Da das Verfahren wegen (3.5.16) quadratisch konvergiert, benötigt es weniger Iterationen als die in Abschnitt 3.4 gezeigten Divisionsalgorithmen. Auch bei Begrenzung der Zwischenwerte auf eine feste Bitzahl konvergiert das Verfahren ausreichend schnell zu dem gesuchten Wert.

Die vorgestellte Konvergenzmethode wurde von Chen [55] verallgemeinert. Mit der verallgemeinerten Konvergenzmethode können auch Funktionen wie $1/x$, $\sqrt{x}$, e^x und $\ln x$ berechnet werden. Aus der Literatur sind in großer Zahl alternative iterative Algorithmen bekannt. Als ein bedeutendes Beispiel wird das CORDIC–Verfahren in dem nachfolgenden Abschnitt im Detail behandelt. Das CORDIC–Verfahren hat einen sehr großen Funktionsumfang und kann in einer modularen Hardware realisiert werden. Eine besondere Eigenschaft ist, daß die Iterationszeit unabhängig vom Argument der Funktion ist.

3.5.2 Das CORDIC–Verfahren

Die Bezeichnung CORDIC steht für Coordinate Rotation Digital Computer. Das Basisverfahren zur Berechnung der Drehung eines Vektors in einem kartesischen Koordinatensystem und zur Berechnung von Betrag und Phase eines Vektors wurde von Volder [56] entwickelt. Das CORDIC–Verfahren wurde später für die Multiplikation, Division, und hyperbolische Funktionen erweitert. Eine Zusammenfassung der verschiedenen Funktionsberechnungen in eine vereinheitlichte Grundstruktur erfolgte von Walther [57].

CORDIC–Algorithmus

Die Drehung eines Vektors $[x_0, y_0]^T$ um den Winkel θ in einem kartesischen Koordinatensystem führt auf einen Vektor $[x_n, y_n]^T$ (Bild 3.5.2). Der Ergebnisvektor kann mit Hilfe einer Matrixberechnung ermittelt werden.

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{3.5.18}$$

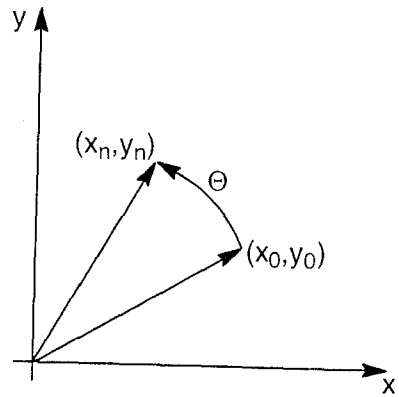

Bild 3.5.2: Vektordrehung im kartesischen Koordinatensystem

Unter Verwendung der Identität

$$\cos\theta = \frac{1}{\sqrt{1 + \tan^2\theta}} \tag{3.5.19}$$

kann durch Ausklammern von $\cos\theta$ die Beziehung (3.5.18) modifiziert werden

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \frac{1}{\sqrt{1 + \tan^2\theta}} \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{3.5.20}$$

Bei dem CORDIC–Verfahren wird die Drehung um den Winkel θ durch mehrere Rotationen mit bekanntem Teilwinkel α_i realisiert. Ähnlich wie jede ganze Zahl in einem endlichen Definitionsintervall durch n Ziffern dargestellt wird, kann jeder Winkel θ eines Definitionsintervalls mit einer gegebenen Genauigkeit durch einen Satz von n Teilwinkeln α_i dargestellt werden. Durch festzulegende Vorzeichen σ_i wird die Summe der Teilwinkel α_i den gegebenen Winkel θ annähern.

$$\theta = \sum_{i=0}^{n-1} \sigma_i \alpha_i \qquad \sigma_i \in \{-1, 1\} \tag{3.5.21}$$

Um n für eine vorgegebene Genauigkeit möglichst gering zu halten, wird der Betrag der α_i mit größerem Index abnehmen. Bild 3.5.3 zeigt ein Verfahren zur Bestimmung der Vorzeichen. Es werden die ersten α_i positiv bewertet bis die Summe den Wert θ überschreitet. Danach werden die α_i negativ bewertet bis die Summe den Wert θ unterschreitet. Dies wird dann entsprechend fortgesetzt. Das bedeutet, daß das Vorzeichen der Differenz zwischen θ und der Summe das Vorzeichen der Teilwinkel steuert.

