für die gute Zusammenarbeit, die weiterführenden Fachgespräche und das angenehme Miteinander. Die Zeit am Institut wird mir zu großem Anteil auf Grund der gemeinsamen Aktivitäten in guter Erinnerung bleiben.

Insbesondere möchte ich mich bei Jun.-Prof. Dr.-Ing. Guillermo Payá Vayá für die Kooperation bedanken, ohne die die Untersuchungen dieser Arbeit weit weniger ausführlich geworden wären. Weiterhin gebührt mein Dank M. Sc. Daniel Pfefferkorn, Dipl.-Ing. Nico Mentzer und Dipl.-Ing. Rochus Nowosielski für die Durchsicht des Manuskripts und die wertvollen Korrekturen und Anregungen.

Meinen Eltern kann ich nicht genug danken für die vielfältige Unterstützung auf meinem gesamten Bildungsweg. Sie haben mich bereits früh an die Elektrotechnik herangeführt und meine ersten Kontakte mit der Informatik ermöglicht. Danke.

Steffen Blume
Hannover, Mai 2015

Kurzfassung

Eingebettete Systeme der digitalen Signalverarbeitung unterliegen herausfordernden Anforderungen bezüglich Herstellungskosten, Verlustleistung, Latenz und Durchsatz der Datenverarbeitung. In Gleitkommaarithmetik mit doppelter Genauigkeit entworfene Verarbeitungsalgorithmen müssen für ihre Umsetzung als eingebettetes System auf eine Arithmetik, die die Randbedingungen erfüllt, umgestellt werden. In dedizierten wie auch in Implementierungen auf programmierbaren Architekturen kommen zumeist Festkommadarstellungen zum Einsatz, da aus deren Verwendung geringere Latenz, Chipfläche sowie Verlustleistung gegenüber Gleitkommadarstellungen resultieren. Jedoch wird auch deren Vorteil des größeren Dynamikumfangs durch die Fortschritte in der Technologie bei angepassten Bitbreiten für Mantisse und Exponent realisierbar. Zur Konvertierung der Arithmetik unter Reduktion der Hardware-bezogenen Kosten muss für jeden darzustellenden Wert das Zahlenformat und die Bitbreiten festgelegt werden, so dass die globalen Anforderungen nur knapp erfüllt werden. Die Bitbreiten können aus den Anforderungen bezüglich Wertebereich und Genauigkeit an jede einzelne Variable im Algorithmus ermittelt werden. Die Ableitung aus den globalen Anforderungen nimmt während der Konvertierung des Algorithmus die meiste Zeit in Anspruch. Allgemeingültige Lösungen für beliebig beschaffene Algorithmen existieren nicht. Vorgeschlagene Ansätze beschränken sich auf lineare oder differenzierbare, zeitinvariante Systeme und vorgegebene Metriken für die Festlegung der Anforderungen.

Das Thema der vorliegenden Arbeit ist daher eine werkzeuggestützte Methode zur Konvertierung der Gleitkommaarithmetik eines gegebenen Algorithmus in eine Arithmetik geeignet für die Realisierung als eingebettetes System. Hierbei wird eine breite Kategorie von Algorithmen unterstützt. Insbesondere unterstützt die Methode auch die Handhabung von datenabhängigem Kontrollfluss. Die Bewertungsmetrik zur Festsetzung der Anforderungen wird nicht vorgegeben und kann somit an den Anwendungsfall angepasst werden. Bekannte, automatische Analysetechniken wurden mit einem Gesamtkonzept vereinheitlicht und für den Einsatz innerhalb der Methode angepasst und erweitert. Weiterhin bezieht die Methode die Evaluierung von Näherungen und spezifische Umsetzungen von mathematischen Funktionen in den Optimierungsprozess mit ein. Eine Funktionsbibliothek für den Einsatz innerhalb der Methode wurde erstellt. Bei der Konzeption der Methode wurde auf eine Reduktion des manuellen Aufwands zur Aufbereitung der Algorithmen geachtet.

Die Methode beweist in drei praxisrelevanten Fallstudien ihre Anwendbarkeit: Für Algorithmen mit reinem Datenfluss sowie datenabhängigem Kontrollfluss können kostenreduzierende Zahlendarstellungen bei Fest- und Gleitkommaarithmetik ermittelt werden.

Schlagworte: Zahlensysteme - Wortbreitenoptimierung - automatische Analysetechniken

Abstract

Embedded systems for digital signal processing are subject to challenging requirements concerning production costs, power dissipation, latency, and data throughput. Floating-point arithmetics with double precision used throughout algorithms during their development have to be converted to arithmetics fulfilling the requirements. In dedicated as well as in implementations based on programmable architectures fixed-point arithmetics are used because of their shorter latency, smaller silicon area, and reduced power dissipation compared to floating-point implementations. Nevertheless, their benefit of the higher dynamic range becomes viable with exponent and mantissa bit-widths adapted to the algorithm because of the progress in technology. During the conversion of arithmetics a number format along with bit-widths has to be assigned to each number to represent, in a way that the global requirements are tightly met in order to reduce the hardware related costs. The bit-widths are determined from the range and precision requirements to each variable in the algorithm. The derivation of the individual from the global requirements consume the most time during the conversion. There is no universal solution to this problem. The proposed approaches are limited to linear or differentiable, time-invariant systems and predetermined metrics for the definition of the requirements.

