Gleitkomma-Hochleistungsprozessor für rechenintensive Echtzeitanwendungen

D. Timmermann, B. Rix, B.J. Hosticka

Fraunhofer Institut für Mikroelektronische Schaltungen und Systeme (FhG-IMS)
Finkenstr. 61, ☎ (0203) 3783 219, FAX (0203) 3783 266
4100 Duisburg 1

Kurzfassung

Zur Lösung rechenintensiver Aufgaben, z.B. in der Signalverarbeitung, wird der arithmetische Prozessor IMSCOR24 vorgestellt, der herkömmliche Signalprozessoren und RISC Mikroprozessoren an Leistung weit übertrifft. Dies wird durch eine Konzentration auf das Zeitaufwendigste erreicht, nämlich die Berechnung auch komplizierter mathematischer Funktionen direkt in Hardware. Damit erschließen sich neue Anwendungsfelder u.a. in der Bildverarbeitung, digitalen Demodulation und Robotik.

Überblick

Sehr häufig ist der Systementwickler mit einem Dilemma konfrontiert: er kennt zwar die für die jeweilige Aufgabenstellung besten Lösungen (Algorithmen), kann diese aber nicht in der geforderten Geschwindigkeit mit Software auf Standardprozessoren implementieren. Denn diese können mit ihrem universell gehaltenen Befehlssatz einfach die benötigte Rechenleistung nicht liefern, sobald die verwendeten Algorithmen komplexere mathematische Funktionen beinhalten.
Der IMSCOR24 wurde am FhG-IMS für solche Aufgabenstellungen entwickelt, bei denen ein kontinuierlicher Datenstrom mit möglichst hohem Durchsatz bearbeitet werden muß. Anwendungsgebiete sind zeitkritische Probleme z.B. in der

- Bildverarbeitung, Mustererkennung
- Grafikbeschleuniger
- Echtzeitsimulation
- Robotik, Kinematik
- Signalverarbeitung, Datenkompression, Signal- und Spektralanalyse
- Telekommunikation.

Bild 1 zeigt das funktionale Blockdiagramm des IMSCOR24. Als Daten an den drei Eingangsports werden 24 Bit Festkommazahlen sowie Gleitkommazahlen einfacher Genauigkeit mit 24 Bit Mantisse und 8 Bit Exponent im IEEE-754 Standardformat verarbeitet. Auf diese Daten können, durch ein 6 Bit Instruktionswort gesteuert, 32

verschiedene Operationen mit einem Datendurchsatz von 10 Millionen Berechnungen pro Sekunde durchgeführt werden. Intern ist der Chip als Pipeline organisiert, so daß für die eingegebenen Daten nach 44 Taktzyklen das Ergebnis an den Ausgangsports bereitsteht. Mit jedem Taktzyklus werden neue Eingangsdaten eingegeben und berechnete Funktionen werden an den Ausgangsports ausgegeben. Die wichtigsten der zur Verfügung stehenden Operationen sind in Tabelle 1 aufgelistet. Außerdem lassen sich die in Tabelle 2 wiedergegebenen Funktionen berechnen.

Technische Einzelheiten

Wie dem Blockschaltbild zu entnehmen ist, werden die extern anliegenden Gleitkommazahlen zunächst in einer Vorstufe in ein spezielles Festkommaformat umgesetzt und ein Referenzexponent bestimmt. Dem schließt sich die Berechnung der eigentlichen Funktion in einer Festkomma-Pipeline nach dem CORDIC Verfahren /1/ an. Der Referenzexponent und das Instruktionswort werden in jeder Stufe mitgeführt. Als letzte Operation erfolgt eine Umwandlung des Ergebnisses in das externe Gleitkommaformat unter Verwendung des Referenzexponenten.

Die mathematische Basis des CORDIC Algorithmus bilden iterative Vektorrotationen in verschiedenen Koordinatensystemen. Da die Iterationen weitgehend regulär und sehr leicht pipelinebar sind, eignen sie sich besonders für eine Integration. Außerdem erzielt man mit dem CORDIC Verfahren eine sehr hohe Funktionalität, verglichen mit der aufzuwendenden Chipfläche, da das Koordinatensystem und damit die berechneten Funktionen besonders einfach geändert werden können.