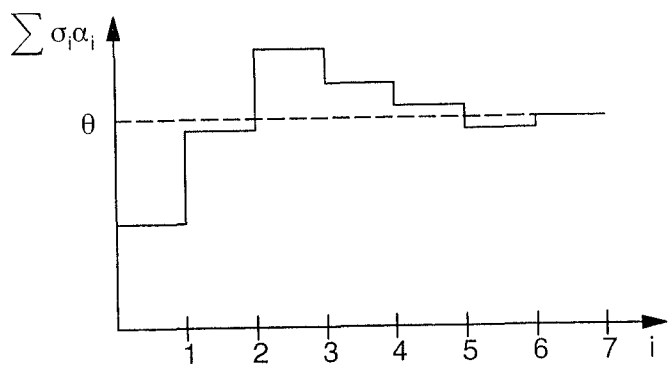

Bild 3.5.3: Iteration der α_i zur Darstellung von θ

Zur vereinfachten Berechnung des Matrixproduktes (3.5.20) werden die Winkel α_i so gewählt, daß der tan α_i eine Folge von Zweierpotenzen repräsentiert.

$$\tan \alpha_i = 2^{-i} \qquad i = 0, 1, \ldots n-1 \tag{3.5.22}$$

Ferner wird eine Hilfsvariable z_i eingeführt, die die aufakkumulierten Teilwinkel enthält und zur Steuerung des Vorzeichens der Teilwinkel benutzt werden kann. Mit $z_0 = \theta$ gilt dann

$$z_{i+1} = z_i - \sigma_i \arctan(2^{-i}) \tag{3.5.23}$$

$$\sigma_i = \begin{cases} +1 & z_i \geq 0 \\ -1 & z_i < 0 \end{cases} \tag{3.5.24}$$

Das Matrixprodukt vereinfacht sich für jeden Teilschritt zu

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = k_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^i & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3.5.25}$$

$$k_i = \frac{1}{\sqrt{1 + 2^{-2i}}} \tag{3.5.26}$$

Die Teilfaktoren k_i lassen sich für alle n Iterationen zu einem Gesamtfaktor zusammenfassen.

$$k = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1 + 2^{-2i}}} \tag{3.5.27}$$

Mit den bisherigen Vereinbarungen gilt für die Iterationsgleichungen

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i 2^{-i} y_i \\ y_{i+1} &= y_i + \sigma_i 2^{-i} x_i \\ z_{i+1} &= z_i - \sigma_i \arctan(2^{-i}) \end{aligned} \tag{3.5.28}$$

Zur Skalierung auf den korrekten Amplitudenwert ist noch eine abschließend Multiplikation durchzuführen.

$$\begin{aligned} k^{-1} x_n &\rightarrow x_n \\ k^{-1} y_n &\rightarrow y_n \end{aligned} \tag{3.5.29}$$

Der bisherige Iterationsablauf wird als "Rotation" bezeichnet, da ein Vekto $[x_0, y_0]^T$ um einen definierten Winkel θ gedreht wird. Wird der Ausgangsvektor der artig gedreht, daß der Vektor nach Abschluß der Iteration auf der positiven x–Achs liegt ($y = 0$), repräsentieren die aufakkumulierten Teilwinkel den Winkel arctan (y_0/x_0) und die x–Koordinate den Betrag des Vektors.

$$\begin{aligned} x_n &= \sqrt{x_0^2 + y_0^2} \\ y_n &= 0 \\ z_n &= z_0 + \arctan\left(\frac{y_0}{x_0}\right) \end{aligned} \tag{3.5.30}$$

Ähnlich wie vorher z_i durch gezielte Vorzeichensteuerung gegen Null konvergierte, muß dies nun für die y_i erfolgen. Für die Bestimmung der Vorzeichen σ_i gil nun

$$\sigma_i = \begin{cases} +1 & x_i y_i \geq 0 \\ -1 & x_i y_i < 0 \end{cases} \tag{3.5.31}$$

Da die Drehrichtung vom Quadranten abhängt, in dem sich der aktuelle Vekto befindet, muß das Vorzeichen des Produktes $x_i y_i$ zur Vorzeichensteuerung verwendet werden. Zur Vereinfachung der Realisierung wird das Vorzeichen des Produkte aus den Vorzeichen von x_i und y_i bestimmt. Die zuletzt dargestellte Betriebsart wir als "Vectoring" bezeichnet.

