Flächenreduzierte CORDIC-Architekturen

Steffen Dolling, Andreas Wassatsch, Dirk Timmermann

Universität Rostock

Fachbereich Elektrotechnik und Informationstechnik

Institut für Angewandte Mikroelektronik und Datentechnik

R.-Wagner-Str. 31, D-18119 Rostock

Tel.: 0381 498 3529, Fax.: 0381 498 3601

e-mail: dol@baltic.e-technik.uni-rostock.de

Kurzfassung

1. Motivation

Der CORDIC-Algorithmus wird in zahlreichen Anwendungen der Informationstechnik für die Berechnung elementarer und komplexer arithmetischer Funktionen genutzt. Dabei zeichnen sich CORDIC-Prozessoren im Vergleich zu Standard-Prozessoren bzw. zu konventionellen Hardwarelösungen durch eine hohe Leistungsfähigkeit sowie eine hohe Rechengeschwindigkeit aus. In diesem Zusammenhang ist eine hinsichtlich der Datendurchsatzrate optimale CORDIC-Implementierung eine Pipelinestruktur mit pfad- und bitparalleler Arithmetik. Allerdings ist diese Lösung mit einer hohen Hardwarekomplexität verbunden. Das Ziel dieser Arbeit besteht entsprechend in der Entwicklung von CORDIC-Architekturen mit einer möglichst stark reduzierten Chipfläche. Die Ausführungszeiten dieser Strukturen werden dabei durch die Minimierung nicht eingeschränkt. Die gezeigten Modifikationen auf Bitebene ergeben die bisher kleinste bekannte Realisierung einer CORDIC-Pipeline.

2. Redundante CORDIC-Strukturen

Die Iterationsgleichungen für den verallgemeinerten CORDIC-Algorithmus lauten entsprechend /1/:

$$x_{i+1} = x_i - m\sigma_i 2^{-S(m,i)} y_i \tag{1}$$

$$y_{i+1} = y_i + \sigma_i 2^{-S(m,i)} x_i \tag{2}$$

$$z_{i+1} = z_i - \sigma_i \alpha_{m,i} \qquad i = 0,1,\ldots, N-1 \tag{3}$$

Dabei bezeichnet m das Koordinatensystem, $S(m,i)$ die Shiftfolge, $\alpha_{m,i}$ den verallgemeinerten Drehwinkel, σ_i die Drehrichtung, n die Genauigkeit und N die Anzahl der Iterationen.
Die Realisierung eines CORDIC-Arrays unter Verwendung einer redundanten Zahlendarstellung auf der Basis von Carry-Save-Addierern /2,3/ oder Signed-Digit-Addierern /4,5,6/ erbringt den entscheidenden Vorteil einer sehr geringen und insbesondere von der Wortlänge unabhängigen Rechenzeit pro Addiererstufe. Allerdings resultieren daraus auch verschiedene Nachteile, wie z.B. ein erhöhter Aufwand in Form mehrerer Bits zu Darstellung eines Digits oder eine im Gegensatz zur nichtredundanten Darstellung komplexere Bestimmung des Vorzeichens und damit von σ_i , welches durch das höchstwertige von Null verschiedene Digit

festgelegt wird. Im worst-case-Fall müssen sämtliche Digits des Wortes vom MSD (Most-Significant Digit) bis zum LSD (Least-Significant Digit) untersucht werden. Eine praktikable Möglichkeit ist die Abschätzung des Vorzeichens aus einigen MSD's des entsprechenden Wortes. Ein dabei auftretender Fehler, der zu einer Verletzung der Konvergenzbedingung führen würde, kann durch Iterationswiederholungen kompensiert werden /2,3,4,5/. Die Iterationswiederholungen sind dabei abhängig von der Betriebsart, dem Koordinatensystem und der Anzahl t der untersuchten MSD-Stellen von z_i (Rotation) bzw. y_i (Vectoring). Eine weitere Möglichkeit ist die in /7/ vorgeschlagene Umformulierung der CORDIC-Iterationsgleichungen und die Berechnung von Absolutwerten, wodurch eine exakte σ_i – Bestimmung erreicht wird. Daraus resultiert eine reguläre VLSI-Struktur und die Vermeidung von Iterationswiederholungen, was aber eine nahezu verdoppelte Anzahl von Registern nach sich zieht.