Der Iterationsteil der Schaltung implementiert die drei in Bild 1 gezeigten Gleichungen, wobei m das Koordinatensystem festlegt, σ_i die Drehrichtung der Rotation, $S(m,i)$ die Shiftfolge, $\alpha_{m,i}$ den Teildrehwinkel und i den Iterationsindex. Der Parameter m wird zu 1, 0, oder -1 gewählt, wodurch die entsprechenden Vektorrotationen als Drehungen auf einem Kreis, einer Geraden oder einer Hyperbel interpretiert werden können.

Die interne Pipeline realisiert die CORDIC Gleichungen als festverdrahtete 32 Bit genaue Additions-und-Schiebeoperationen. Jede Stufe implementiert eine Iteration des Algorithmus.

Das Hardwareinterface ist sehr einfach gehalten. Außer zwei zueinander komplementären Taktsignalen, den Eingangsdaten und dem Instruktionswort sind keine weiteren Signale oder eine externe Beschaltung erforderlich. Durch den Verzicht auf ein Multiplexen der Ein- und Ausgänge lassen sich extrem leicht mehrere IMSCOR24 kaskadieren und damit auch komplizierteste Funktionen in Echtzeit realisieren, z.B. in der inversen Kinematik.

Anwendungsbeispiele

Was den IMSCOR24 für die genannten Aufgaben, verglichen mit anderen Signal- oder Mikroprozessoren, auszeichnet, ist die Vielzahl der bereitgestellten mathematischen Funktionen, die Möglichkeit mit jedem Taktzyklus neue Eingangsdaten zu bearbeiten und,

auf der Softwareseite, das Fehlen von Lesen-/Speichern- und Schleifenbefehlen. Die Palette der mit diesem Befehlssatz berechenbaren Funktionen erstreckt sich über die 4 Grundrechenarten über trigonometrische und hyperbolische Funktionen, Potenzrechnung und Logarithmen bis zu Vektoroperationen.

Durch den Verzicht auf Programmsteuerbefehle ergibt sich eine extrem einfache Programmierung, die auf eine eigene Assemblersprache verzichten kann. Jeder Satz von Eingangsdaten wird am Instruktionsport durch den gewünschten Befehl begleitet. Dabei kann dieser Befehl für alle Daten konstant gehalten werden oder auch variieren. Für den Anwender stellt der IMSCOR24 also einfach ein Element mit drei Eingangs-, einem Instruktions- und zwei Ausgangsports dar.

Einige Applikationsbeispiele sollen den Einsatz verdeutlichen. In einem Bildverarbeitungssystem sei auf die Daten im Bildschirmspeicher online eine Hough-Transformation zur Linienerkennung durchzuführen. Diese Transformation ist durch die Vorschrift $\rho = x \cos \pi N/511 + y \sin \pi N/511$ gekennzeichnet, die für jeden Bildpunkt mit der Adresse (x,y) für N = 0,1,..,511 zu berechnen ist. Die Funktionstabelle 1 zeigt, daß die komplette Operation mit dem Befehl ROTM durchgeführt werden kann, wobei die Adressen x,y der Bildpunkte direkt vom Bus des Bildverarbeitungssystems stammen und auf den x- bzw. y-Eingangsport gegeben werden. Am Ausgangsport x erscheint das Ergebnis ρ der Transformation. Bei einer Taktfrequenz von 10 MHz können so 10 Millionen Transformationen pro Sekunde durchgeführt werden.

Eine andere Anwendung ist die Echtzeitrealisierung einer vollständigen Koordinatenrotation gemäß der Vorschrift:

$$x = a \cos \phi - b \sin \phi$$
$$y = b \cos \phi + a \sin \phi$$

Diese Berechnung benötigt nur einen Befehl (ROT), wobei a, b und ϕ an den x,y und z Eingangsport angelegt werden und die Ergebnisse x und y am x- und yz-Ausgang erscheinen.