Mit dem bisher definierten Iterationsprozeß lassen sich trigonometrische Funktionen und Quadratwurzeln bestimmen. Durch die von Walther [57] formulierten Erweiterungen können zusätzliche Elementarfunktionen ermittelt werden. Die Norm R und die Phase Φ eines Vektors $[x_0, y_0]^T$ wird mit einem Parameter m für verschiedene Koordinatensysteme definiert.

$$R = \sqrt{x_0^2 + m y_0^2} \tag{3.5.32}$$

$$\Phi = \frac{1}{\sqrt{m}} \arctan\left(\sqrt{m}\frac{y_0}{x_0}\right) \tag{3.5.33}$$

bzw. als inverser Zusammenhang

$$\begin{aligned} x_0 &= R\cos(\sqrt{m}\,\Phi) \\ y_0 &= \frac{1}{\sqrt{m}} R \sin\left[\frac{\Phi}{\sqrt{m}}\right] \end{aligned} \tag{3.5.34}$$

wobei für m gilt

$$m \in \{\ 1,0,-1\} \tag{3.5.35}$$

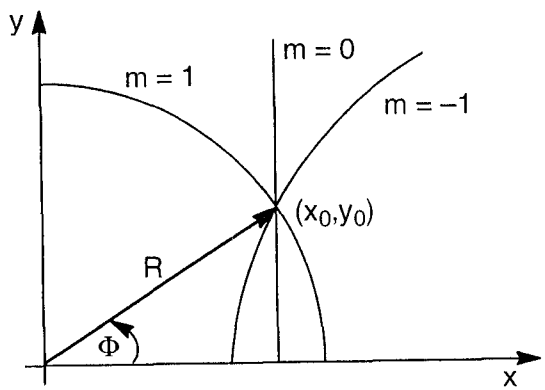

Bild 3.5.4: Kurven für R = const. in verschiedenen Koordinatensystemen

Die drei Koordinatensysteme werden nach den Kurven R = const. jeweils als hyperbolisch ($m = -1$), linear ($m = 0$) und zirkular ($m = 1$) bezeichnet (s. Bild 3.5.4). Analog zu den Gleichungen (3.5.28) ergeben sich somit für den verallgemeinerten CORDIC-Algorithmus:

$$\begin{aligned} x_{i+1} &= x_i - m\,\sigma_i \delta_{m,i}\, y_i \\ y_{i+1} &= y_i + \sigma_i \delta_{m,i}\, x_i \\ z_{i+1} &= z_i - \sigma_i\, \alpha_{m,i} \end{aligned} \tag{3.5.36}$$

Für die einzelnen Parameter gilt folgende Bezeichnung

m	Koordinatensystemparameter	$\{\ 1, 0, -1\}$
σ_i	Drehrichtung im i–ten Iterationsschritt	$\{-1, 1\}$
$\delta_{m,i}$	Schrittweite des i–ten Iterationsschrittes	$\{0 < \delta_{m,i} \le 1\}$
$\alpha_{m,i}$	Drehwinkel im i–ten Iterationsschritt	
i	Iterationsindex	$\{0,1, \ldots n-1\}$

Die sich für die verschiedenen Koordinationssysteme ergebenden Skalierungsfaktoren und Winkel der einzelnen Iterationsschritte (auch Mikrorotationen genannt) sind in Tabelle 3.5.1 zusammengefaßt.

Tabelle 3.5.1: Teilwinkel und Teilskalierungsfaktoren in Abhängigkeit von m

m	$\alpha_{m,i}$	$k_{m,i}^{-1}$
1	$\arctan \delta_{1,i}$	$\sqrt{1 + \delta_{1,i}^2}$
0	$\delta_{0,i}$	1
−1	$\operatorname{artanh} \delta_{-1,i}$	$\sqrt{1 - \delta_{-1,i}^2}$

