A MAD nagysebességű kombinációs 16 · 16+35 bites szorzó-összeadó alapcella és az erre épülő TMC2010MAC szorzó-akkumuláló integrált áramkör

SZŐKE SÁNDOR—DR. TUZSON TIBOR
Mikroelektronikai Vállalat

ÖSSZEFOGLALÁS

A cikkben bemutatott konstrukció jó példa egy új jellegű fejlesztési megközelítésre. A kiindulást a 2's komplemens kódú szorzás alapegyenlete adta, ebből alakítottunk ki egy hatékony algoritmust, amit végül átültettünk szilíciumra. Nem volt lehetőségünk külföldi referencia chip vizsgálatára, ami hátrányos mert nem tudtuk felhasználni a benne testet öltő ismeretanyagot, viszont előnyös mert így semmi sem kötötte meg a kezünket s lehetőségünk nyílt a mások által használtnál jobb kapcsolás kifejlesztésére. Eszközünkkel további két fontos irányba tettünk újabb lépést. Gyártás szempontjából a nagy bonyolultságú integrált áramkörök hazai előállításának irányába, alkalmazás szempontjából a digitális jelfeldolgozás szélesebbkörű elterjedése irányába.

Az áramkör tervezése 1986 augusztusában befejeződött, s a maszkok elkészülte után megindul a kísérleti sorozat gyártása a *MEV* fejlesztő gyártósorán.

SZŐKE SÁNDOB

1983-ban végzett a BME Villamosmérnöki Kar Műszaki Fizika Ágazatán. Ezt követően a MEV fejlesztő mérnökeként két éves nappali szakmérnökképzésen vett részt a BME Híradástechnikai Elektronika Intézetében. A digitális jelfeldolgozással és az ott felhasználható integrált áramkörök tervezésével foglalkozott, 1985-ben szerezte meg a szakmérnöki oklevelet. Jelenleg a MEV-ben folytatott fejlesztő munka mellett az egyetemi doktori cím megszerzésére készül.

DR. TUZSON TIBOR

1947-ben született Kolozsváron, 1970-ben végzett a Bukaresti Műszaki Egyetemen. Jelenleg a Mikroelektronikai Vállalat BOÁK tervező Osztályán csoportvezető. Fő érdeklődési területe a digitális jelfeldolgozás és eszközei, a témában több mérnöktovábbképző előadás és jegyzet szerzője és társszerzője, számos magyar és külföldi konferencián vett részt, mint előadó. 1985-ben egyetemi doktori címet szerzett.

1. Bevezetés

Az utóbbi időben erősen jelentkezik az a megközelítés, hogy matematikai, algoritmus, strukturális újításokkal, fejlesztéssel kompenzálják a technológiai lemaradásból adódó hátrányokat. Ez nem minden áramkörtípusnál valósítható meg, de jó eredményekkel kecsegtet például a modern számítástechnika, a párhuzamos feldolgozás és digitális jelfeldolgozás területén. Itt céltudatos fejlesztő munkával olyan megoldásokat találhatunk, hogy a végtermék paramétereiben megközelítheti esetleg lehagyhatja a jobb technológiai bázison készülő, de kevésbé kimunkált algoritmust használó áramköröket.

Jó példa erre a cikkben bemutatott *MAD* 16·16+35 bites szorzó-összeadó makrocella és a felhasználásával kiépített *TMC2010MAC* szorzó-akkumuláló eszköz, ahol egy új szorzó algoritmus konstruálásával nyertünk jó paraméterekkel rendelkező eszközt.

2. Szorzó-összeadó alapcella

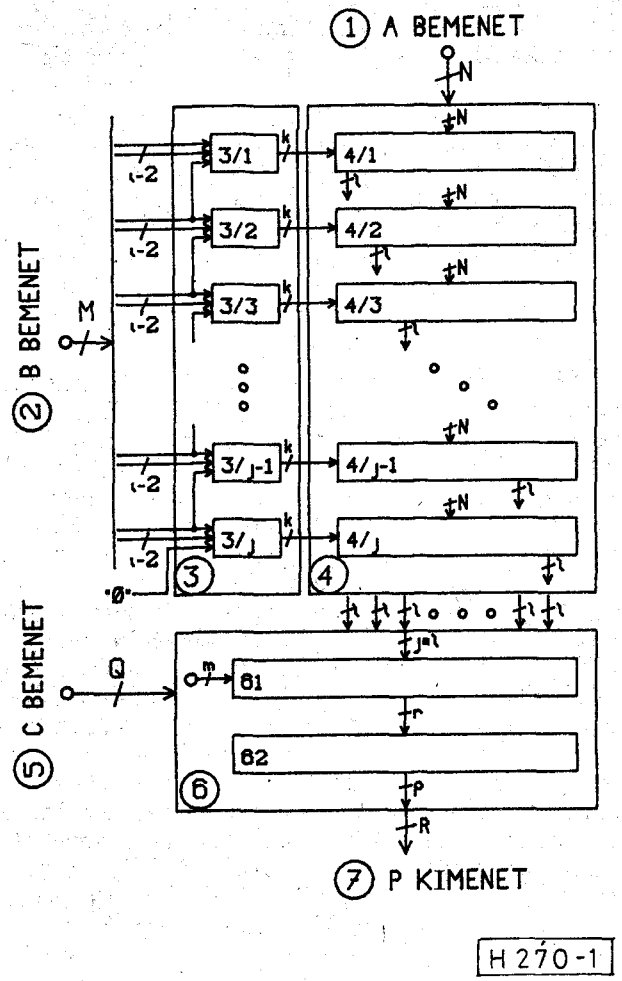
A digitális szorzás témakörében végzett alapos irodalomkutatás eredményeképp arra a megállapításra jutottunk, hogy az eddig bemutatott algoritmusok korántsem optimálisak, s a különféle megoldásokban előforduló ötletek kombinálásával és matematikai átalakításokkal lehetőség nyílik jobb algoritmusok konstruálására. Számos matematikai levezetés után jutottunk el a cikkben részletesen bemutatandó, tudomásunk szerint eddig még sehol sem publikált szorzó struktúrához, amely különféle szempontok szerint rokonságot mutat már jól ismert algoritmusokkal, de mindegyikkel szemben rendelkezik határozott előnyökkel. A részletes matematikai levezetés előtt nézzük meg a működést az általános blokkvázlat alapján *(1. ábra)*.

Beérkezett: 1986. XII. 8. ( )

2.1. *A cella blokkvázlata*

Általános esetben egy N bites szorzandót szorzunk meg egy M bites szorzóval, s a szorzathoz adunk egy Q hites összeadandót. Ha az eredményt $R > Max(Q, M+N-1)$ biten ábrázoljuk, nem fordulhat elő túlcsordulás a számolás során. A felhasznált algoritmus alapötelete, hogy az $a_i \cdot b_j$ bitszorzatok helyett a szorzó i—i bitjét úgy fogjuk össze j csoportba, hogy $j < M$, s a j db részszorzat bitjeinek száma kevesebb mint $M \cdot N$, ezáltal az összeadást végző egység kisebb komplexitású és gyorsabb működésű lehet. A szorzás elvégzése három fázisra bontható:

I. A vezérlő egység (3) j teljesen egyforma alegységből áll, amelyek mindegyike a szorzó bemeneten (2) megjelenő M bites szorzó i



1. ábra. Az $M \cdot N + Q$ bites szorzó-összeadó blokkvázlata

esetleg átlapoló bitje alapján k vezérlő jelet állít elő.