Ein Blick auf die Befehlsliste verdeutlicht auch die ideale Eignung für Umwandlungen vom polaren zum kartesischen Koordinatensystem und umgekehrt, womit u.a. Algorithmen der digitalen Demodulation direkt implementierbar sind.

Eine spezielle Betriebsart ist für die Beschleunigung von Matrixanwendungen vorgesehen, z.B. QR-Zerlegungen und Givens-Rotation. In diesen und vielen anderen Signalverarbeitungsproblemen beginnt man mit der Berechnung der Phase (arctan(y/x)) eines Vektors (x,y) und dreht weitere Vektoren (Matrixelemente) um diese Phase. Die speziellen I(nitialize)-Befehle benutzen daher den Winkel der vorhergehenden Phasenberechnung. Angenommen, es liege ein Datenstrom von Vektorkoordinaten (a_1,b_1), (a_2,b_2), (a_3,b_3), ... vor und die Vektoren (a_2,b_2), (a_3,b_3), ... sollen um $\arctan(b_1/a_1)$ gedreht werden. Dann lautet die Befehlssequenz: VECT, IROT, IROT, ...

Die Beispiele zeigen, daß mit dem IMSCOR24 erstmals Single-Chip Lösungen für derartige und kompliziertere Transformationen möglich wird.

Kundenspezifische Lösungen möglich

Um voll kundenspezifische Lösungen mit äußerst niedrigen Entwicklungszeiten zu ermöglichen, wurde für die interne Pipeline ein flexibler Modulgenerator entwickelt. Nach Eingabe der gewünschten Genauigkeit, der Anzahl Pipelinestufen und der wichtigsten CMOS-Technologieparameter wird ein sehr kompaktes, flächenoptimiertes Layout in der Zieltechnologie erstellt. Auf diese Weise kann sich der System- und ASIC-Entwickler die Funktionalität des IMSCOR24 für leistungsfähige Systemlösungen zunutze machen, z.B. im Zusammenspiel mit einem beliebigen Mikrokontroller oder anderen peripheren Funktionselementen.

Prinzipiell stehen im Modulgenerator zwei Architekturversionen zur Verfügung, die unterschiedliche Kosten/Durchsatz-Relationen aufweisen. In Bild 2a ist die Architekturvariante dargestellt, in welcher der IMSCOR24 gefertigt wurde. Mit zunehmender Genauigkeit (Wortbreite) sinkt die Durchsatzrate und steigen die Stückkosten infolge der größeren Chipfläche. Bei 24 Bit Genauigkeit (vgl. IMSCOR24) sind in einem 1,6 µm CMOS-Prozeß eine Taktfrequenz und damit Durchsatzrate von 10 MHz erreichbar. Die normierten Kosten betragen dann 1.

Alternativ kann die zweite Variante gewählt werden (Bild 2b), die zwar höhere Kosten aufweist, aber auch eine von der Genauigkeit unabhängige konstant hohe Durchsatzrate (Beispiel: 65 MHz in einem 1,6 µm CMOS-Prozeß). Bei kleineren Strukturgrößeren skalieren sich die Durchsatzraten und Kosten entsprechend.

Die Tabelle 2 gibt die wesentlichen technischen Daten wieder und Bild 3 zeigt das Chipfoto. Weitere Angaben können einem ausführlichen Datenblatt entnommen werden /2/.

Literatur

/1/ J.S. Walther, "A unified algorithm for elementary functions", Spring Joint Computer Conference (SJCC), 1971, S. 379-385

/2/ Data Sheet, "IMSCOR24: Floating Point Arithmetic Function Evaluator", FhG-IMS, Duisburg, 1992