3. Bisherige Ansätze zur Chipflächenreduktion

Ein Ansatzpunkt zur Hardwareminimierung ist die Art der Kompensation des Skalierungsfaktors k_m, die z.B. durch die Integration der Skalierung in den Algorithmus und die Optimierung der Skalierungs-Iterationen vorgenommen werden kann /8/.
Die angesprochene Problematik der Bestimmung der Drehrichtungen σ_i und eine damit verbundene Reduzierung der notwendigen Iterationswiederholungen ist der Gegenstand weiterer Arbeiten. In /4,5/ wird gezeigt, daß für Iterationen mit dem Index $i > n/2$ der Skalierungsfaktor keinen Einfluß mehr auf die Länge des zu drehenden Vektors hat und entsprechend $\sigma_i = 0$ zugelassen werden kann. In /6/ wird gezeigt, daß für $i > n/4$ für den Fall $\sigma_i = 0$ durch modifizierte Iterationsgleichungen $x_{i+1} = x_i \left(1 + m2^{-2i-1}\right)$ bzw. $y_{i+1} = y_i \left(1 + m2^{-2i-1}\right)$ Iterationswiederholungen vermieden werden können. Ebenfalls in /6/ wird eine Methode zur parallelen Vorausberechnung der drehrichtungsbestimmenden σ_i in der Betriebsart Rotation vorgestellt, was zu einer Einsparung von 2/3 Iterationsstufen im Z-Pfad führt. Dieses Verfahren kann entsprechend /9/ auf den Vectoring-Modus adaptiert werden. Die Anwendung der genannten Methode ist mit einem weiteren Vorteil verbunden. So lassen sich durch eine Umkodierung der σ_i (z.B. Booth-Recoding) zwei Iterationen im X- bzw. Y-Pfad in eine Stufe integrieren, wobei die Auswahl der auszuführenden Iteration über zwei unterschiedliche Schiebeoperationen erfolgt /6/. Dieses Konzept wurde in /3/ für eine gemischte Basis-2-4-Architektur auf die Betriebsart Vectoring erweitert.
In /9/ werden Architekturen gezeigt, die auf der Basis einer nichtredundanten Zahlendarstellung zu einer Minimierung der CORDIC-Datenpfade und damit der benötigten Chipfläche führen.
Diese Ausführungen werden in /10/ auf redundante Systeme übertragen und durch weitere Reduktionsmaßnahmen, die sich auf die kombinatorische Fläche beschränken, ergänzt. Zur Realisierung der Digitadditionen in den einzelnen X- bzw. Y-Pfaden kann ein mit 42 Transistoren günstig zu implementierender 4-2-Redundant-Redundant (RR) Adder /11/ eingesetzt werden. Dabei macht eine Betrachtung der Gleichungen (1-3) deutlich, daß mit jeder Iteration, mit Ausnahme der Wiederholungen, die Anzahl der Nullen in den MSD-Positionen, die zu den korrespondierenden Digits x_i bzw. y_i addiert werden, ansteigt. Entsprechend wurde in /10/ eine Struktur angegeben, die an den genannten Positionen eine vereinfachte 3-2-Redundant-Zero (RZ) Adderzelle verwendet. Bild 1 verdeutlicht diesen Sachverhalt, der zu einer Reduzierung der kombinatorischen Chipfläche führt, anhand von 3 Iterationen. Mit diesem Ansatz beläuft sich die Hardwareersparnis auf die ca. um den Faktor 2 kleineren 3-2 Zellen an den entsprechenden Digitpositionen. Diese Strukturen bilden den Ausgangspunkt für unsere weiteren Darlegungen.

4. Flächenreduktion

4.1. Betriebsart Rotation

Die genannte Architektur kann durch den Einsatz neu konzipierter Adderzellen an den Positionen, wo $x_{i,j}\, x_{i,j+1}\, x_{i,j+2}\, x_{i,j+3}\, x_{i,j+4} + 0000\, y_{i,0}$ gilt, weiter minimiert werden (analog im *Y*-Pfad). Das Prinzip ist in Abbildung 2 verdeutlicht.