II. Az összeadandó biteket előállító egység (4) ugyancsak j egyforma alegységből áll, amelyek mindegyike a szorzandó bemeneten (1) lévő szorzandó N bitje és a megfelelő k vezérlő jel felhasználásával l pozitív összeadandó bitet állít elő. Mivel $j < M$, az összeadó egység (6) gyorsabb lehet, s $j \cdot l < M \cdot N$ így ezzel párhuzamosan komplexitása is csökken. A bitek ilyen előállítása lehetővé teszi, hogy az összeadandó bemeneten (5) lévő összeadandó Q bitjót kisebb csoportokra tördelve a szorzat számolásakor szimultán figyelembe vegyük, s ezáltal egy gyors működésű, az $A \cdot B + C$ összetett műveletet közvetlenül számoló áramkört nyerjünk.

III. A $j \cdot 1$ bitnyi részszorzat, a Q összeadandó bit és m fix értékű korrekciós bit az összeadó egység (6) bemenetére jut, ami két eltérő funkciójú és felépítésű alegységre osztható. A bitek gyors összeadását egy alapvetően párhuzamos elvű, de bit szinten szervezett váltott logikájú átvitelmegőrzős *(CS: Carry Save)* összeadó hálózat (61) végzi. A *CS* kódú eredményt egy átvitelgyorsítós összeadó (62) állítja vissza 2's komplemens kóddá. Az R bites eredmény az eredmény kimenetre (7) kerül.

2.2. Az algoritmus levezetése általános esetre

A blokkszintű működés vizsgálata után foglalkozzunk a legkritikusabb kérdéssel, a részszorzatok vagyis az összeadandó bithalmaz előállításával. A 2's komplemens kód törtszámos értelmezését használva a kiszámítandó szorzat értéke

$$P_v = A_v \cdot B_v = \left(-a_0 + \sum_{i=1}^{N-1} a_i 2^{-i}\right) \times$$

$$\times \left(-b_0 + \sum_{j=1}^{M-1} b_j 2^{-j}\right) \qquad (1)$$

Bontsuk fel a B szorzót $(i-1)$ bites csoportokra, a legkisebb helyiértékű bit után kiegészítve a szükséges számú nullával!

$$\begin{aligned} B_v = (&-b_0 + 2^{-1}b_1 + \ldots + 2^{-(i-2)}b_{i-2}) + \\ &+ 2^{-(i-1)}(b_{i-1} + 2^{-1}b_i + \ldots + 2^{-(i-2)}b_{2i-3}) + \ldots \\ &+ 2^{-q(i-1)}(b_{q(i-1)} + 2^{-1}b_{q(i-1)+1} + \ldots + \\ &+ 2^{-(i-2)}b_{q(i-1)+i-2}) + \ldots \end{aligned} \qquad (2)$$

Az M bites szorzó bitjei ily módon $j = int \times$ $\times [(M+i-2)/(i-1)]$ csoportra oszthatók, amelyek sok hasonlóságot mutatnak, sőt a teljes homogenitást csak egyetlen bit, az első zárójelben található előjegy negatív súlyozása bontja meg. Az első kivételével minden zárójelezett csoport első tagját bontsuk fel (3) azonosság felhasználásával, s a pozitív értékű tagot vigyük át az előző csoportba, míg a negatívot tartsuk meg az eredeti helyén!

$$b_k = 2 \cdot b_k - b_k$$

$$\begin{aligned} B_v = (&-b_0 + 2^{-1}b_1 + \ldots + 2^{-(i-2)}b_{i-2} + 2^{-(i-2)}) + \\ &+ \ldots + 2^{-q(i-1)}(-b_{q(i-1)} + 2^{-1}b_{q(i-1)+1}) + \ldots + \\ &+ 2^{(i-2)}(q(i-1)+i-2 \pm b_{q(i-1)+i-1}) + \ldots \end{aligned} \qquad (3)(4)$$

Az összeg felírható egyetlen szumma segítségével

$$B_v = \sum_{q=0}^{j-1} 2^{-q(i-1)} \times$$

$$\times (-b_{q(i-1)} + \ldots + 2^{-(i-2)}(b_{q(i-1)+i-2} +$$

$$+ b_{q(i-1)+i-1}) = \sum_{q=0}^{j-1} 2^{-q(i-1)} B_{qv} \qquad (5)$$

A szummában szereplő összeget B_q-val jelölve a szorzat az alábbi formában írható fel:

$$P_v = A_v \cdot \sum_{i=0}^{j-1} 2^{-q(i-1)} B_{qv} = \sum_{i=0}^{j-1} 2^{-q(i-1)} B_{qv} A_v \qquad (6)$$

A 2 egész hatványával való osztás megvalósítható a részszorzatok léptetésével, így a teljesen egyformára tervezhető $4/1 \ldots 4/j$ alegységek feladata, hogy B_q aktuális értékétől függően A-ból előállítsák X_q bitsorozatokat, amelyek a $B_{qv} \cdot A_v$ értéket képviselik valamilyen átkódolt formában.

X_q-k bitjeinek előállítására nem adható általános érvényű matematikai formula, a konkrétan választott i esetén meg kell határozni a lehető legegyszerűbb leképzést.
Általánosságban a következők mondhatók el.
A számoláshoz $j = int[(M+i-2)/(i-1)]$ rész-szorzatot kell összegeznünk, az egyes B_q-k értéke a szorzó i egymást egy biten átlapoló bitjének függvénye. B_q értékére fennáll, hogy $-1 \leqq B_{qv} \leqq 1$, a felbontás finomsága $\Delta B_{qv} = 2^{-(i-2)}$, így a B_q-k lehetséges értékeinek száma

$$\frac{1-(-1)}{2^{-(i-2)}}+1 = 2^{i-1}+1,$$

tehát ennyi féle részszorzatot kell előállítanunk A ból. A részszorzatok előállításánál előnyös, ha B_q értéke csak $B_{qv} = \pm 2^{-I} (I = 0, 1, 2 \ldots)$ lehet, de ez csak $i < 4$ esetekre teljesül.
Az már általánosabban is levezethető, hogy a 2's komplemens kódban értelmezett X_q-kból hogyan lehet előállítani a csak pozitív biteket tartalmazó bithalmazt. Belátható, hogy X_q-k hossza $L = N+i-2$ bit, tehát

$$X_{qv} = A_v \cdot B_{qv} = -x_0 + \sum_{k=1}^{L-1} x_{qk} 2^{-k} \qquad (7)$$

A túlcsordulások megelőzése érdekében az eredményt az $(M+N-1)$ bit helyett $(M+N-1+E)$ biten ábrázoljuk (E bittel növeltük a számábrázolási tartományt MSB irányba), így a szorzat értéke

$$P_v = -2^E p_{-E} + \sum_{i=1-E}^{M+N-2} 2^{-i} p_i \qquad (8)$$

A szorzat értékéhez hozzáadhatunk 2^{E+1}-t, hiszen ez nem okoz változást a figyelt tartományban.

