Integrált CORDIC-bázisú jelprocesszorok és alkalmazásuk

PROF. DR.—ING. JOHANN BÖHME
Ruhr-Egyetem

Összefoglalás:

A már 1959 óta ismert CORDIC-eljárással iteratív módon kiszámítható többek között kétkomponensű vektorok adott szöggel való elforgatása, trigonometrikus és hiperbolikus függvények, hányadosok. Egy idetartozó processzor az integráltságot tekintve előnyös, egyöntetű struktúrával rendelkezik. A következőkben az olyan különböző funkciójú létező CORDIC-algoritmusok egységesítésére irányuló vizsgálatokról számolunk be, amelyek a realizációt tekintve CMOS-integrált pipeline CORDIC-processzorok. Továbbá rámutatunk arra, hogy bizonyos jelfeldolgozási feladatokra a CORDIC-chipek alkalmazása előnyös, a szokásos jelprocesszorokkal szemben. Példaként algoritmusokat ismertetünk diszkrét Fourier transzformációra, hullámdigitális szűrőre valamint beszédfeldolgozás és lineáris algebrabeli feladatokra, amelyek rotációkra vagy hiperbolikus rotációkra visszavezethetők.

PROF. DB. ING—J. F. BÖHME
1966-ban a Hannoveri Műszaki Egyetemen matematikusi diplomát szerzett. 1970-ben informatikai doktori címet kapott az Erlangeni Egyetemen. 1977 óta a digitális jelfeldolgozás magántanára. Először a Krupp Atlas Elektronik-nél dolgozott Brémában, közben az Erlangeni Egyetem Matematikai Gépek és Adatfeldolgozási Tanszék tudományos segédmunkatársa volt. 1970—74 között kutatási csoportvezető a Krupp Atlas Elektronik-nél. 1974—78 között kutatási csoportvezető az Informatikai Digitális Jelfeldolgozási Intézetnél a Bonni Egyetemen. 1978-tól 1980-ig az FGAN Nagyfrekvenciás Fizikai Intézet, Wachtberg-Werthofen tudományos munkatársa. 1980 óta a Ruhr Egyetem, Bochum Jelelmélet Tanszékén dolgozik.

1. Bevezetés

A digitális jelfeldolgozás feladatai gyakran nagy számítási teljesítmény rendelkezésre állását igénylik. Az olyan felhasználás mint a beszédfeldolgozás, csak a nagyintegráltság *(VLSI)* irányába való továbblépés által lehetségesek. Alapja mindennek az, hogy számos processzorelem *(PE)* egy chipen megvalósítható. A feladat lényege ilyenkor az, hogy algoritmusokat és ezekhez illeszkedő architektúrákat tervezzünk, hogy a szükséges adatáramlást a szűk chipfelületen elérjük. Megoldásokat kaphatunk egyszerű processzorelemek olyan architektúrájú felhasználásával, amelyek csak helyi kommunikációt igényelnek, továbbá megoldásokat kaphatunk nagymértékben párhuzamos algoritmusok és pipeline elvek alkalmazásával.

A kereskedelemben elérhető processzorok vagy mikroszámítógépek struktúrája komplexitásuk miatt többé nem alkalmazható. A jelfeldolgozás és a lineáris algebra számos feladatára azok a *PE*-k használatosak vagy ajánlottak, amelyek a mátrixszorzást különösen támogatják. Más feldolgozási feladatokhoz négyzetgyökvonás, osztás, trigonometrikus és hiperbolikus függvények, valamint ezek inverzei szükségesek, amelyek az ilyen *PE*-kkel kevésbé hatékonyan számíthatók. Példák erre: a diszkrét Fourier-transzformáció, a mátrixok faktorizációjára szolgáló Givens-eljárás és a normalizált létraszűrő. Mint alternatívákat az irodalomban (pl. Ahmed, 1980; Ahmed és szerzőtársai 1982) a Volder (1959) által bevezetett és Walter (1971) által általánosított *CORDIC*-algoritmusokat (*C*ordinate *R*otation *Di*gital *C*omputer) javasolták. Ez az algoritmus iterativan működik és az iterációs lépés előtt csak kevés összeadást és léptetést használ. Bizonyos jelfeldolgozási feladatokhoz, amelyek nagy adatáramlást igényelnek, a struktúrák *OORDIC*-alapú *PE*-hkel előnyösebb megoldásokhoz vezetnek, mint a szokásos *PE*-kkel, amelyek szoroznak és tárolnak a megoldás folyamán. Másrészt szinte minden szokásos jelfeldolgozási feladat számára *PE*-kkel viszonylag egyszerű megoldásokat létrehozni. *CORDIC*-algoritmusokat már régóta használtak a jelfeldolgozási feladatokkal kapcsolatban, pl. Schlüsser (1976) dolgozatában. Ebben a leírásban a *CORDIC*-algoritmus realizálására szolgáló pipeline-processzorokról van szó, amely Andrews és Eggerding (1978), valamint Deprettere és szerzőtársai gondolatain alapul. A következőkben leírjuk a *CORDIC*-eljárás változatait és egy iterativan működő *PE* struktúráját, amely összeadót és Barrel-léptetőt tartalmaz. Ezután tárgyaljuk a processzor elemekkel rendelkező pipeline-processzor struktúráját, mely csak előrehuzalozott léptetéseket alkalmaz. Beszámolunk a struktúra egységesítésére irányuló vizsgálatokról; azért hogy ezzel minden lehetséges *CORDIC*-függvényt számíthassunk. Példákat adunk, melyeknél *CORDIC*-processzorok alkalmazása célszerű.