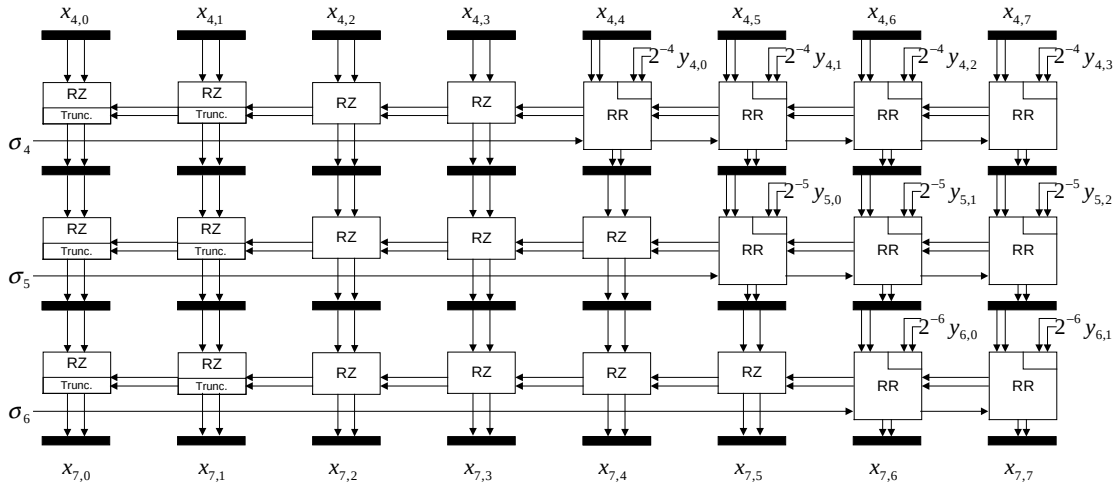

Bild 1: Reduzierter *X*-Datenpfad entsprechend /10/

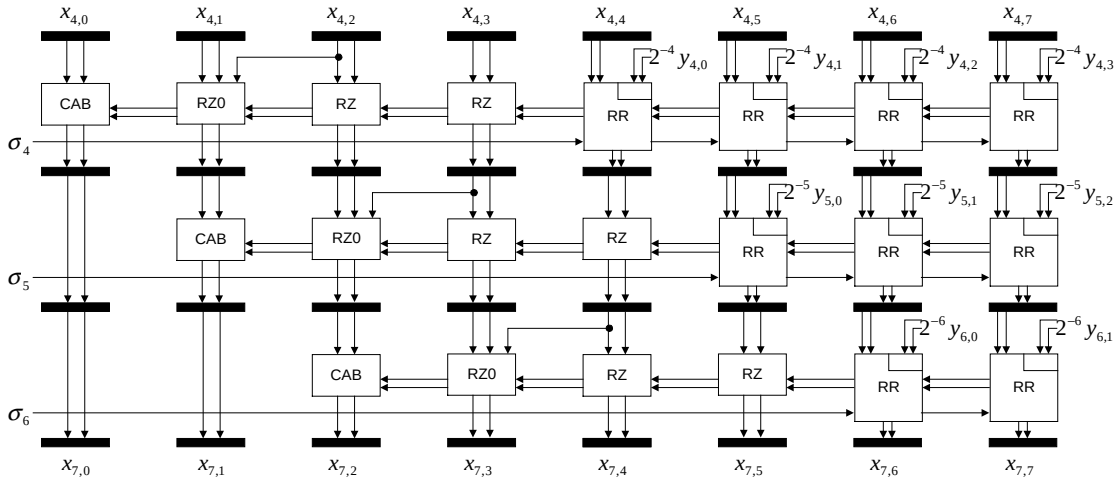

Bild 2: Weiter reduzierter *X*-Datenpfad mit speziellen Addiererzellen

Dabei wird durch die angegebenen RZ-Zellen, die Redundant-Zero_0 (RZ0) Zelle sowie die Carry-Absorber (CAB) Zelle eine Umkodierung des *X*- bzw. *Y*-Vektors an den entsprechenden Digitpositionen und eine Carry-Absorbierung durchgeführt. In diesem Zusammenhang sei erwähnt, daß wir für die interne Struktur dieser Zellen zwei Alternativen entwickelt haben, die sich hinsichtlich der Hardwareaufwands und der Latenzzeit unterscheiden. Die detaillierten Strukturen und die entsprechenden mathematischen Grundlagen sowie die daraus resultierenden Kodierungsvorschriften werden in der Langfassung dieses Artikels ausführlich beschrieben. Eine Besonderheit ist bei Iterationswiederholungen zu beachten, da hier *X* bzw. *Y* um den gleichen Wert 2^{-i} wie in der vorhergehenden Iteration verschoben werden und damit das beschriebene Prinzip nicht mehr anwendbar ist. In diesem Zusammenhang bestehen zwei Lösungsmöglichkeiten, die ebenfalls in der Langfassung beschrieben werden.
Die interne Struktur der speziellen Addiererzellen führt zu keiner Vergrößerung der Rechen-