$$2^{E+1} = \sum_{k=-E}^{(j-1)(i-1)} 2^{-k} + 2^{-(j-1)(i-1)} =$$

$$= \sum_{q=0}^{j-1} 2^{-q(i-1)} + \sum_{k=-E}^{-(i-1)} 2^{-k} +$$

$$+ \sum_{q=0}^{j-1} 2^{-q(i-1)} \sum_{n=1}^{i-2} 2^n + 2^{-(j-1)(j-1)} \qquad (9)$$

$$\overline{x_{qk}} = 1 - x_{qk} \qquad (10)$$

A bitkomplemens képzés (10) szabályát felhasználva a levezetés végeredménye:

$$P_v = \sum_{q=0}^{j-1} 2^{-q(i-1)} \left[\overline{x_{q\varnothing}} + \sum_{k=1}^{L-1} x_{qk} 2^{-k} + \sum_{n=1}^{i-2} 2^n \right] +$$

$$+ \sum_{k=-E}^{-(i-1)} 2^{-k} + 2^{-(j-1)(i-1)} \qquad (11)$$

A képlet kiszámolásához csak pozitív bitek összeadását és 2 hatványával való osztását, vagyis egyszerű jobbra léptetést kell megvalósítanunk. Pozitív számok osztásakor nullákat kell írnunk az MSB előtt elhagyott pozíciókba, amelyeket azonban felesleges ténylegesen összeadnunk, elhagyásukkal csökken az összeadó hálózat komplexitása. A képletben szereplő x_{qk} bitek általános esetben nem fejezhetők ki egyszerűen az A és B bitjeiből. Mivel univerzális elrendezési elv nem adható az összeadó hálózat szervezésére sem, célszerű megvizsgálni a megvalósításra kerülő konkrét esetet.

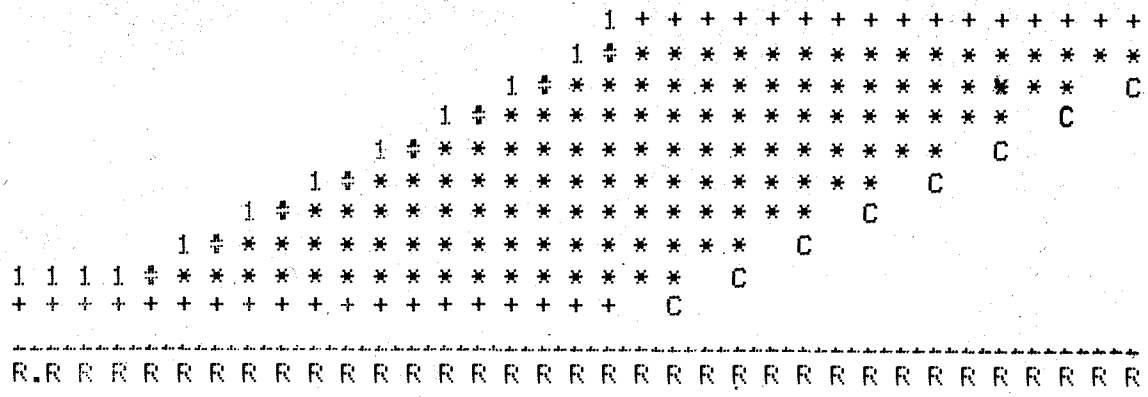
2.3. *Az algoritmus a megvalósított esetben*

A MAD makrocellában a szorzó ós a szorzandó egyaránt 16 bites, tehát $N = M = 16$. Egy részszorzat képzésénél a szorzó 3 bitjét vizsgáltuk, tehát $i = 3$, amiből $j = 8$ és $1 = 17$ adódik. Nem volt érdemes i értékét nagyobbra választani, mert a 8 részszorzat még viszonylag egyszerűen összeadható, s mivel teljesül az $i < 4$ feltétel, a bitsorok is aránylag egyszerűen képezhetők. Mivel B_{qv} 0, $\pm 1/2$, ± 1 lehet, a részszorzatok előállítása csak léptető ós komplementáló áramköröket igényel.
A 2. *ábra* mutatja (11) alapján, hogy hogyan állítandók elő egy részszorzatnak megfelelő bitek B_{qv} függvényében. Minden bit előállításához a szorzó 3 szomszédos bitje által meghatározott értékre és a szorzandó 2 szomszédos bitjére van szükség. Mivel B_{qv} meghatározása minden bitnél ugyanazt az áramköri kapcsolást igényelné, célszerű ezt kiemelni a bitelőállító cellákból, mert így minden rószszorzatnál csak egy ilyen egységre van szükség, s ezzel csökken az áramköri komplexitás. Ezt a kiemelt közös részt neveztük el vezérlő egységnek, amely úgy kódolja át a bemenő 3bitet, hogy azok felhasználásával a bitelőállító egységek a lehető legegyszerűbbek legyenek.
A 3. *ábrán* látható a MAD makrocella bitképe, ahol minden jel egy összeadandó pozitív bitet jelent a megfelelő pozíción. A részszorzatok bitjei — amelyek itt nem egyszerű bitszorzatok — *, a többihez képest negáltan előállítandó bitet #, míg az összeadandó bitjeit + jelöli. A $C = k$ a részszor-

X / B_{qv}	x_0	x_1	x_2	x_3	...	x_{14}	x_{15}	x_{16}
-1	a_0	$\overline{a_1}$	$\overline{a_2}$	$\overline{a_3}$		$\overline{a_{14}}$	$\overline{a_{15}}$	1
$-\frac{1}{2}$	a_0	$\overline{a_0}$	$\overline{a_1}$	$\overline{a_2}$		$\overline{a_{13}}$	$\overline{a_{14}}$	$\overline{a_{15}}$ 1
0	1	0	0	0		0	0	0 0
$\frac{1}{2}$	$\overline{a_0}$	a_0	a_1	a_2		a_{13}	a_{14}	a_{15} 0
1	$\overline{a_0}$	a_1	a_2	a_3		a_{14}	a_{15}	0 0

H 272—0

2. ábra. A q részszorzatból képzendő bitek B_{qv} függvényében



H 270 - 3

3. ábra. Az összeadandó bitek képe a *MAD* makrocellánál

zatok kivonását megvalósító komplementálásnál szükséges korrekciók, az 1-k fix egy értékű korrekciókat jelentenek. A bitképet összehasonlítva [7] algoritmusával jól látható, hogy mind a bitek, mind pedig a bitsorok száma kisebb a jelenlegi algoritmusnál. A kevesebb bitsor lehetővé teszi párhuzamos elvű összeadó hálózat alkalmazását, ami 17 sor esetén már túlságosan bonyolult huzalozást igényelne. Az áramkör igen vázlatos, csak az alapvető funkciókat elkülönítő blokkvázlata látható a 4. *ábrán.*