Therefore, the topic of this thesis is a computer-aided method for converting the floating-point arithmetics of a given algorithm to convenient ones for the realization of embedded systems. A wide varity of algorithms is supported. Notably the method can also handle data-dependend control flow. The metric for defining the requirements is not predetermined. Known automatic analysis techniques were unified with an overall concept, adapted, and extended for their use in the method. In addition the method incorporates the evaluation of approximations and specific realizations of mathematical functions during the optimization process. A function library for the use in the method was created. During the design of the method attention was paid to the reduction of the manual effort for the prepartion of the algorithm.

The method proofs its operability in three case studies: For algorithms composed of pure data flow and data-dependend control flow number representations reducing the hardware-related costs were determined using fixed- or floating-point arithmetics.

Keywords: number systems - wordlength optimization - automatic analysis techniques

Inhaltsverzeichnis

Abkürzungsverzeichnis

AA	*Affine Arithmetik* Arithmetik für affine Formen
AccAbsDiffError	*Accumulated absolute Differences of required and resulting Error* Bewertungsmaß des erzielten Fehlers während einer Suche
AccError	*Accumulated Error* Bewertungsmaß des erzielten Fehlers während einer Suche
AF	*Affine Form* Darstellung eines Wertes in der affinen Arithmetik
AF1	*Affine Form 1* Kategorie 1 der affinen Formen nach [1]
AGE	*Adaptive Gain Equalizer* Verfahren zur Rauschunterdrückung in Sprachsignalen
ASA	*adaptives simuliertes Abkühlen, adaptive simulated annealing* Suchheuristik, welches dem Prozess des langsamen Abkühlens von Metallen nachgebildet wurde
ASIC	*Application-specific Integrated Circuit* elektronische Schaltung als anwendungsspezifischer, integrierter Schaltkreis
AT-Produkt	*Area-Time-Produkt* Produkt aus benötigter Fläche und Taktperiode einer synthetisierten Implementierung
BRAM	*Block Random Access Memory* größere Speicherblöcke in einem FPGA

CAD	*Computer-Aided Design* rechnerunterstütztes Konstruieren
CDFG	*Control and Dataflow Graph* Kontrolldatenflussgraph
CPU	*Central Processing Unit* Prozessor
DANSE	*Digital Arithmetic and Number System Evaluation* Framework zur Realisierung der in dieser Arbeit vorgeschlagenen Methode
DBD	*Digit-by-Digit* Verfahren zur bitweisen Berechnung von Division oder Wurzelfunktion
DCD	*Dichotomous Coordinate Descent* Verfahren zur Approximation von lokalen Minima
DCT	*Discrete Cosine Transform* Transformation, welche ein diskretes Zeitsignal auf ein diskretes Frequenzspektrum abbildet. Sie unterscheidet sich von der DFT hinsichtlich der Randfortsetzung
DFT	*Discrete Fourier Transform* Transformation, welche ein diskretes, periodisches Zeitsignal auf ein diskretes, periodisches Frequenzspektrum abbildet
DSG	*Darstellungsstellengraph* Ein dem Datenflussgraphen ähnlicher Graph, in dem die Kanten den DSS des Algorithmus entsprechen
DSP-Block	*Digital Signal Processing Block* Komplexe Bausteine in einem FPGA geeignet zur Realisierung von Filtern und Transformationen

DSS	*Darstellungsstelle* Stelle im Datenfluss, an dem Werte dargestellt werden
FF	*Flip Flop* Speicher für ein Bit
FFT	*Fast Fourier Transform* Schneller Algorithmus nach dem Teile-und-herrsche-Verfahren zur Berechnung der DFT
FIR-Filter	*Finite Impulse Response* nichtrekursives Filter, welches bei Anregung mit einem Impuls mit einer endlich langen Antwort reagiert
FPGA	*Field Programmable Gate Array* Integrierter Schaltkeris, in den Schaltwerke und Schaltnetze programmiert werden können
FRIDGE	*Fixed-point Programming Design Environment* Verfahren zur Bestimmung von Bitbreiten nach [2]
HYBRID	*Hybride Suchheuristik* Suchheuristik, welche die Heuristiken MIN+B und MAX-1 kombiniert
IA	*Intervall Arithmetik* Arithmetik für Intervalle
IDFT	*Inverse Discrete Fourier Transform* Inverse Transformation der DFT
IF	*Interface* Schnittstelle
IIR-Filter	*Infinite Impulse Response* rekursives Filter, welches bei Anregung mit einem Impuls mit einer unendlich langen Antwort reagiert
KLT-Tracker	*Kanade-Lucas-Tomasi-Tracker*