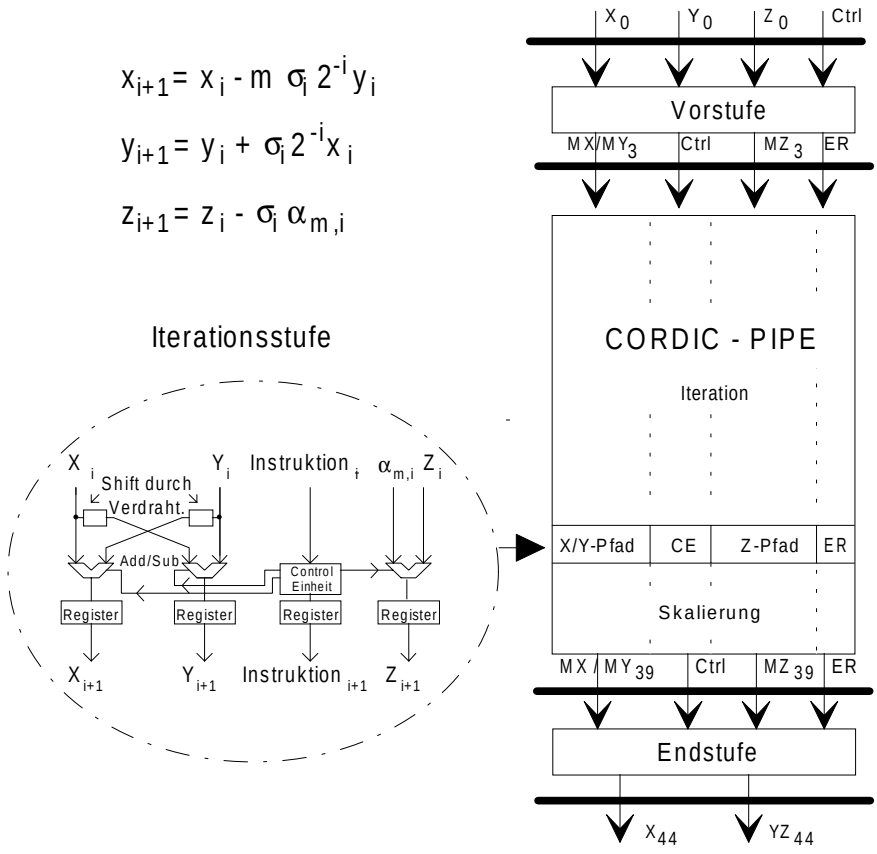

Bild 1: Blockschaltbild des IMSCOR24

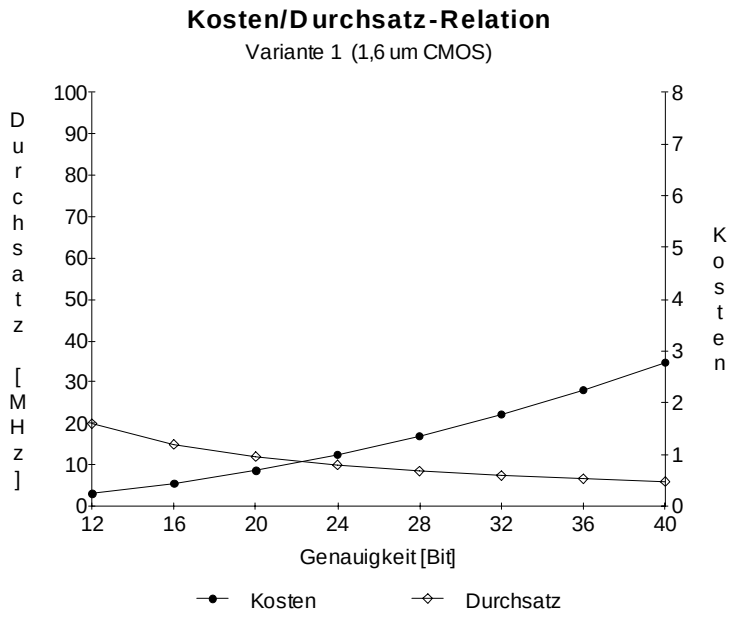

Bild 2a: Kosten/Durchsatz-Relation Version 1 (Kosten des IMSCOR24 = 1)