Analog zu den Beziehungen (3.5.27) und (3.5.21) erhält man für den Skalierungsfaktor und den Gesamtwinkel nach n Iterationen

$$k_m = \prod_{i=0}^{n-1} k_{m,i} = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1 + m\delta_{m,i}^2}} \tag{3.5.37}$$

$$\alpha_m = \sum_{i=0}^{n-1} \sigma_i\, \alpha_{m,i} = \begin{cases} \sum_{i=0}^{n-1} \sigma_i \arctan \delta_{1,i} & m=1 \\ \sum_{i=0}^{n-1} \sigma_i\, \delta_{0,i} & m=0 \\ \sum_{i=0}^{n-1} \sigma_i\, \operatorname{artanh}\, \delta_{-1,i} & m=-1 \end{cases} \tag{3.5.38}$$

Für die Betriebsart "Rotation" ($z_n \to 0$) und "Vectoring" ($y_n \to 0$) gelten nach n Iterationen und einer abschließenden Skalierung mit $1/k_m$ somit die in

Tabelle 3.5.2 aufgelisteten Beziehungen. Aus der Tabelle kann entnommen werden, daß durch die Verallgemeinerung zusätzlich Multiplikation, Division und hyperbolische Funktionen möglich sind. Durch eine spezielle Wahl der Eingangsgrößen können aus den CORDIC–Funktionen aus Tabelle 3.5.2 Elementarfunktionen bestimmt werden. Mit $y_0 = 0$ werden im Rotationsmode $x_0 \cos z_0$, $x_0 \sin z_0$, $z_0 x_0$, $x_0 \cosh z_0$, $x_0 \sinh z_0$ ermittelt. Entsprechend werden mit $z_0 = 0$ im Vectoringmode $\arctan y_0/x_0$, y_0/x_0 und $\operatorname{artanh} y_0/x_0$ ermittelt. Weitere Funktionen können anhand der nachfolgend aufgelisteten speziellen Beziehungen erzeugt werden.

$$
\begin{aligned}
e^z &= \cosh z + \sinh z \\
e^{-z} &= \cosh z - \sinh z \\
\ln w &= 2 \operatorname{artanh} \frac{w-1}{w+1} \\
\sqrt{w} &= \sqrt{\left(w + \frac{1}{4}\right)^2 - \left(w - \frac{1}{4}\right)^2} \\
\tan w &= \frac{\sin w}{\cos w} \\
\tanh w &= \frac{\sinh w}{\cosh w}
\end{aligned}
\tag{3.5.39}
$$

Tabelle 3.5.2: CORDIC–Funktionen

Mode	m	CORDIC–Funktionen
$z_n \to 0$	1	$x_n = x_0 \cos z_0 - y_0 \sin z_0$ $y_n = y_0 \cos z_0 + x_0 \sin z_0$
	0	$x_n = x_0$ $y_n = y_0 + z_0 x_0$
	–1	$x_n = x_0 \cosh z_0 + y_0 \sinh z_0$ $y_n = y_0 \cosh z_0 + x_0 \sinh z_0$
$y_n \to 0$	1	$x_n = (x_0^2 + y_0^2)^{1/2}$ $z_n = z_0 + \arctan y_0/x_0$
	0	$x_n = x_0$ $z_n = z_0 + y_0/x_0$
	–1	$x_n = (x_0^2 - y_0^2)^{1/2}$ $z_n = z_0 + \operatorname{artanh} y_0/x_0$

Die Erzeugung von besonderen Elementarfunktionen erforder einigen Fällen auch die zweimalige Anwendung des CORDIC–Verfahrens. Beispielsweise

wird zur Erzeugung von $\tan w$ in einem ersten Schritt $\cos w$ und $\sin w$ erzeugt und dann in einem zweiten Schritt der Quotient gebildet. Die Funktionen wie cot, arcsin, arccos, coth, arsinh und arcosh erfordern die zweimalige Anwendung des CORDIC–Verfahrens.

Zur einfachen Realisierung des CORDIC–Verfahrens werden die jeweiligen Iterationsschrittweiten $\delta_{m,i}$ als Zweierpotenzen gewünscht.

$$\delta_{m,i} = 2^{-S(m,i)} \qquad i \in \{0, 1, \ldots n-1\} \tag{3.5.40}$$

Hierdurch wird die Matrixmultiplikation durch Shift und Addition implementiert. Die ganzzahlige Folge $S(m,i)$ bezeichnet man als Shiftfolge des CORDIC–Algorithmus.