A szorzó 3 bites, egy biten átlapolódó csoportjait figyelve a 3/1 . . . 3/8 alegységek vezérlő jeleket állítanak elő, amelyek felhasználásával a 4/1 . . . 4/8 áramkörök a szorzandóból meghatározzák a 8 részszorzatnak megfelelő bithalmazt. A bitek először az átvitelmegőrzős összadó hálózatra kerülnek, amely a párhuzamos szervezésnek köszönhetően $5T_s$ alatt kiszámolja a *CS* kódban lévő végeredményt (T_s az egy bites teljes összeadó cella számolási ideje). A *CS* kódot 2's komplemens kóddá visszalakító átvitelgyorsítós összeadó 3 kisebb részre osztható, mivel egy-egy 6 bites végeredmény-rész már korábban előáll, így a sebesség szempontjából kritikus utolsó egységnek (62/3) 21 bites összeadást kell elvégeznie. A blokkvázlat alapján számítógépes szimulációval becsülhető a cella sebessége:

$$T_{MAD} = T_3 + T_4 + T_{61} + T_{62} \qquad (12)$$

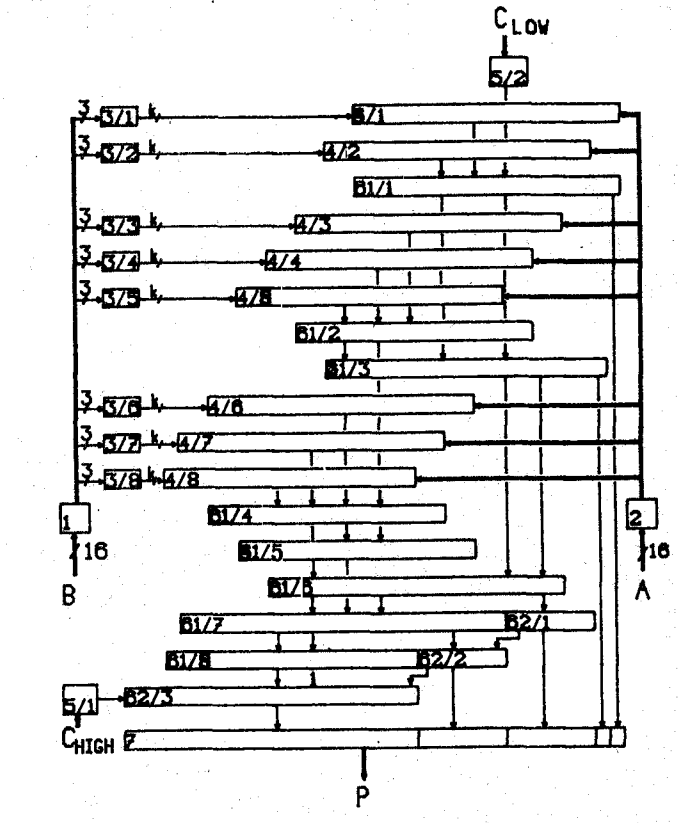
ahol az egyes idők jelentése és becsült értéke:

T_3 A vezérlő jeleket előállító egység működési ideje. A nagy terhelések miatt az egyébként egyszerű cellát ki kell egészíteni meghajtó fokozatokkal, így az idő 16ns-ra adódik.

T_4 Az összeadandó biteket előállító egység a logikai szimulációban egy komplex kapuból állt, amelynek műveleti idejére 8ns adódik.

T_{61} 5 db sorba kötött összeadó késleltetése, ami 60ns.

T_{62} Az átvitelgyorsítós összeadó számolási ideje, 48ns.



H 270 - 4

4. ábra. A megvalósított *MAD* makrocella blokkvázlata

A logikai szimulációból adódó sebesség így

$$T_{MAD} = 16ns + 8ns + 60ns + 48ns = 132ns . \qquad (13)$$

Ez önmagában is igen biztató eredmény, de az algoritmus jósága jobban elbírálható, ha ugyanolyan szimulációs feltételek mellett megvizsgálunk több különféle algoritmust használó szorzó kapcsolást. Az *5. ábrában* összefoglaltuk a tervezés során szimulált áramkörök főbb paramétereit.

	komplexitás C, kapuszám	sebesség T, μa	$T \cdot C$	Funkció
BOOTH algoritmus átvitelterjedéses összeadósorral	735	2,7	1985	16×16
BOOTH algortmus átvitelgyorsítós összeadóval	884	1,8	1591	16×16
Módosított BOOTH átvitelgyorsítós összeadó	927	1,0	927	16×16
$\bar{P}$ algoritmus átvitelterjedéses összeadó mátrix	2176	0,354	770	16×16
$\bar{P}$ algoritmus *CS* mátrix és *CLA* összeadó	2625	0,222	583	16×16+31
VUILLEMIN-féle kapcsolás	3814	0,138	526	16×16
a publikált kapcsolás	3258	0,132	430	16×16+35

H 270—5

5. ábra. Néhány szorzó áramkör adatai a logikai szimulációjuk alapján

Logikai szimuláció szintjén ez természetesen nem lehet más, mint a kapcsolást felépítő logikai kapuk száma és a szorzat kiszámolásának ideje. Magyarázatra szorul a két paraméter szorzatának feltüntetése a táblázatban. Azonos technológia mellett az áramkör fogyasztása arányos a komplexitással, s ha egy teljesítmény dimenziójú mennyiséget megszorzunk a műveleti idővel, akkor az így kapott mutató az egy szorzás elvégzéséhez felhasznált energiával arányos, tehát gazdaságosság szempontjából jellemzi az algoritmust.