Fordította: Csopaki Gyula
Elhangzott az 1987. máj. 6—7-én tartott VDE konferencián.

2. A CORDIC-algoritmus

A *CORDIC*-algoritmust Deprettere és szerzőtársaihoz (1984) hasonlóan iterativan definiáljuk.

$$\left.\begin{aligned} x_{i+1} &= x_i - m\sigma_i \delta_{mi} y_i \\ y_{i+1} &= \sigma_i \delta_{mi} x_i + y_i \end{aligned}\right\} \quad (1)$$

$$z_{i+1} = z_i + \sigma_i \varepsilon \alpha_{mi} \quad (2)$$

$$i = 0, \ldots, n_m - 1.$$

Legyen

$$\sqrt{m}\,\mathrm{tg}\,(\sqrt{m}\alpha_{mi}) = \delta_{mi} = 2^{-S_{mi}} + \eta_{mi} 2^{-S_{mi}} \quad (3)$$

Az m paraméter a kiszámítandó függvények fajtáját definiálja ($m=1$: trigonometrikus m = 0: lineáris m = −1: hiperbolikus).

A $\sigma_i \in \{-1, 1\}$ miatt az (1) formula — a

$$\cos(\sqrt{m}\alpha_{mi}) = (1 + \delta_{mi}^2)^{-1/2}\text{-val}$$

történő skálázást figyelembevéve — az $(X_i, Y_i)'$ vektorok α_{mi} „szöggel" való „rotációjának" fogható fel, ahol σ_i a forgásirányt adja meg. Kiindulva a z_0 kezdóértékből, a z_{i+1} a (2) képletben a szögeket összegezi, ahol $\varepsilon \in \{-1, 1\}$ valamely kiszámítandó függvény előjelét határozza meg. Az $S_{mi} < S'_{mi}$ egész számok nem negatívak és $\eta_{mi} \in \{-1, 0, 1\}$. Ezek meghatározzák az algoritmus konvergenciatartományát és a priori alkalmasan rögzítendők. Egy (1) és (2)-beli iterációs lépés csak összeadást és léptetést igényel, és ezt mikrorotációnak nevezik.

Az $(X_{n_m}, y_{n_m})'$ eredményeket a

$$K_m^{-1} = \prod_{i=0}^{n_m-1} (1 + m\delta_{mi})^{-1/2}$$

tényezővel skálázva,

az $(x, y)'$ vektort kapjuk, (4)

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos(\sqrt{m}\alpha_m) & -\sqrt{m}\sin(\sqrt{m}\alpha_m) \\ \dfrac{1}{\sqrt{m}}\sin(\sqrt{m}\alpha_m) & \cos(\sqrt{m}\alpha_m) \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \quad (5)$$

és

$$z = z_{n_m} = z_0 + \varepsilon\alpha_m, \quad \alpha_m = \sum_{i=0}^{n_m-1} \sigma_i \alpha_{mi} \quad (6)$$

n_m a mikrorotációk száma.

Mivel $(x^2 + my^2)^{1/2} = (x_0^2 + my_0^2)^{1/2}$, az (5)-t mint az (x_0, y_0) vektorok egy α_m szöggel való rotációját értelmezhetjük, amit makrorotációnak nevezünk.

A a_i forgásirányok mindig úgy vannak megválasztva, hogy vagy y_i, $y=0$-val ellentétesen, vagy z_i, $z=0$-val ellentétesen; növekvő i felé konvergál. Ha elérjük ezt az értéket, az 1. táblázatban megadott függvényeket egy makrorotációval kiszámíthatjuk. A kiválasztott kombináció m, ε, $y=0$ (vektor-mód) vagy $z=0$ (rotációs mód) által adott, amely a táblázat szerint kerül kiszámításra.

Walter (1971) és Yang (1986) szerint a σ_i a következőképpen választható: $y=0$ esetére $\sigma_i = -\mathrm{sgn}(x_i)\,\mathrm{sgn}(y_i)$ és $z=0$ esetére $\sigma_i = -\varepsilon\mathrm{sgn}(z_i)$. Így lehet iterálni a korrekt forgásirányokkal. Az S_{mi}, S'_{mi}, η_{mi} paramétereket ($m=-1, 0, 1$; $i=0,\ldots,n_m-1$), *CORDIC*-sornak nevezik, me-