	Tracking basierend auf optischem Fluss benannt nach Entwicklern
LUT	*Lookup Table* kleine Speicher in einem FPGA zur Nachbildung von logischen Schaltungen
MAX-1	*Maximale Bitbreite minus eins* Suchheuristik, welche von der nach Vorgabe maximalen Bitbreitenkonfiguration ausgeht und schrittweise die Bitbreiten erniedrigt
MILP	*Mixed-Integer linear Programming Problem* Ein Problem der ganzzahligen linearen Optimierung
MIN+B	*Minimale Bitbreite plus B* Suchheuristik, welche von einer knapp nicht ausreichenden Bitbreitenkonfiguration ausgeht und schrittweise die Bitbreiten erhöht
MSB	*Most Significant Bit* Bit mit der höchsten Wertigkeit
NaN	*Not a Number* Anzeige einer Ausnahme bei Gleitkommazahlen, wenn der dargestellte Wert keine Zahl ist
NR-Verfahren	*Non-Restoring Verfahren* günstiges Verfahren zur Berechnung von Division und Wurzelfunktion
PESQ	*Perceptual Evaluation of Speech Quality* Maß zur Verständlichkeit der Sprache im Signal
RISC	*Reduced Instruction Set Computer* Designkonzept für einen Prozessor nach dem der Instruktionssatz auf simple Befehle beschränkt wird
SE-Klasse	*statistische Erhebungsklasse*

	generische Klasse im DANSE-Framework zur Erhebung statischer Daten
SIMD	*Single Instruction, Multiple Data* Rechnerarchitektur nach Flynnscher Klassifikation, welches pro Befehl mit einer Mehrzahl von Datenwörtern rechnet
SNR	*Signal-to-Noise Ratio* Signal-Rauschabstand, Störabstand. Verhältnis der mittleren Leistung des Nutzsignals zur mittleren Leistung des Rauschens
SQNR	*Signal-to-Quantization-Noise Ratio* Signal-Quantisierungsrauschabstand
SUIF	*Stanford University Intermediate Format* Austauschformat der Compiler Suite der Stanford Universität
VHDL	*Very High Speed Integrated Circuit Hardware Description Language* Beschreibungssprache für Hardware
WOLA	*Weighted Overlap Add* Verfahren zur schnellen Faltung, die die zyklische Faltung in eine aperiodische überführt

Formelzeichenverzeichnis

ΔX	absoluter Fehler	absolute Abweichung einer Zahl dargestellt durch eine fehlerbehaftete Zahlendarstellung zum Originalwert
x_n	Bit einer dargestellte Zahl	Wert des Bits an der Stelle n einer Festkommadarstellung
X	dargestellte Zahl	Zahlenwert einer dargestellten Zahl
E	Exponentenbitbreite	Anzahl der Bitstellen des Exponenten einer Gleitkommadarstellung
e_n	Exponentenbit	Wert des Bits an der Stelle n des Exponenten einer Gleitkommadarstellung
N_F	Fehlernutzung	Maß zur Nutzung des erlaubten Fehlers
N	Länge einer Zahlenfolge	Anzahl der Ziffern einer Zahlenfolge
M	Mantissenbitbreite	Anzahl der Bitstellen der Mantisse einer Gleitkommadarstellung
e_n	Mantissenbit	Wert des Bits an der Stelle n der Mantisse einer Gleitkommadarstellung
$\Delta_{max}X$	maximaler absoluter Fehler	maximale, absolute Abweichung einer Zahl dargestellt durch eine fehlerbehaftete Zahlendarstellung zum Originalwert
X_{max}	Maximalwert	Maximal darstellbarer Wert

$\mathbb{H}$	Menge von Implemetierungsvarianten einer mathematischen Funktionen	Menge der verfügbaren Implementierungsvarianten einer mathematischen Funktion
$\mathbb{E}$	Menge von Kanten	Menge von Kanten eines Graphs
$\mathbb{V}$	Menge von Knoten	Menge von Knoten eines Graphs
$\mathbb{M}$	Menge von mathematischen Funktionen	Menge von mathematischen Funktionen eines Algorithmus wie Division, Wurzelfunktion, Sinusfunktion, ...
$\mathbb{O}$	Menge von Operationen	Menge von Operationen wie Addition, Substraktion, ...
X_{min}	Minimalwert	Minimal darstellbarer Wert
μ	Mittelwert	arithmetischen Mittel einer Zahlenfolge
F	Nachkommastellenzahl	Anzahl der Bitstellen einer Festkommadarstellng nach dem Komma
$\Delta_{\%}X$	relativer Fehler	relative Abweichung einer Zahl dargestellt durch eine fehlerbehaftete Zahlendarstellung zum Originalwert
σ	Standardabweichung	Maß für die Streuung
X_{sub}	Subnormal Number	Nicht normalisierte Gleitkommadarstellung
σ^2	Varianz	Maß für die Streuung
I	Vorkommastellenzahl	Anzahl der Bitstellen einer Festkommadarstellung vor dem Komma
x_S	Vorzeichenbit	Wert des Vorzeichenbits einer Festkommazahl. S ist die Stelle des Vorzeichenbits

s	Vorzeichenbit	Wert des Vorzeichenbits einer Gleitkommazahl