zeit einer Iterationsstufe. Die Latenzzeit τ_I einer Addiererreihe ist damit gleich der einer RR-Zelle ($\tau_I = t_{RR}$). In den MSD-Positionen des *X*- bzw. *Y*-Pfades können zunehmend die Addiererzellen entfallen, womit gegenüber /10/ eine weitere Reduzierung der kombinatorischen Fläche erfolgt. Dies zeigt auch der in Bild 3 dargestellte Ausschnitt aus einem prinzipiellen Datenpfad-Layout für mehrere *X*- und *Y*-Iterationsstufen. Hier steigt in den als Ellipsen markierten Ausschnitten (Markierung nur im *X*-Pfad) die unbenutzte Chipfläche, die aus Gründen der Verdeutlichung hier bewußt freigehalten wurde. Auch die Shiftverdrahtung wurde aus demselben Grund nicht optimiert. In einem Standardzellen-Layout wirkt sich die reduzierte Chipfläche natürlich direkt aus.

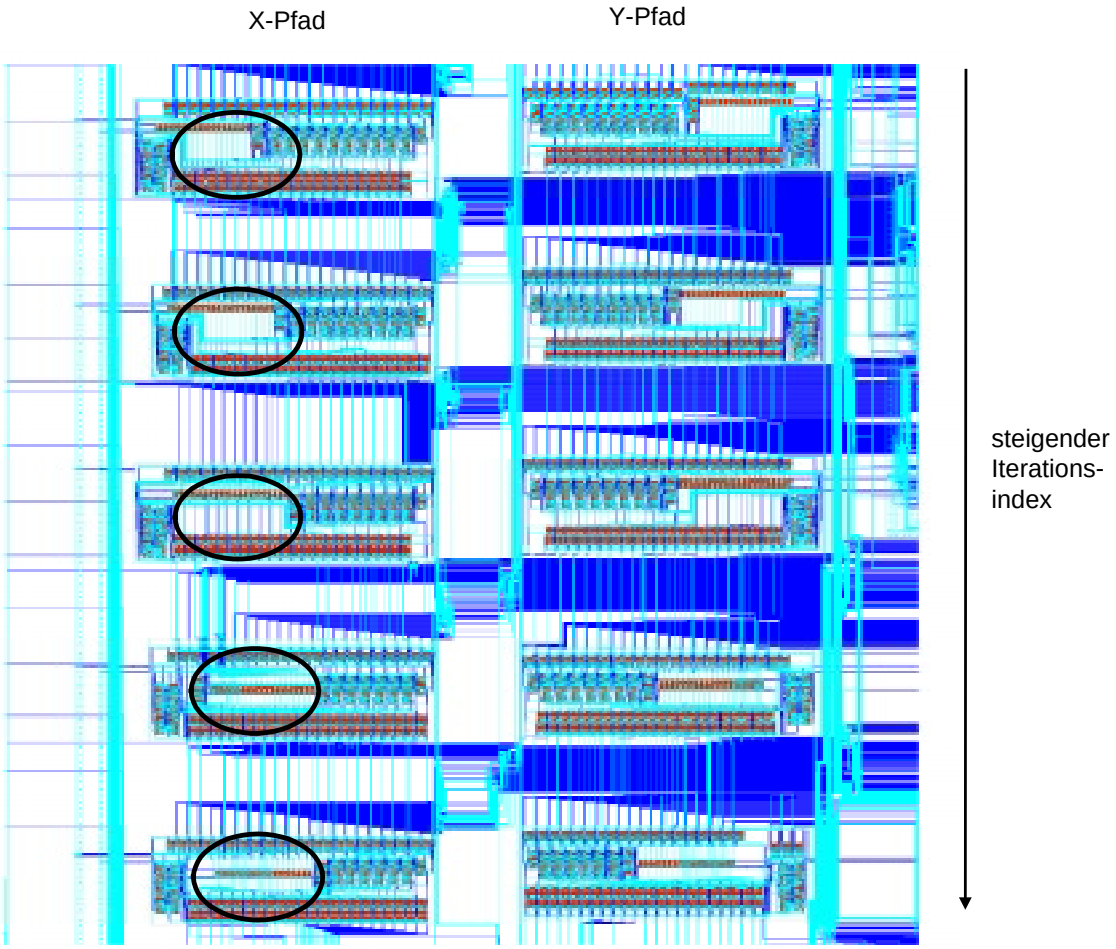

Bild 3: Ausschnitt aus dem Datenpfad-Layout für mehrere *X*- und *Y*-Iterationsstufen

Bevor weitere Merkmale der Architektur aufgeführt werden, soll kurz auf die Implementierung des *Z*-Pfades eingegangen werden. Dabei wird Gleichung (3) zu $z_{i+1} = 2\left(z_i - \sigma_i 2^{S(m,i)} \alpha_{m,i}\right)$ /5/ modifiziert. Die Ermittlung der drehrichtungsbestimmenden σ_i erfolgt damit in jeder Iteration an der gleichen Position über $t = 4$ führende Digit-Stellen. Entsprechend /10/ kann der *Z*-Pfad mit 3-2-Redundant-Binary (RB) Addierer/Subtrahierer Zellen und vom LSD her beginnend zunehmend reduzierten Addiererreihen ausgeführt werden. Ab der Iteration mit dem Index $i = (n + 1)/3$ kann die Berechnung in *Z* abgebrochen werden, wobei sich die restlichen σ_i aus dem zuletzt ermittelten z_i ergeben. Entsprechend /9/ sind die Iterationswiederholungen dann bis zu diesem Zeitpunkt auszuführen.
Die bisher beschriebenen neuen Architekturen ergeben eine erheblich geringere kombinatori-

sche Chipfläche als bei bisher bekannten Lösungen. Durch die erhebliche Reduzierung an nötigen Addiererzellen wird nunmehr die Chipfläche durch die Pipelineregister dominiert. In der Langfassung wird eine Architektur beschrieben, die auch hier zu einer verbesserten Architektur führt.
Damit ergibt sich in der Betriebsart Rotation eine Chipfläche gemäß Bild 4.