1. táblázat

CORDIC-függvények

		m=−1	m=0	m=1
$z \to 0$	$\varepsilon=1$	$x=x_0\,\mathrm{cosh}z_0 - y_0\,\mathrm{sinh}z_0$ $y=y_0\,\mathrm{cosh}z_0 - x_0\,\mathrm{sinh}z_0$ $z=0$	$x=x_0$ $y=y_0 - z_0x_0$ $z=0$	$x=x_0\,\mathrm{cos}z_0 + y_0\,\mathrm{sin}z_0$ $y=y_0\,\mathrm{cos}z_0 - x_0\,\mathrm{sin}z_0$ $z=0$
	$\varepsilon=-1$	$x=x_0\,\mathrm{cosh}z_0 + y_0\,\mathrm{sinh}z_0$ $y=y_0\,\mathrm{cosh}z_0 + x_0\,\mathrm{sinh}z_0$ $z=0$	$x=x_0$ $y=y_0 + z_0x_0$ $z=0$	$x=x_0\,\mathrm{cos}z_0 - y_0\,\mathrm{sin}z_0$ $y=y_0\,\mathrm{cos}z_0 + x_0\,\mathrm{sin}z_0$ $z=0$
$y \to 0$	$\varepsilon=1$	$x=(x^2-y^2)^{1/2}$ $y=0$ $z=z_0 - \mathrm{artanh}(y_0/x_0)$	$x=x_0$ $y=0$ $z=z_0 - y_0/x_0$	$x=(x^2+y^2)^{1/2}$ $y=0$ $z=z_0 - \mathrm{arctan}(y_0/x_0)$
	$\varepsilon=-1$	$x=(x^2-y^2)^{1/2}$ $y=0$ $z=z_0 + \mathrm{artanh}(y_0/x_0)$	$x=x_0$ $y=0$ $z=z_0 + y_0/x_0$	$x=(x^2+y^2)^{1/2}$ $y=0$ $z=z_0 + \mathrm{arctan}(y_0/x_0)$

lyeket úgy kell megválasztani, hogy az $\alpha_{mi} \geq 0$ szög i-vel ne növekedjen és az

$$\alpha_{m1} - \sum_{j=i+1}^{n_m-1} \alpha_{mj} < m, n_m - 1 \quad (l = 0, \ldots, n_m - 2) \quad (7)$$

konvergenciafeltétel teljesüljön. Ezenkívül α_m, $n_m - 1$ elegendő kicsi kell legyen; azért, hogy pl. az x, y és z 16-bites fixpontos ábrázolásánál a 16-bites pontosság elérhető legyen. A konvergenciatartomány az összes szög halmazára a következőképpen írható le:

$$|\alpha| \leq \sum_{i=0}^{n_m-1} \alpha_{mi} + \alpha_m, \; n_{m-1} = c_m, \quad (8)$$

amely tehát egy α_m, $n_m -$ í hibáig, a (6) értelmében ábrázolható.

A C_m szám $m=1$ esetén nem kisebb π-nél, $m=0$ és lebegőpontos adatforma esetén nem kisebb, mint a számrendszer alapja; egyébként olyan nagy, amilyen csak lehet. Az utóbbi követelmény az α_{mi} más tulajdonságai miatt csak egy korlátozott mértékben teljesíthető. Walter (1971) vizsgálta a lehetőségeket az argumensredukálás által történő konvergenciatartomány-növelésre. Egy ilyen előfeldolgozás különösen a hiperbolikus függvények számára igényel speciális *PE*-ket.

Az $(x_{n_m}, y_{n_m})'$-nek a K_m^{-1}-gyel mint ismert felveendő (4)-beli faktorral való szükséges skálázásához; azért, hogy végül $(x, y)'$-t megkaphassuk, szorzásra lenne szükség. A következőkben úgy tekintjük, hogy K_m^{-1} a *CORDIC*-sor kiválasztása által elegendő pontosságú, így a következőképpen ábrázolható:

$$K_m^{-1} \approx 2^{-T_m} + \eta_{T_m} 2^{-T_m}, \quad (9)$$

Ezzel a skálázást szintén csak összeadásokkal és léptetésekkel végrehajtjuk, ahogyan az (1)-ben is. A (9) approximációnak olyan pontosnak kell lenni, hogy az abszolút hiba az x, y számábrázolás LSB-értéknél kisebb legyen. Egyébként az (5) ortogonális transzformáció normalizáló tulajdonsága elvész. Stabilitási megfontolások miatt a (9) jobb oldala nem lehet nagyobb, mint a (4)-é.

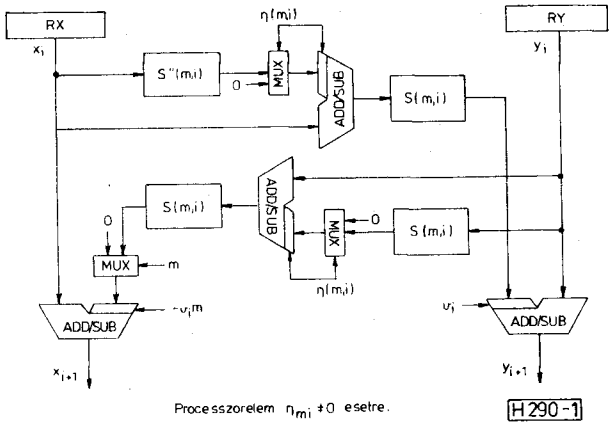
Kézenfekvőek azok a PE struktúrák, amelyekkel az (1) és (2) mikrorotációk $i=0,\ldots,\ n_m-1$ esetére, valamint a skálázás a (9) szerint végrehajthatók. Ezeket azonban itt nem ismertetjük, mivel a következő fejezetben egy hasonló architektúráról van szó, amely egy PE-pipeline számára készült. Más skálázási módokra az irodalomban számos javaslatot vizsgáltak meg, pl. Haviland és Tuzsinski (1980), Ahmed (1980) és Deprettere és szerzőtársai (1984) munkáiban. Ezekben bit-soros és bit-paralell aritmetikájú koncepciók vannak.