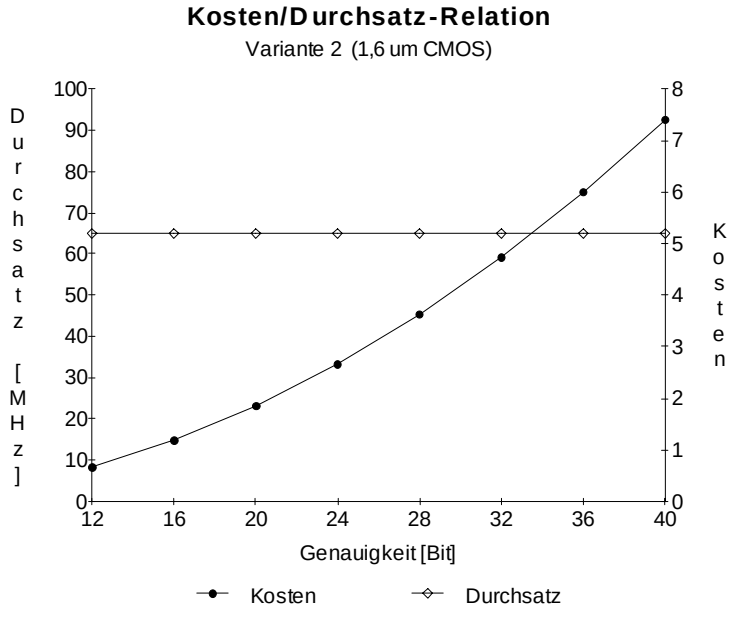

Bild 2b: Kosten/Durchsatz-Relation Version 2 (Kosten des IMSCOR24 = 1)

Bild 3: Chipfoto IMSCOR24

Instruktion	Ergebnis an Augang	
	x	**yz**
DIVADD	x	z+y/x
DIVSUB	x	z-y/x
MULADD	x	y+z*x
MULSUB	x	y-z*x
HVECT	$\sqrt{x^2 - y^2}$	z+artanh(y/x)
HVECTM	$\sqrt{x^2 - y^2}$	z-artanh(y/x)
HROT	x cosh z + y sinh z	y cosh z + x sinh z
HROTM	x cosh z - y sinh z	y cosh z - x sinh z
VECT	$\sqrt{x^2 + y^2}$	z+arctan(y/x)
VECTM	$\sqrt{x^2 + y^2}$	z-arctan(y/x)
ROT	x cos z - y sin z	y cos z + x sin z
ROTM	x cos z + y sin z	y cos z - x sin z
IHROT#	x cosh z' + y sinh z'	y cosh z' + x sinh z'
IHROTM#	x cosh z' - y sinh z'	y cosh z' - x sinh z'
IROT#	x cos z' - y sin z'	y cos z' + x sin z'
IROTM#	x cos z' + y sin z'	y cos z' - x sin z'
FX....	wie oben, Festkomma	

Anmerkung: Bei den mit # gekennzeichneten Instruktionen ist z' = arctan(y'/x'), wobei unter x' und y' die letzten Eingangsdaten vor dieser Instruktion zu verstehen sind. Damit lassen sich effizient Matrixlösungsverfahren (Givens Rotation u.ä.) implementieren.

Tabelle 1: Funktionstabelle

Instruktion	Eingabe			Ergebnis	
	x	y	z	x	yz
ROT	x	0	z	x cos(z)	x sin(z)
HROT	x	0	z	x cosh(z)	x sinh(z)
HVECT	x+1	x-1	0	$2\sqrt{x}$	½ ln(x)
HROT	x	x	z	x e^z	x e^z
HVECT	x	1	0	$\sqrt{x^2-1}$	arcoth(x)
HVECT	x+¼	x-¼	0	$\sqrt{x}$	ln(¼/x)
HVECTM	1	y	π/2	$\sqrt{y^2+1}$	arccot(y)
HVECT	x+y	y-x	0	$2\sqrt{x \cdot y}$	½ ln(y/x)

Tabelle 2: Zusätzlich zur Verfügung stehende Funktionen

Prozess	Double-metal CMOS
Design Regeln	1.6 µm
Transistoren	210 000
Gehäuseform	280 Pin PGA
Takt	10 MHz
Pipelinestufen	(3+37+4)= 44
Datenformate:	
extern	24b Mantisse, 8b Exponent (IEEE-754 einf. Genauigkeit) oder 24b Festkomma
intern	32b Mantisse, 10b Exponent
Instruktion	6 Bit

Tabelle 3: Technische Daten des IMSCOR24