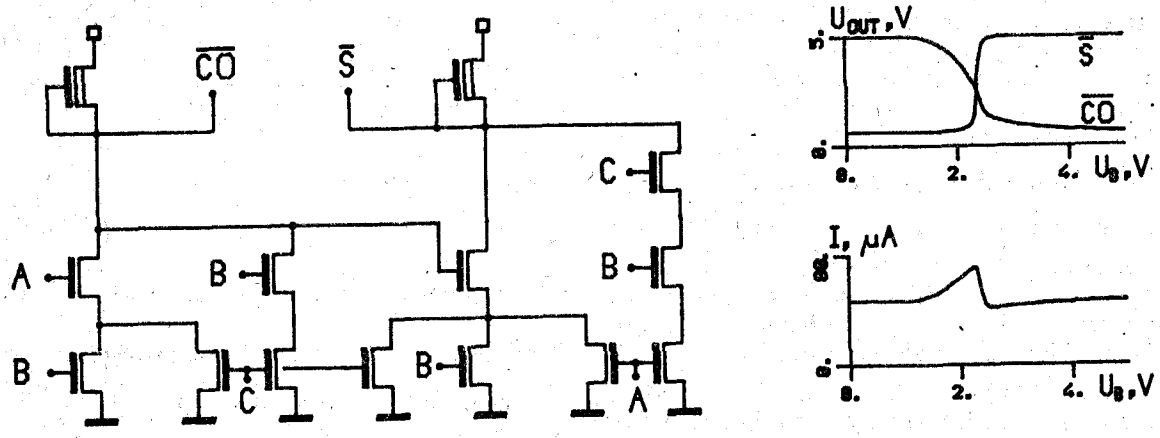
A táblázatban 3 szekvenciális és 4 kombinációs szorzó adatai láthatók. A *Booth* [3] és *módosított Booth* [4] algoritmusokkal számoló áramköröket váltott logikájú átvitelterjedéses és 2 szintű átvitelgyorsítót tartalmazó összeadókkal próbáltuk ki [2]. A katalóguslapok utalásai alapján feltehető, hogy a nyugati szorzók nagy részében P [5] vagy ehhez nagyon hasonló algoritmust használnak [6]. Táblázatunkban egy léptet és összead elvű összeadó mátrixot illetve egy átvitelmegőrzős összeadó mátrixot és egy átvitelgyorsítós végső összeadót tartalmazó kapcsolást találhatunk. A *Vuillemin algoritmust* [8] használó áramkört nem szimuláltuk, ennek paraméterei csak becsült értékek. Látható, hogy mind a sebesség, mind pedig a $T \cdot C$ paraméter alapján az új algoritmus a legelőnyösebb.

2.4. A logikai szimuláció és mérés problémája

Az áramkör szimulációja és mérése a nagy számú bemenet miatt ugyanolyan problémát jelent, mint a nagy memóriaáramköröké. Egy komplett ellenőrző méréshez legalább néhány mintaáramkört ki kellene próbálni az összes lehetséges bemeneti variációra. Tegyük fel, hogy rendelkezésünkre áll egy speciális mérőeszköz, amely 500ns-onként képes ellátni a mérendő áramkört az új bemenő adatokkal, s ellenőrizni az eredmény helyességét! A mérés ideje így

$$T_m = 2^{16+16+35} \cdot 500\ \text{ns} = 2^{67} \cdot 5 \cdot 10^{-7}\text{s} \approx 2\ 339\ 770\text{év} \qquad (14)$$

Nyilvánvaló, hogy egy ilyen mérés elvégezhetetlen. A tervezéshez használt *LOGSIM* logikai szimulációs program egyetlen bemeneti variáció esetén kb. 70 perc *CPU* idő felhasználásával szolgáltatta az eredményt, ami a nagy kapacitású de



6. *ábra.* Összeadó cella tranzisztorszíntű kapcsolása ós *DC* transzfer karakterisztikája a kimenetek feszültségével és az áramfelvétellel

H 270-6

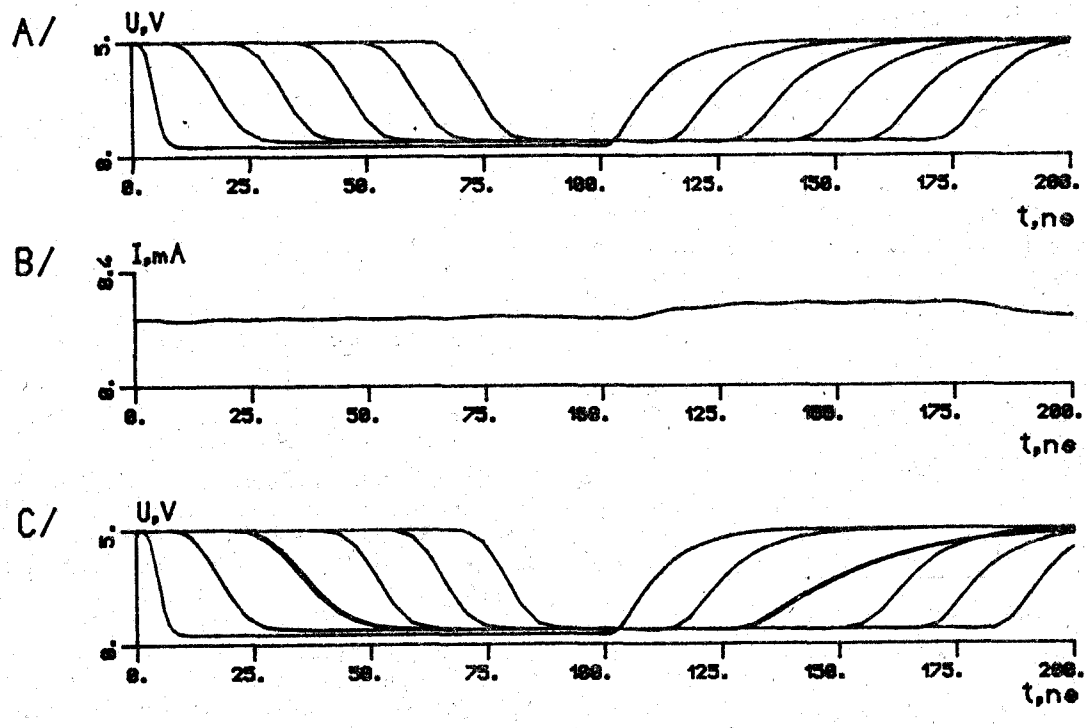
egyszerre több felhasználót kiszolgáló számítógépen csak napi égy futtatást tett lehetővé. Kis számú ellenőrző futtatás alapján nem garantálható ilyen nagy komplexitású kapcsolás hibátlansága, ezért írni kellett egy *FORTRAN* célprogramot a logikai kapcsolás helyességének megerősítésére. A program véletlenül generált vagy általunk megadott bemenő adatokkal kapu szinten, de csak logikai helyesség szempontjából vizsgálja az áramkört, s a kapcsolásból adódó végeredményt összehasonlítja a kiszámolt helyes eredménnyel. Segítségével néhány perc alatt több ezer bemenő adatra próbálható ki az áramkör. Futtatási tapasztalatok alapján hibás áramkörleírás esetén nagy valószínűséggel hibás eredményt kapunk, így nagy számú, csak helyes eredményt adó futtatás alapján a kapcsolás megfelelő biztonsággal helyesnek tekinthető.

2.5. *Analóg szimuláció*

Az analóg szimulációs futtatások során fontos információkat kapunk az áramkör várható egyenáramú és tranziens viselkedéséről, segítségükkel határozhatók meg a layouton alkalmazandó tranzisztorméretek. A logikai szimulációnál nagyobb biztonsággal becsülhető meg az áramkör sebessége, s kiszámolható a várható fogyasztás is. A szimulációkat *HERMES*-program felhasználásával végeztük. Az analóg szimuláció a véges számítógépi kapacitás miatt nem végezhető el egyszerre az egész áramkörre, egy futtatás során általában néhányszor tíz, legfeljebb száz tranzisztort tartalmazó áramköri részletet vizsgálunk. Áramkörünk szerencsére könnyen felosztható önálló funkciójú és kis komplexitású alapcellákra, amelyek így külön vizsgálhatók. Ezek közül csak a két legfontosabb cellát mutatjuk be, az egy bites teljes összeadót és az összeadandó biteket előállító kapcsolást.