Lebegőpontos formára szintén ismeretesek megoldások, amelyekből néhányat mint bizonyos tudományos kalkulátorok perifériális processzorbeli chipjét már realizáltak.

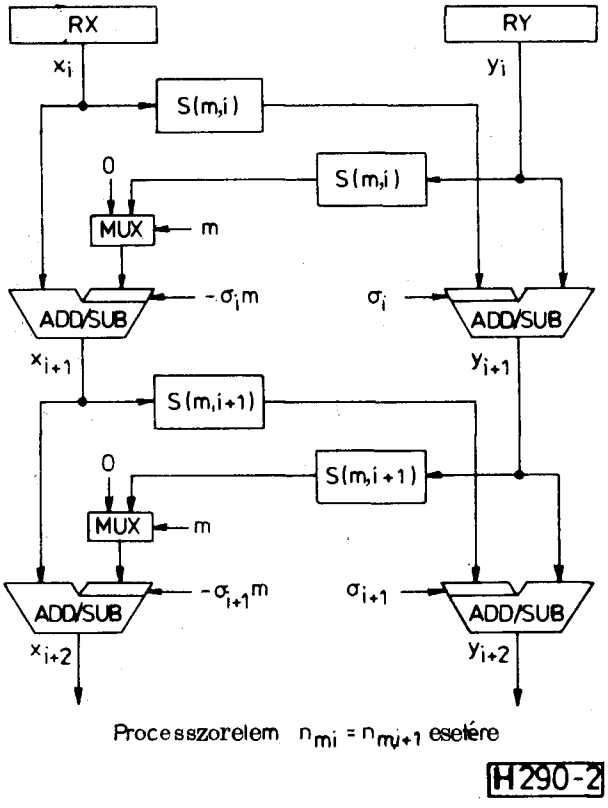
3. Pipeline-struktúrájú processzorok

Az (1)—(3)-beli $CORDIC$-algoritmus közel állahhoz, hogy a pipeline-elvet alkalmazni lehessen a mikrorotációk területén (Andrews és Eggerding 1978, Deprettere és szerzőtársai 1984). A bemenő és kimenő adatok ugyanis hasonlóan struktúráltak, az i-edik iterációs lépés számára a forgásszöginformáció a σ_i által adott, amely vagy az x_i és y_i előjeléből vagy az ε-ból és a z_i előjeléből adódik, továbbá az S_{mi}, S'_{mi} és η_{mi} paraméterek ismerteknek feltételezhetők. Egy rögzített m-re az n_m PE-k egy olyan sorozatát állítjuk elő, hogy az i-edik PE kimenetei az $i+1$-edik PE bemenőregisztereivel vannak összekötve, és minden PE egy mikrorotációt tud végrehajtani. Amikor az első adatok a „pipe"-ba beadásra kerülnek, akkor az első mikrorotáció az első PE-ben, a második a másodikban kerül kiszámításra stb. Ha egy PE a mikrorotációt befejezte és az eredmény a következő PE regiszterébe kerül, akkor a saját regiszterében új adatokkal egy egy új mikrorotáció kezdődik, ezáltal az S_{mi}, S'_{mi}, η_{mi} és ε, paraméterek megváltoztatása lehetséges. Ha ezek a paraméterek nem változnak, akkor a mindig egyforma feladatok az egymást követő adathármasok (x_0, y_0, z_0) sorozatában hajtódnak végre. Más esetekben a processzor kimenetét és bemenetét az n_m adatáramokra multiplexáljuk, minden PE-hez egy gyűrűs léptetőregisztert biztosítunk, amely a megfelelő $(S_{mi}, S'_{mi}, \eta_{mi})$-t előállítja és ezzel n_m-t a különböző feladatokhoz kiszámítja.

Az utolsó bekezdésben feltételeztük, hogy m nem változik feladatról feladatra vagy $n=n_m$ függetlenül m-től, mivel ezek a mikrorotációhoz szükséges PE-k darabszámai. A továbbiakban a processzorok számítási ideje minden PE számára meg kell hogy egyezzen. Az (1) számára a maximális számítási idő $\mu_{mi}\neq 0$-nál szükséges. Ebben az esetben a PE struktúrája fixpontos adatok számára az 1. ábra szerint célszerű (Deprettere et al,



1. ábra. Processzorelem $n_{mi}\neq 0$ esetre



2. ábra. Prooesszorelem $n_{mi}=n_{m,\,i+1}$ esetére

1984). Ekkor $S''_{mi}=S'_{mi}-S_{mi}$ és (3)-ban $2^{-S_{mi}}$ közös kiemelt tényező. Ha $\eta_{mi}=0$, akkor közelítőleg a fele számítási idő szükséges. Ha $\eta_{mi}=\eta_{mi+1}=0$, akkor két mikrorotáció hajtható végre egy PE-ben, ahogy a 2. ábrán látható.