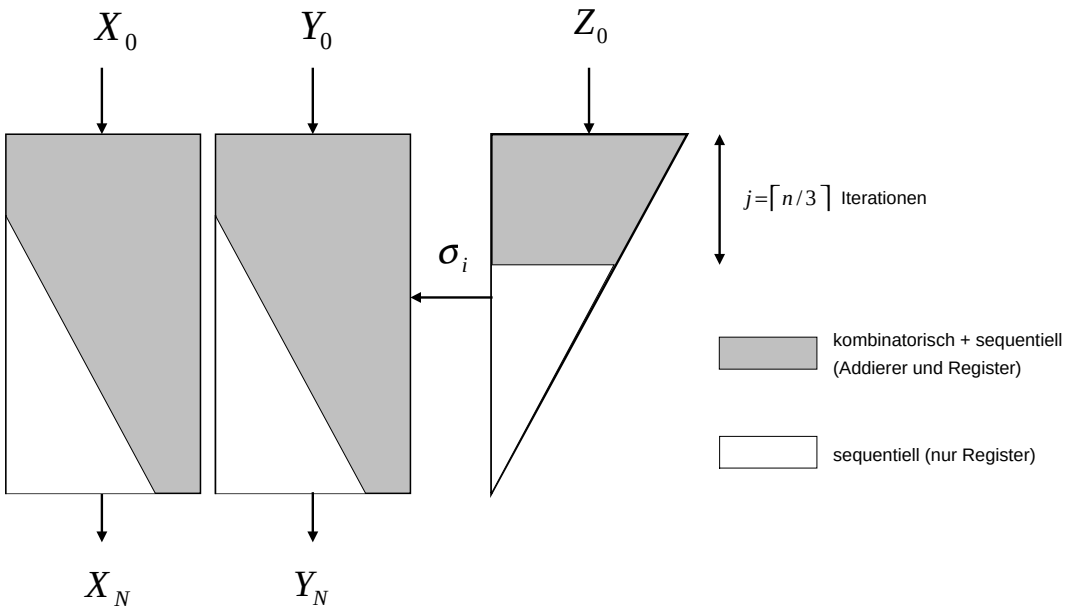

Bild 4: Chipfläche in der Betriebsart Rotation

4.2. Betriebsart Vectoring

Für den Einsatz redundanter Addierer werden die Gleichungen (1) und (2) zu $x_{i+1} = x_i - m\sigma_i 2^{-2S(m,i)} y_i$ und $y_{i+1} = 2\left(y_i + \sigma_i x_i\right)$ modifiziert. Entsprechend werden im *Z*-Pfad sich stetig vermindernde Drehwinkel $\alpha_{m,i}$ zu z_i addiert, womit die gleichen Bedingungen wie im *X*- bzw.*Y*-Pfad des Rotation-Modus gegeben sind. Damit kann unter Verwendung der speziellen Addiererzellen gemäß dem Prinzip in Abbildung 2 eine nahezu dreiecksförmige Architektur des *Z*-Pfades bzgl. der kombinatorischen Fläche gebildet werden. Dabei ist zu beachten, daß anstelle der 4-2-RR-Zellen die 3-2-RB Addierer/Subtrahierer Zellen eingesetzt werden. Ebenso ist das genannte Prinzip der Minimierung der Addiererreihen im *X*-Pfad anwendbar, wobei gegenüber der Betriebsart Rotation zwei Besonderheiten bestehen. Einerseits erhöht sich die Anzahl der Nullen, die in den MSD-Positionen addiert werden, mit jeder Iteration um zwei, womit sich ebenfalls die Anzahl der gesparten Addiererzellen erhöht. Andererseits können die Berechnungen im *X*-Pfad nach $n/2$ Iterationen abgebrochen werden, da der *X*-Wert dann *n*-Digit-Genauigkeit erreicht hat. Die Situation im *Y*-Pfad ist vergleichbar mit der im *Z*-Pfad des Rotations-Modes. Allerdings ist hier die Gesamtzahl der Iterationen auszuführen und als Addiererzellen sind die 4-2-RR-Adder zu implementieren. Somit ergibt hier eine zunehmende vom LSD her beginnende Reduzierung der Addiererreihen eine Dreiecksstruktur hinsichtlich der kombinatorischen als auch der sequentiellen Fläche. Abbildung 5 verdeutlicht die Aufteilung der Chipfläche.