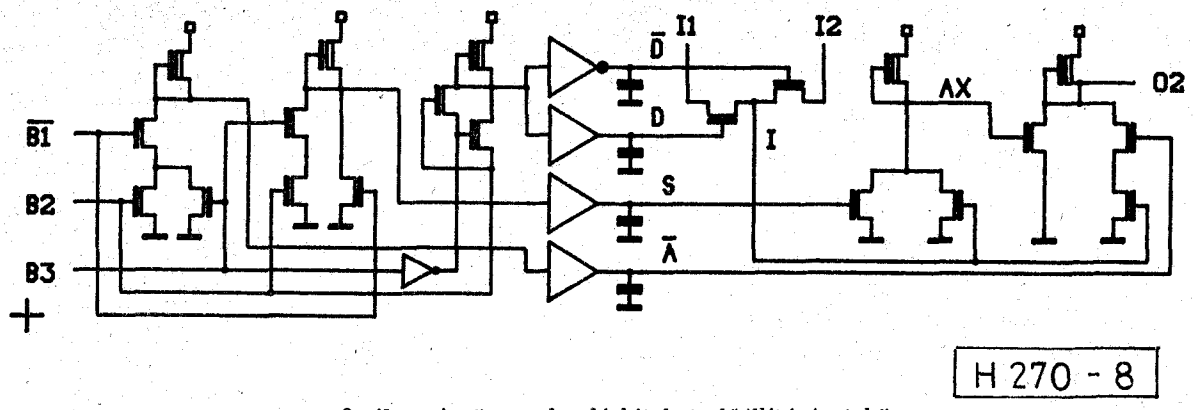
Az összeadó cella gondos tervezése azért különösen fontos, mert az átvitelmegőrzős összeadó hálózat a *MAD* cella műveleti idejének kb. felét, komplexitásának s így fogyasztásának kb. harmadát teszi ki. A használt *NSGT—1 N* csatornás poliSi vezérlőelektródás technológiához legjobban illeszkedő összeadó cella tranzisztorszintű kapcsolása és *DC* transzfer karakterisztikája látható a *6. ábrán*. A kapcsolás ugyanaz, mint a korábbi tervekben is szereplő [7], de a kisebb fogyasztás és helytakarékosság érdekében kisebb *W/L* arányú tranzisztorokat használunk. A tranziens futtatások szerint ez jelentős fogyasztáscsökkenést és némi lassulást okozott. Az 5 fokozatú átvitelmegőrzős összeadó hálózat működését elég 5 db sorba kötött összeadó cella szimulációjával vizsgálni. A *7/a. ábrán* a gerjesztő feszültség és az összeadó cellák szumma kimenetének feszültsége látható az idő függvényében, a *7/b. ábra* az 5 egység együttes áramfelvételét mutatja. Az 5 cella eredő késleltetése a korábbi 60ns helyett 70ns lett, az áramfelvétel pedig 800 μA helyett 260 μA, így a közel 150 összeadót tartalmazó átvitelmegőrzős összeadó hálózat fogyasztása 7,5 mA körül várható.

A párhuzamos összeadó hálózat következtében a logikailag egymást követő összeadó cellák a chip-en nem mindig helyezhetők egymás mellé, s mivel a jelterjedés iránya merőleges a tápfeszültség és a vezérlő jelek fém vezetékeire, az össze-



7. ábra. a) Az összeadó sor tranziens feszültségválasza; *b*) Az összeadó sor áramfelvétele; *c*) A polisi csíkkal kiegészített összeadó sor tranziens feszültségválasza

H 270-7



8. ábra. Az összeadandó biteket előállító áramkör

köttetések csak poliSi-mal valósíthatók meg. A layouton a leghosszabb ilyen csík kb. 800 μm-es, aminek késleltető hatását egy *RC* lótrahálózattal vehetjük legpontosabban figyelembe. A szimuláció során a létrát a 2. és 3. összeadó cella közé kötöttük, az így végzett újabb futtatás eredménye a *7/c. ábrán* látható. A poliSi csík elején és végén lévő feszültséget is kirajzoltattuk, de olyan kicsi köztük az időeltolódás, hogy ez csak a vonal vastagodását eredményezi az ábrán. Jól megfigyelhető, hogy a hosszú vezeték kapacitása terheli az előző fokozatot, ami a jelváltozás sebességét csökkenti, így az eredő késleltetés is megnő 80ns-ra. Nagyobb áramú tranzisztorokkal végzett futtatások nem hoztak elegendő gyorsulást, így megmaradtunk a kicsit lassabb kis fogyasztású és kis helyigényű megoldás mellett.

A másik fontos alapcella az összeadandó biteket állítja elő, amiből $8 \cdot 17 = 136$ szükséges az áramkörben. Mivel ezek egymással párhuzamosan működnek, s így a számolási idő kevesebb mint 10 %-át teszik ki, itt elsődleges szempont a lehető legkisebb méret és fogyasztás elérése volt. Egy kis cella tranziens vizsgálatából nem kapunk értékelhető eredményt, ezért ezt a vezérlő jelek előállításával összevontan szimuláltuk.

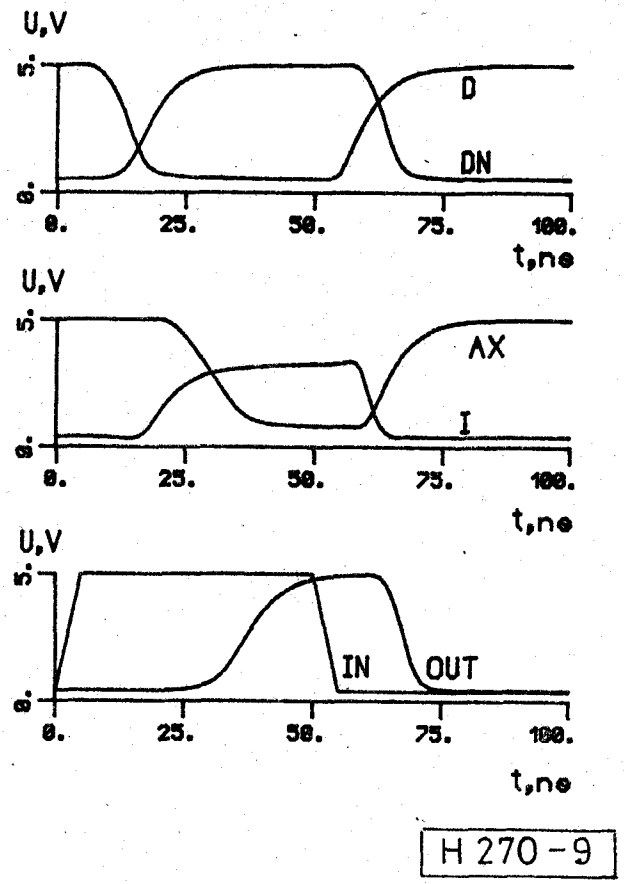
Több lehetséges megoldás összehasonlítása után az tűnt legcélszerűbbnek, ha osztó (D), összeadó (A), és kivonó (S) utasításokat állítunk elő vezérlő jelként. Ha D aktív, akkor az eggyel magasabb helyiértékű bit kerül feldolgozásra, $A=1$ esetén ponált, $S=1$ esetén negált értékkel kerül a kimenetre. $A=S=0$ esetén a kimenet is 0, $A=S=1$ nem fordulhat elő. A vezérlő jelek egyszerűen előállíthatók a szorzó 3 megfelelő bitjéből:

$$\overline{A}_q = \overline{b}_{2q} \cdot (b_{2q+1} + b_{2q+2}) \quad q = 0, \mathrm{i}, \ldots, 7 \qquad (15)$$

$$D_q = b_{2q+1} + b_{2q+2} \qquad (16)$$

$$S_q = \overline{b_{2q} + b_{2q+1} \cdot b_{2q+2}} \qquad (17)$$

Mivel a vezérlő jeleket 17 cellához kell eljuttatni, minden kimeneten egy meghajtó fokozatot kell használnunk. A bitek előállításánál szükséges osztást célszerű két transzfer kapuval, tehát passzív nem fogyasztó hálózattal megvalósítani. Az általa számolt I jelből egyszerűen számolható a kimenet:



9. ábra. Az összeadandó biteket előállító áramkör szimulációja

$$02 = \overline{A} \cdot I + \overline{S + I} \qquad (18)$$

A szimulált kapcsolás látható a *8. ábrán* a vezérlő jelek előállításával, a meghajtó fokozatokkal és a bitelőállító egységgel. A kapacitások a kb. 2,5 mm-es fémcsíkok és a többi 16 cellában meghajtandó tranzisztorok terhelő kapacitását modellezik. A kapcsolás így a szorzó 3 bitje és a szorzandó két bitje alapján egy kimenő bitet szolgáltat. A vezérléseket úgy választottuk meg, hogy a jelváltozásnak a lehető leghosszabb úton kelljen terjednie. Ez akkor valósul meg, ha az osztó jel változik ós $II = \overline{I2} \cdot b_{2q+2}$ átváltásakor D, $\overline{D}$ vezérlő jelek változnak, s mivel a két bemenő bit is eltérő,

I értéke is átvált. Összeadáskór, tehát $S = \bar{A} = 0$ esetén I változása a belső *NOR* kapu átbillenése után változtathatja csak meg a kimenetet, így valóban az összes kapu sorba kötve működött. A *9. ábrán* láthatók a szimulációs eredmények, ahol jól megfigyelhető a jelterjedós. A harmadik grafikonról közvetlenül leolvasható, hogy legkedvezőtlenebb esetben is 30ns alatt előállítható az összes összeadandó bit. A vezérlő jeleket előállító egység fogyasztása a meghajtó fokozatokkal együtt 600 μA, tehát a 8 egység együttes áramfelvétele 4,8 mA körül várható. A biteket generáló egység árama 40 μA, a teljes áramkör 136 cellája így a szimulációk alapján 5,5 mA-t fog fogyasztani.

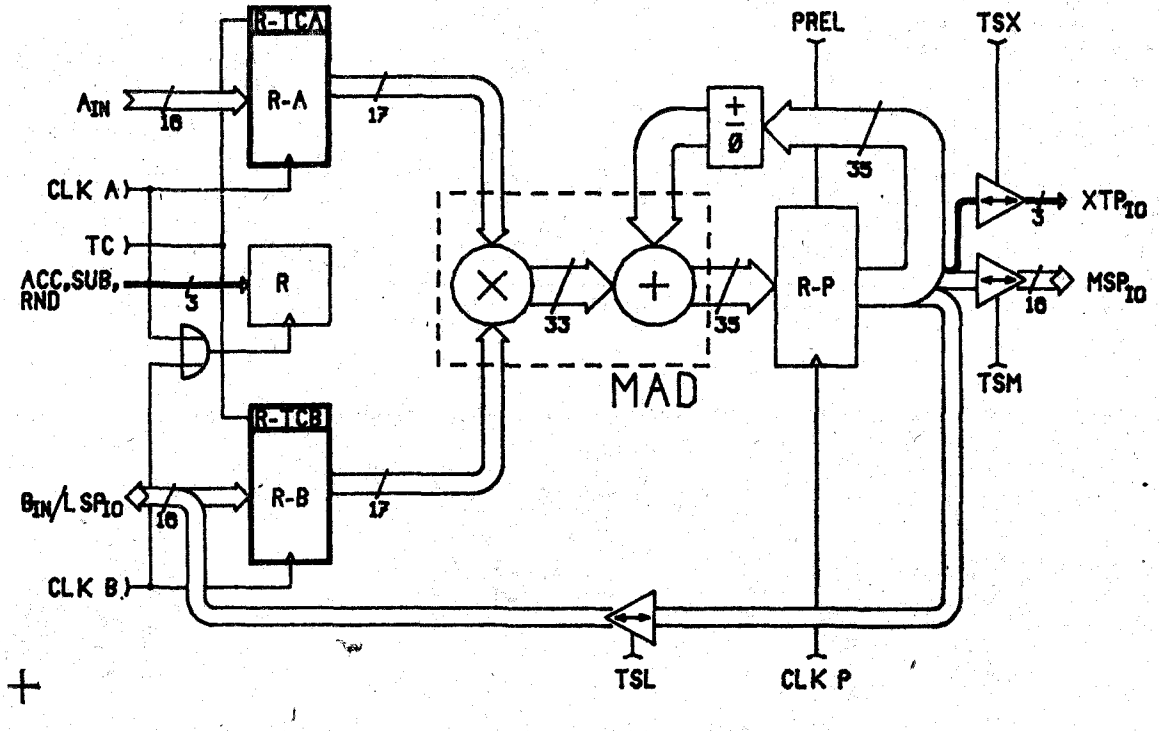
Nem térünk ki részletesen a 21 bites átvitelgyorsítós összeadó szimulációjára, mivel ez a kb. 800 tranzisztoros komplexitás miatt elég összetett probléma, s teljesen korrekt vizsgálatra nincs is lehetőség. Az egyes blokkokon végrehajtott szimulációk alapján az egység fogyasztása 3,5 mA, számolási ideje pedig 50—60 ns körül várható.