Die Shiftfolge beeinflußt die Konvergenz, den Konvergenzbereich und den Skalierungsfaktor. Einerseits sollen die Teilwinkel $\alpha_{m,i}$ und die daraus resultierenden Teilsummen eine monoton fallende Folge bilden. Darüber hinaus muß die Shiftfolge so ausgelegt sein, daß die Erreichbarkeit des jeweiligen Iterationszieles mit einer gegebenen Genauigkeit garantiert ist. Die Winkel müssen somit die nachfolgende Beziehung erfüllen.

$$\alpha_{m,i} - \sum_{j=i+1}^{n-1} \alpha_{m,j} < \alpha_{m,n-1} \qquad i \in \{0, 1, \ldots n-2\} \tag{3.5.41}$$

Dies bedeutet, daß ein Teilwinkel der Iteration i im Falle einer nachfolgenden Richtungsumkehr durch alle folgenden Teilwinkel bis auf einen Restfehler kompensiert werden kann. Der Restfehler ist durch den letzten Teilwinkel der Folge gegeben. Die Erreichbarkeit ist beispielsweise immer dann gegeben, wenn die Teilwinkel ihrem Betrage nach nicht schneller als eine geometrische Reihe abnehmen. Für eine Reihe mit der Bedingung

$$\alpha_{i+1} > \frac{1}{2}\alpha_i \tag{3.5.42}$$

ist die geforderte Erreichbarkeit gegeben. Diese Bedingung wird für eine ganzzahlige Shiftfolge bei einem hyperbolischen Koordinatensystem ($m = -1$) nicht erfüllt. Deshalb sind in diesem Fall komplexere Shiftfolgen erforderlich. Bei der zirkularen Drehung ist die Bedingung erfüllt. Im linearen Fall ist das System am Rande des Konvergenzbereiches, da die Gleichheitsbeziehung gilt.

Die zu einer Shiftfolge zugehörige Summe aller Teilwinkel gibt den maximalen Winkel A_0 (Konvergenzbereich) an, für den das Iterationsziel erreichbar ist.

$$\max |A_0| = \sum_{i=0}^{n-1} \alpha_i + \alpha_{n-1} \tag{3.5.43}$$

In Tabelle 3.5.3 sind Shiftfolgen nach Walther [57] mit dem zugehörigen Konvergenzbereich und dem Skalierungsfaktor angegeben. Im hyperbolischen Fall werden zur Erfüllung der Konvergenzbedingung die Shifts { 4, 13, ... k, $3k+1$, ... } wiederholt. Da artanh (2^0) nicht definiert ist, beginnt die Shiftfolge für $m = -1$ bei 1. Mit den in Tabelle 3.5.3 aufgelisteten ganzzahligen Shiftfolgen ist die Anzahl der Zyklen für eine gegebene Genauigkeit vom Koordinatensystemparameter m abhängig. Beispielsweise erfordert eine Genauigkeit entsprechend 16 bit für $m = 1$ eine Shiftfolge mit 17 Zyklen, für $m = 0$ 16 Zyklen und für $m = -1$ 18 Zyklen.

Die Anwendung des CORDIC–Algorithmus mit den ganzzahligen Folgen aus Tabelle 3.5.3 erfordert im letzten Schritt eine Skalierung mit $1/k_m$. Diese Skalierung kann durch einen CORDIC mit $m = 0$ (Konstantenmultiplizierer) realisiert werden. Die Gesamtzyklenzahl einschließlich Skalierung erhöht sich somit erheblich. Für das genannte Beispiel von 16 bit Genauigkeit sind einschließlich Skalierung folglich 33 Zyklen für $m = 1$ und 34 Zyklen für $m = -1$ nötig. Die abschließende Skalierung kann auch mit einem Multiplizierer realisiert werden. Durch eine CSD–Codierung des Skalierungsfaktors kann die Anzahl der Addierer zur Realisierung des Multiplizierers verringert werden. Für die speziellen Koeffizienten einer 16 bit Realisierung sind 7 Addierer / Subtrahierer ausreichend.

Die in Tabelle 3.5.3 angegebenen Shiftfolgen haben einen begrenzten Konvergenzbereich. Mit Hilfe einiger Basisbeziehungen kann auch die Funktionsberechnung für größere Argumente erreicht werden. Im Falle der trigonometrischen Funktionen basiert eine Wertebereichserweiterung auf der Periodizität der Funktionen. Für hyperbolische Funktionen hat Walther [57] komplexere Beziehungen zur Argumentenreduktion zusammengestellt.