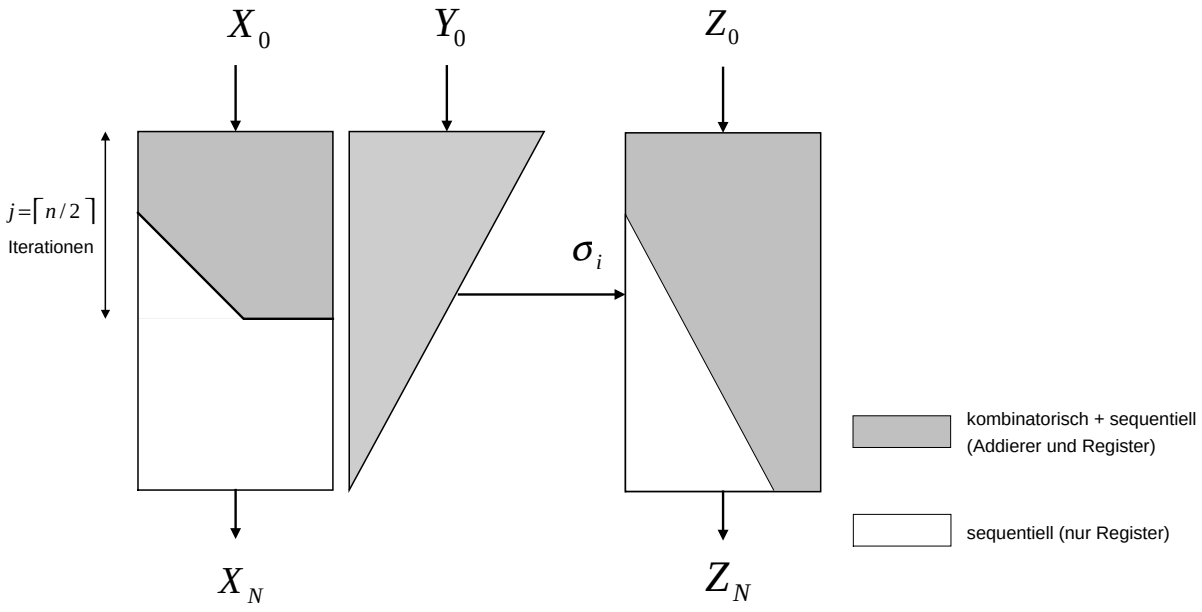

Bild 5: Chipfläche in der Betriebsart Vectoring

5. Zusammenfassung und Bewertung

Die vorliegende Arbeit beschäftigt sich mit einer Flächenreduzierung der CORDIC-Architektur in den Betriebsarten Rotation und Vectoring ohne Minderung der Durchsatzrate. Der wesentliche Punkt ist der Einsatz spezieller Addiererzellen, die zu einer inkrementellen Reduzierung der Addiererwortbreite in den einzelnen Iterationsstufen führt. Daraus resultieren Einsparungen sowohl hinsichtlich der kombinatorischen als auch sequentiellen Fläche. Zur Evaluierung wurden VHDL-Modelle für einen Standard-CORDIC und die vorgeschlagenen neuen Architekturen entwickelt, simuliert und auf eine Standardzellenbibliothek synthetisiert. Auf dieser Basis wurden Chip-Layouts für eine 1.0 µ CMOS-Technologie und Datenpfade mit einer Genauigkeit von extern 16 Bit realisiert. Durch die oben beschriebenen algorithmischen Modifikationen konnte eine Flächeneinsparung von 25% gegenüber einem Standard-CORDIC erreicht werden. Dabei ist zu bemerken, daß sich für größere Wortbreiten der prozentuale Anteil der Reduzierungen weiter erhöht. In der Langfassung werden die Flächenverhältnisse detaillierter aufgezeigt und bewertet.

/1/ J.S. Walther, “A unified algorithm for elementary functions“, Proc. of Spring Joint Computer Conference, pp 379-385, 1971

/2/ T. Noll, “Carry-save architectures for high speed signal processing“, Journal of VLSI Signal Processing, vol 3, pp. 121-140, 1991