A továbbiakban úgy tekintjük, hogy olyan $CORDIC$-sorokkal rendelkezünk, amelyek megengedik azt, hogy egy processzort a közvetlen közelében lévő n_d PE-vel az 1. ábra szerint és ehhez illeszkedve, $n-n_d$ esetén, $(n-n_d)/2$ PE-vei a 2. ábra szerint állítsunk össze. Végül egy PE-t kell még beiktatni, amely a skálázást a (9) szerint elvégzi. Az $\eta_{Tm}\neq 0$-val az 1. ábra struktúrájának

egyik változata alkalmazható. A feldolgozandó szögtartomány kiszélesítésére célszerű egy járulékos *PE*-t a processzor előfokozataként alkalmazni. Ha például $\pi/2 < C_1 < \pi$, akkor egy 90°-kal való elfordulást hoz a kívánt szélesítés. Ha egy *PE*-ben az adatok latens ideje T_{mic}, akkor a processzor számára $T_{mac} = [(n+n_d)/2+2]T_{mic}$. Az átbocsátóképesség, tehát a bemeneten két egymást követő adathármas közti idő $1/T_{mic}$.

A hardware-ráfordítás a pipeline-struktúrájú processzorokra nézve minimális, ha a léptetések behuzalozhatók a *PE*-kbe. Ez lehetséges, ha vagy m marad mindig változatlan, vagy olyan CORDIC-sorok találhatók, amelyeknek a léptetési paraméterei, $S_{mi}=S_i$ és $S_{mi}=S'_i$, nem függenek m-től. Schmidt és szerzőtársai (1986) egy olyan vizsgálatot írtak le, amelyben ilyen *CORDIC*-sorokat kerestek szisztematikusan. Ezenkívül azok voltak az optimalizálási célok, hogy a mikrorotáció céljára szolgáló *PE*-k $(n+n_d)/2$ számát minimalizálják úgy, hogy a pontosság- és konvergenciatartomány-követelményeket teljesítsék. A processzor be- és kimenő adatai számára a 16-bites fixpontos adatformánál és egy 24-bites *PE*-beli adatformánál a kísérletek például a 2. táblázatbeli *CORDIC*-sorokat adták.

Az 1. példa $n_d=6$ és $n=21$ esetére adja a megoldást.

A skálafaktorok $K_{-1}^{-1}=0{,}444^{-1}=2^1+2^{-2}$ és

$$K_1^{-1}=1{,}778^{-1}=2^{-1}+2^{-4}=$$
$$=2^{-2}K_{-1}^{-1}$$

2. táblázat

Példák univerzális CORDIC-sorokra

	No. 1					No. 2				
		n					n			
i	s	m=-1	m=0	m=1	s'	s	m=-1	m=0	m=1	s'
0	1	1	1	1	3	1	1	1	1	6
1	1	1	1	1	3	1	1	1	—1	11
2	1	1	1	0	5	2	1	1	0	3
3	2	1	1	1	3	3	1	1	1	4
4	2	1	1	—1	8	3	1	1	0	6
5	2	—1	0	—1	7	4				
6	3					5				
7	4					5				
8	5					6				
9	6					7				
10	7					8				
11	8					9				
12	9					9				
13	9					10				
14	10					11				
15	11					12				
16	12					13				
17	13					14				
18	14					15				
19	15					16				
20	16									
K_m		0,44	1,00	1,78			0,67	1,00	1,33	
C_m		3,21	2,91	2,67			2,00	1,88	1,65	

A 2. példa esetén $n_d=5$, $n=20$

$$K_{-1}^{-1}=0{,}667^{-1}=1+2^{-1}$$

ós

$$K_1^{-1}=2^{-1}K_{-1}^{-1}$$

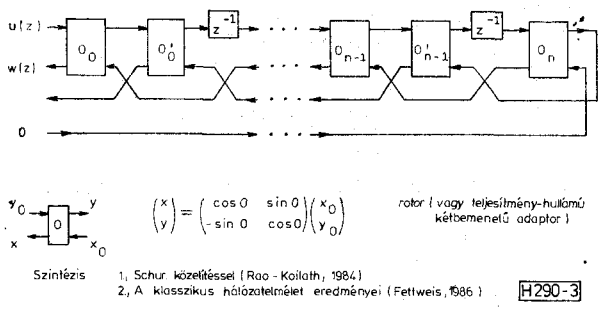
Egy adott lépésszámig, amely a kimenetek különféle elágazásával realizálható, a skálázó *PE*-k is egységesek. Figyelemre méltó, hogy az 1. példában az utolsó előtti *PE* $i=20$ esetén egy mikrorotációt hajt végre, aztán várakoznia kell.

Az idézett vizsgálatokban számos más előnyös *CORDIC*-sor található; külön előnyös, ha például nem kell hiperbolikus funkciókat számítani. Itt nincs arra lehetőség, hogy ezeket a megoldásokat bemutassuk. Yang (1986) megmutatta, hogy 32-bites lebegőpontos formátumú adatok számára megfelelő pipeline-struktúrájú processzort lehet építeni. Ennek a processzornak az előtagja a lebegőpontos számok denormalizálására is szolgál. A mikrorotációk a mantisszákkal a fixpontos számokhoz hasonlóan hajthatók végre, az exponens változtatása nélkül. Az utolsó lépésben következik a skálázás és normalizálás. A felül- és alulcsordulási bitek kielégítő számának, valamint egy alkalmas kerekítési algoritmusnak az alkalmazásával ezeknek a processzoroknak a numerikus pontossága csak lényegtelenül gyengébb, mint azoké a processzoroké, amelyek minden *PE*-ben lebegőpontos aritmetikát alkalmaznak. A hardware-többletfelhasználás egy fixpontos processzorhoz képest összehasonlítható szóhosszúságnál csak kb. 15%. A javasolt lebegőpontos processzorok alkalmazhatósága azonban sokkal nagyobb, mivel nem kell normalizált algoritmusokat használni.