Tabelle 3.5.3: Beispiele positiver ganzzahliger Shiftfolgen mit zugehörigem Konvergenzbereich [57]

m	$S(m,i)$	max $\lvert A_0 \rvert$	k_m
1	0,1,2,3,4,5,...,i,...	1,743287	1,646760
0	1,2,3,4,5,6,...,i+1,...	1,000000	1,000000
–1	1,2,3,4,4,5,...12,13,13,14,...	1,118173	0,828159

Aus der Literatur sind verschiedene Verfahren zur Vereinfachung der Skalierungsfaktorkompensation bekannt. Bei den meisten dieser Vorschläge werden neben Standarditerationen mit Einzelshifts auch Iterationen mit Doppelshifts eingeführt. Der reziproke Skalierungsfaktor kann bei vielen dieser Vorschläge besonders einfach realisiert werden. Bei dem Vorschlag nach Deprettere [58] kann der reziproke Skalierungsfaktor durch einen Shift und eine Addition realisiert werden. In [59] sind

mehrere aus der Literatur bekannte Verfahren zur Skalierungsfaktorkompensation gegenübergestellt.

CORDIC–Architekturen

Der CORDIC–Algorithmus kann prinzipiell durch N in Kaskade betriebene CORDIC–Elemente (CE) realisiert werden. Jedes dieser CORDIC–Elemente führt entsprechend der Beziehung (3.5.36) eine Mikrorotation durch. Als steuernde Größe muß jedem Element der Mode–Parameter (Rotation oder Vectoring), der Koordinatensystemparameter m und die Iterationsebene i zugeführt werden. Bild 3.5.5 zeigt eine solche Anordnung. Eine effizientere Ausnutzung der Hardware wird durch eine rekursive Anordnung nach Bild 3.5.6 erzielt. Es werden die Anfangswerte x_0, y_0, z_0 in die Register geladen und dann mit jedem Iterationszyklus verändert. Eine rekursive Anordnung berücksichtigt auch die unterschiedliche Länge der Shiftfolgen. Die parallele Realisierung eines jeden CORDIC–Elementes für ganzzahlige Shiftfolgen zeigt Bild 3.5.7. Basiselemente sind umschaltbare Addierer / Subtrahierer und steuerbare Shifter. Das ROM im Kontrollpfad wird benötigt zur Bestimmung der zur Schrittweite $2^{-S(m,i)}$ zugehörigen Winkel $\alpha_{m,i}$. Aufgrund des geringen Wertebereichs für i und m sind nur wenige Werte im ROM zu speichern. Für die Kaskadenrealisierung kann das CORDIC–Element für jede Iterationsebene weiter vereinfacht werden, da der Parameter i fest vorgegeben ist.

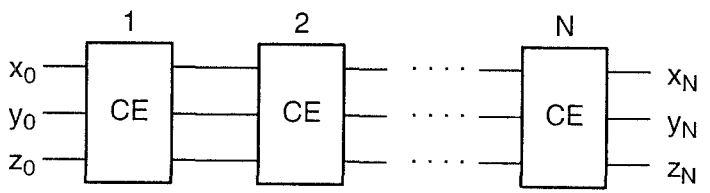

Bild 3.5.5: CORDIC–Realisierung durch eine Kaskade von CORDIC–Elementen (CE)

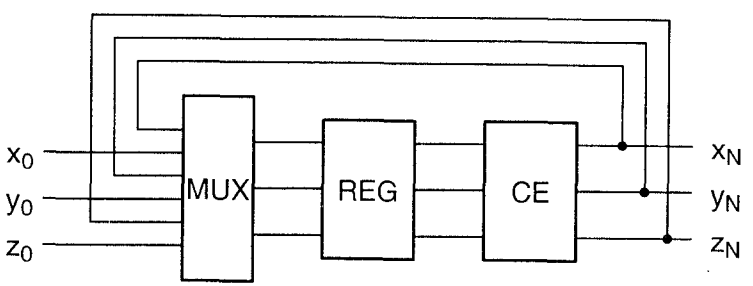

Bild 3.5.6: Rekursive Realisierung des CORDIC