/3/ E. Antelo, J.D. Brugera, and E.L. Zapata, “Unified mixed radix 2-4 redundant CORDIC processor“, IEEE Trans. on Computers, vol. 45, no. 9, pp. 1068-1073, Sept. 1996

/4/ N. Takagi, T. Asada, and S. Yajima, “Redundant CORDIC methods with a constant scale factor for sine and cosine computation“, IEEE Trans. on Computers, vol. 40, no. 9, pp. 989-995, Sept. 1991

/5/ J.-A. Lee and T. Lang, “Constant-factor redundant CORDIC for angle calculation and rotation“, IEEE Trans. on Computers, vol. 41, no. 8, pp. 1016-1025, Aug. 1992

/6/ D. Timmermann, H. Hahn, B.J. Hosticka, “Low latency time CORDIC algorithms“, IEEE Trans. on Computers, vol. 41, no. 8, pp. 1010-1015, Sept. 1992

/7/ H. Dawid, H. Meyr, “The differential CORDIC algorithms: constant scale factor redundant implementation without correcting iterations“, IEEE Trans. on Computers, vol 45, no. 3, pp 307-318, March 1996

/8/ G. Schmidt, et al., “Parameter optimization of the CORDIC-algorithm and implementation in a CMOS-chip“, Proc. EUSICO-86, Pt. 2, pp. 1219-1222, Hague, Netherlands, 1986

/9/ D. Timmermann, I. Sundsbø, “Area and latency efficient CORDIC architectures“, Proc. ISCAS’92, pp. 1093-1096, San Diego, May 1992

/10/ D. Timmermann, S. Dolling, “Unfolded Redundant CORDIC VLSI Architectures With Reduced Area and Power Consumption“, VLSI’97, Gramado, Brasilien, August 1997

/11/ S. Kuninobu, et.al., “Design of high speed MOS multiplier and divider using redundant binary representation“, Proc. 8th Symp. Computer Arithmetic, pp. 80-86, New York, 1987

Zuordnung des Vortrags:

1. Bauelemente der Mikroelektronik
2. Schaltungstechnik
3. Mikrosystemtechnik
4. Low Power
5.