4. Alkalmazások

Ahmed (1980) és Ahmed et al (1982) részleteiben vizsgálták, hogy a *CORDIC*-bázisú processzorok és a megfelelő algoritmusok felhasználása előnyös hardware-struktúrák kialakítását teszi lehetővé. Ezért ebben a fejezetben legfeljebb csak utalunk az ismert vagy nyilvánvaló alkalmazási lehetőségekre és csak egyes újabb eredményeket ismertetünk.

A szférikus trigonometriai formulák, mint amelyek pl. a navigációs feladatoknál a nagykörzetvezérlés esetén felmerülnek, csak trigonometrikus függvényeket tartalmaznak, amelyek *CORDIC*-processzorokkal $m=1$ esetén számíthatók. Már Voldernél (1959) található egy javaslat, amely korlátozás nélkül megvalósítható fixpontos processzorokkal. Mivel a jelfolyam a számítás során csak egy irányba halad, pipeline-struktúrájú processzorok előrehuzalozott léptetésekkel realizálhatók. Hasonlóan strukturált feladatok találhatók megfelelő megoldási lehetőségekkel a háromdimenziós vetítéseknél a számítógépes grafika területén, robotkarok vezérlésénél, ahol a manipulátorpozíció meghatározásához állandóan homogén transzformációkat kell végrehajtani. Egy másik alkalmazás a diszkrét Fourier-transzformáció. Az $\exp(-j2\pi kl/N)$-nel való szorzás elfordulásként értelmezhető ós *CORDIO*-processzorral számítható. Egy *DPT*-processzor lényegében *COR-*



3. ábra. Digitális szűrő kanonikus ábrázolása CORDIC-processzorokkal történő realizálásához

DIC-processzorokból és összeadókból áll (Despain 1979, Ahmed 1980, Lerch et al. 1986). Itt a struktúrák pipeline-szerkezetűek, amelyek a processzorok közötti globális kommunikációt — ahogy azok az *FFT* algoritmusokban fellépnek — kiküszöbölik; azért, hogy a megvalósítást lehetővé tegyék a nagyintegráltság irányába. Végül szükséges a lebegőpontos processzorok alkalmazása akkor is, ha a transzformálandó adatok nem normalizáltak.

Ha szeretnénk egy stabil, passzív racionális $w(z)=\boldsymbol{H}(z)u(z)$ digitális szűrőt kaszkádba kapcsolt egyforma processzorok formájában, amelyek csak a szomszédaikkal kommunikálnak és pipeline struktúrával realizálhatók, akkor ez Rav és Kailath (1984) nyomán a 3. ábra értelmében megvalósítható. A szétbontás a 3. ábra értelmében kanonikus, úgy, hogy a késleltetőelemek és az elforgató elemek száma minimális.

Ez a struktúra ortogonális szűrőproblémába történő beágyazás által és a Schur-rekurziók alkalmazásával állítható elő. Ahogy Fettweis (1986) megmutatta, a struktúrák hullámdigitális-szűrőként kétkapus adapterekből is kialakíthatók. amely egy transzformátorokból és cirkulátorokból álló analóg referenciaszűrő. Ezzel a szintézist, tehát a Θ_i és Θ'_i forgásszögek megállapítását, a hálózatelmélet klasszikus módszereivel is végrehajthatjuk. Az elfordulás realizálása *CORDIC*-processzorok ($m=1$) által nyilvánvaló, ahol szintén pipeline-struktúra alkalmazható. A szűrőstruktúra jó stabilitási tulajdonságai miatt normalizált bemenőjelekre fixpontos processzorok használhatók. A pipeline-struktúra teljesen kihasználható a be- és kimenőjel L-szeres multiplexálásával, ahol L a processzor *PE*-jeinek száma. Minden bemenőjel nyilvánvalóan csak $1/(3T_{mac})$ rátával adható be bitparalell módban, ahol T_{mac} a makrorotáció ideje. A bemenőjel áthaladási sebessége nagyságrendileg $1/T_{mic}$, ahol T_{mic} egy mikrorotáció ideje; ez olyan szűrőstruktúra számára érhető el, amelynek jelfolyama egyirányú. Példa erre a normalizált létraszűrő (Ahmed et al 1982). Ezek univerzális *CORDIC*-processzorokkal realizálhatók.

Utolsó példaként tekintsük egy $A\underline{x}=\underline{b}$ lineáris egyenletrendszer $\underline{x}$-re történő megoldását. Egy olyan eljárásra van szükség, amely nem igényel processzorok közötti globális kommunikációt. Egy ilyen eljárás használja pl. a Givens' rotációkat az A mátrix QR-szétosztására, ahol a Q ortogonális és R egy felsőháromszög mátrixnak a szorzata, amelyek mint alkalmasan leírt rotációk a Θ_{ri} forgásszöggel interpretálhatók, és amelyek A-ból meghatározhatók. A Θ_{ri} és az $R\underline{x}=\boldsymbol{Q}^{-1}\underline{b}=\underline{c}$ egyenletrendszer együtthatói a következő rekurzióval számíthatók:

$$i>r\text{-re}:\ \Theta_{ri}=\operatorname{arctg}\,(a_{ir}/a_{rr}) \tag{10}$$

$$a_{rr}=(a_{rr}^2+a_{ir})^{1/2} \tag{11}$$

$$j>r\text{-re}:\ \begin{pmatrix}a_{rj}\\ a_{ij}\end{pmatrix}=\begin{pmatrix}\cos\Theta_{ri} & \sin\Theta_{ri}\\ -\sin\Theta_{ri} & \cos\Theta_{ri}\end{pmatrix}\begin{pmatrix}a_{rj}\\ a_{ij}\end{pmatrix}, \tag{12}$$

$$\begin{pmatrix}b_r\\ b_i\end{pmatrix}=\begin{pmatrix}\cos\Theta_{ri} & \sin\Theta_{ri}\\ -\sin\Theta_{ri} & \cos\Theta_{ri}\end{pmatrix}\begin{pmatrix}b_r\\ b_i\end{pmatrix}. \tag{13}$$

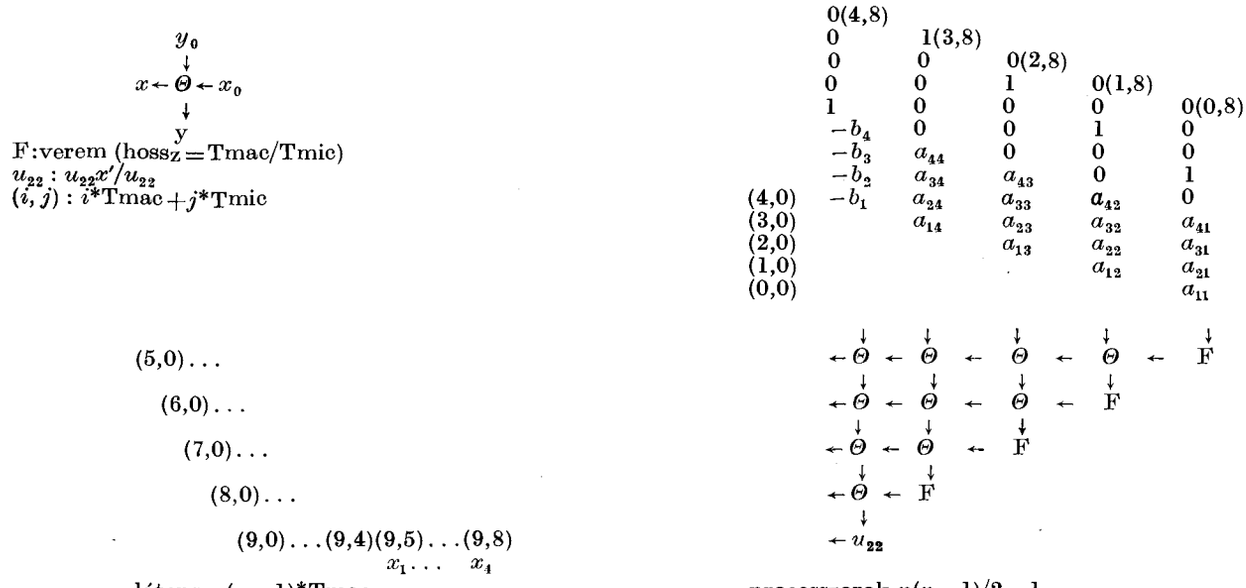
Ennek az algoritmusnak *CORDIO*-processzorokkal ($m=1$) szisztolés vagy hullámfront-tömbben való realizálásai ismeretesek. (Gentleman és Kung 1981, Ahmed et al. 1982). A *CORDIC*-processszorokat pipeline-struktúrával előnyösen alkalmazhatjuk, mivel a (10) számítása a σ_0, $a_1,\ldots a_{n-1}$ forgásirányok szekvenciális számítását jelenti, amely azonos sorrendben a (12) és (13)-ban van kidolgozva. Természetesen a mikrorotáció szintjén egy pipelining kapható (Deprettere et al 1984). Jainandunsig és Deprettere (1986) megmutatták, hogy az $\underline{x}=R^{-1}c$ Givens' rotációkra vonatkozó Faddeeva algoritmus egyik változatának visszahelyettesítése is $U_{22}\underline{x}$ és $U_{22}=\pm(1+x'x)^{1/2}$ formában oldható meg. A 4. ábrán végül egy további változat található, amely x-et szolgáltatja egy szisztolés tömb 4×4-es A mátrixának példáján. Θ-val jelöltük azokat a *CORDIC*-processzorokat amelyek a következőkben egy szöget számítanak. Amint az i-edik *PE*-ben σ_i kiszámításra kerül, σ_i tárolódik és a következő mikrociklusban a (12) vagy (13) rotációkkal kezdődik. A megfelelő kontroll a bemenőadatok egy járulékos bitjével könnyen elvégezhető. Az F jelölés az $L=T_{mac}/T_{mic}$ várakozási sor hosszát adja meg, hogy egy T_{mac} időkésleltetést létrehozzon. A leírt eljárás módosítható, hogy lineáris egyenletrendszer rekurzív legkisebb-négyzetes-megoldásait megkaphassuk; (McWirter 1983). Jainandunsig és Deprettere (1986 a és b) megoldásokat ad pozitív definit és különösen Töplitz-szerű mátrixokra.

5. Köszönetnyilvánítás

Bár még nincs a *CORDIO*-algoritmusokkal kapcsolatos összes probléma megoldva, különösen nincs a hiperbolikus függvények konvergenciaproblémája, ebben a cikkben bemutatjuk azokat a fejtegetéseket, melyek szerint számos alkalmazásban előnyös a *CORDIC*-processzorok alkalmazása. Északrajna-Wesztfália tartomány Tudományos és Kutatási Minisztériuma támogatja ezeket a vizsgálatokat és különösen azon integrált áramkörök fejlesztését, (Schmidt et al. 1986) amelyeket a duisburgi mikroelektronikai áramkörök és rendszerek Frauehhofer Intézetével közösen végeztünk. Szeretnék Zimmer és Hosticka kollégáimnak, valamint Hahn, Schmidt, Timmermann és Yang munkatársaimnak a sok értékes észrevételéért köszönetet mondani.

A $x=b$ lineáris egyenletrendszerek Givens rotációval történő megoldása

$$\begin{pmatrix} U_n & u_{12} \\ u'_{21} & u_{22} \end{pmatrix} \begin{pmatrix} A' & O_n & I_n \\ -b' & 1 & O'_n \end{pmatrix} = \begin{pmatrix} R & u_{12} & U_{11} \\ O'_n & u_{22} & u_{22}x' \end{pmatrix}$$



4. ábra. Lineáris egyenletrendszer CORDIC-processzorok szisztolós tömbbel való megoldása

IRODALOM

[1] *Ahmed, M. A. (1980):* Signal processing algorithms and architectures Ph. D. Thesis, Dept. of Electrical Eng., Stanford University

[2] *Ahmed, M. A., Delosme, J. M. and Morf, M.* (1982): Highly concurrent computing structures for matrix arithmetic and signal processing. IEEE Computer 15, 65—82.

[3] *Andrews M. and Eggerding, D. A.* (1978): A pipelined computer architecture for unified elementar functions. Computer and Electr. Eng. 5. 189—202.

[4] *Deprettere, E. F. A., Dewilde, P. M. and Udo, R.* (1984): Pipelinded CORDIC architectures for fast VLSI filtering and array processing. Proc. IEEE ICASSP San Diego, 41A. 6.1-4.

[5] *Despain, A. V.* (1979): Very Fast Fourier transform algorithms hardware for implementation. IEEE Trans. C-28, 333—341.

[6] *Fettweis, A.* (1986): Passive and lossless digital filter structures. Proc. And. Nordic Symp. VLSI in Computers and Communication, Linköping.

[7] *Gentleman, M. W. and Kung, H. T.* (1981): Matrix Triangulation by Systolic Arrays. Proc. S. P. I. E. 298 „Real Time Signal Processing IV", 19—26.

[8] *Haviland, G. L. and Tuszinski, A. A.* (1980): A CORDIC arithmetic processor chip. IEEE Trans. C-29, 68—79.

[9] *Hosticka, R. J. et al.* (1985): Power wave digital filters using CORDIC adaptors, AEÜ 39, 242—244.

[10] *Jainandunsig, K. and Deprettere, E. F. A.* (1986a): Design and VLSI Implementation of a concurrent solver for N-coupled least-squares fitting problems. IEEE J. SAC-4, 39—48.

[11] *Jainandunsig, K. and Deprettere, E. F. A.* (1986b): Solving sets of linear equations for real time signal processing. In *Signal Proclssing III: Theory and Applications*, I. T. Young *et al.* (eds.), North-Holland, Amsterdam, 1287—1290.

[12] *Lerch, R. et al.* (1986): CORDIC realization of a DFT processor. In *Signal Processing III, Theory and Applications.* I. T. Young *et al.* (eds.), North Holland, Amsterdam, 1319—1322.

[13] *McWhirter, J. G.* (1983): Recursive least-squares minimization using a systolic array. Proc. S. P. I. E. „Real Time Signal Processing VI".

[14] *Rao, S. K. and Kailath, T.* (1984): Orthogonal digital filters for VLSI implementation. IEEE Trans. CAS-31, 922—945.

[15] *Sehmidt, G.* (1984): Filterverfahren zur Spektralanalyse auf der Basis von CORDIC-Prozessoren. Diplomarbeit am Lehrstuhl für Signaltheorie der Ruhr-Universität Bochum.

[16] *Sehmidt, G. et al.* (1986): Parameter optimization of the CORDIC-algorithm and implementation in a CMOS-chip. In *Signal Processing III: Theory and Applications*, I. T. Young *et al.* (eds.), North Holland, Amsterdam, 1219—1222.

[17] *Schüßler, W. (Hsg.)* (1976): Ausgewählte Arbeiten über Nachrichtensysteme, Heft 22, Lehrstuhl für Nachrichtentechnik, Universität Erlangen.

[18] *Volder, E. J.* (1959): The CORDIC trigonometric computing technique.IRE Trans. EC-8, 330—334.

[19] *Walter, J. S.* (1971): A unified algorithm for elementary functions. Proc. Spring Joint Comp. Conf. 38, 379—388.

[20] *Yang, R.* (1986): Untersuchung zu einem 32-Bit--Gleitkomma CORDIC-Prozessor. Diplomarbeit am Lehrstuhl für Signaltheorie der Ruhr-Universität Bochum.