Ezzel a működést döntően befolyásoló egységeket megvizsgáltuk, s becslést adhattunk a *MAD* makrocella sebességére. A részeredményeket összegezve a cella számolási ideje 160—170ns, áramfelvétele pedig kisebb mint 25 mA. Ezek csak szimuláció alapján becsült, tehát fenntartásokkal fogadható értékek, amelyek a szimulációs bizonytalanságok és a technológiai szórások miatt módosulhatnak a kész áramkörben, az azonban az elkészült layout alapján pontosan megmondható, hogy a *MAD* makrocella összesen 4099 db meghajtó és 1029 db terhelő tranzisztorból épül fel.

3. A TMC2010MAC szorzó-akkumuláló áramkör

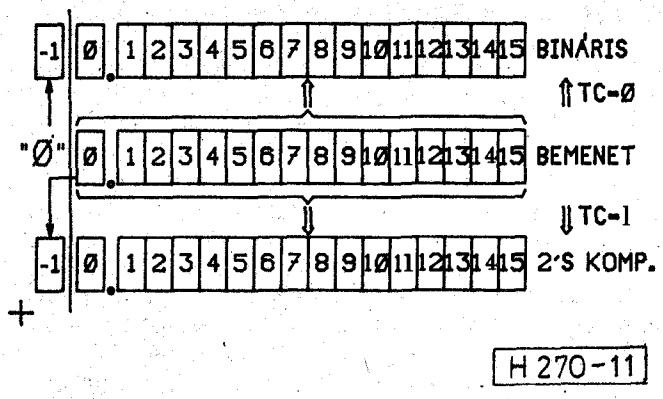
Az eddigiekben részletesen bemutatott szórzó-összeadó cella önmagában még nem kész áramkör, de egy népes termékcsalád alakítható ki belőle. Nagyobb bonyolultságú áramkörnek lehet egy szorzó-összeadó funkciót ellátó makrocellája, vagy ki- és bemenő fokozatokkal valamint vezérlő egységgel ellátva önálló áramkörré alakítható. A jelenlegi technológiai szint mellett a cella mérete olyan nagy, hogy csak az utóbbi megoldás jöhet szóba. A cella első konkrét alkalmazása egy szorzó-akkumuláló integrált áramkör lesz, amely teljesen láb és funkciókompartibilis a *TRW* cég közismert *TDC1010*-es és *TMC2010*-es áramköreivel.

Az eszköz blokkvázlata a *10. ábrán* látható. Az áramkör legfontosabb egysége a szorzó-összeadó funkciót megvalósító *MAD* cella. A 16 bites B szorzó ós A szorzandó egymástól teljesen függetlenül olvasható be egy-egy belső tárolóba. A *MAD* kimenetén lévő eredmény *CLK P* felfutó élére a 35 bites eredmény tárba íródik. Az eredmény két 16 és egy 3 bites csoportra osztva külön is kiolvasható a 3 állapotú kimeneteken, amelyeken keresztül az eredmény tár kívülről is feltölthető. Az áramkör 64 kivezetőjű tokban kerül forgalomba ami nem elegendő az összes ki- ós bemenetnek, így a B bemenet ós az eredmény alsó 16 bitje ugyanazokat a kivezetéseket használja multiplexálva. Így is lehetőség nyílik arra, hogy egyidejűleg beírjuk az A B bemeneteket s kiolvassuk az előző eredmény felső 19 bitjét, ami igen gyors adatforgalmat eredményez. A szorzó-akkumuláló funkció



10. ábra. A *TMC2010MAC* szorzó-akkumuláló áramkör blokkvázlata

H 270-10



11. ábra. A választott kód figyelembe vétele a belső számábrázolásnál

úgy valósul meg, hogy az előző részeredményt tartalmazó eredmény tár kimenete *IC*-n belül vissza van csatolva a *MAD* cella összeadandó bemenetére. A visszacsatoló ágban lehetőség van az adatok negálására ós törlésére is, így az eszköz $A \cdot B$, $A \cdot B - P$ és $A \cdot B + P$ műveleteket tud végrehajtani. Párhuzamos adatforgalomnál előnyös, hogy lehetőség van az alsó 16 bit egyszerű elhagyáis helyett kerekített eredmény képzésére is, ami kisebb hibát eredményez a feldolgozás során. Az eszköz 2's komplemens kódú és előjel nélküli számokkal tud dolgozni.

Ennél részletesebben itt nem foglalkozunk az eszköz működésével, bővebb információ egyelőre a *TRW* katalógusaiban található [9]. Egyetlen kis eltérést szeretnénk kiemelni, amelyben eszközünk felülről kompatibilis a *TRW* áramkörökkel. A szorzó algoritmusok egy része könnyen átalakítható úgy, hogy többféle kódban is alkalmasak legyenek a számolásra, amire példa a *MEV U400EBM* 8 bites szorzója is [1]. Az új algoritmus azonban csak 2's komplemens kódú számok szorzására alkalmas, nem végezhetők el egyszerű módosítások az előjel nélküli számok fogadásához. Mivel a kompatibilitáshoz mindenképpen szükséges. mindkét kód fogadása, a MAD cellát kibővítettük 17·17 bites szorzás elvégzésére, s így a 11. ábra szerint számolunk a két kóddal. Ha a beérkező 16 bites szám 2's komplemens kódú, akkor a bővítési szabály értelmében az új biten megismételjük az eredeti előjegyet. Ha a szám előjel nélküli, akkor a bővítés helyére 0-t írunk, mert így a 17 bites 2's komplemens szám értéke megegyezik a 16 bites előjel nélküli számével. Az alsó 16 bitre mindkét esetben változtatás nélkül beírjuk a beérkező számot. A bővítés következtében elvégezhető az egyébként hibás eredményt adót $(-1)\cdot(-1)=1$ szorzás is, hiszen az új tartományban a szorzat értéke $-4 \leqq P_v < 4$ lehet. Előny a *TRW* algoritmusával szemben, hogy *A* és *B* mindegyike bármelyik kódban lehet, így vegyes kódú szorzás is elvégezhető. Megfelelő biztonsági tartalékkal adott becslések szerint az eszköz 500 ns-onként képes szolgáltatni egy új eredményt, ami 2 MHz-es működési frekvenciának felel meg. A 35 kimeneti meghajtó fokozat áramfelvétele jelentősen függ a kimeneti állapottól, hiszen LOW szintnél nagy áramot kell elnyelni. Átlagos működést alapul véve az eszköz teljes fogyasztása 60 mA körül várható.

IRODALOM

[1] *Tuzson Tibor, Erdélyi János:* Az U400EBM 8-bites párhuzamos szorzó; Magyar elektronika II. évf. 2. szám 1985, 45—49 oldal

[2] *Bupprich Péter:* Digitális aritmetika Mérnöktovábbképző jegyzet

[3] *Bohus Miklós, Horváth László:* Digitális számítógépek BME jegyzet, J5—1116, Tankönyvkiadó, 1984

[4] *L. P. Rubinfield:* A Proof of the Modified Booth 's Algorithm for Multiplioation; IEEE Trans. Comp. Oct. 1975 pp. 1014—1015

[5] *J. A. Gibson, R. W. Gibbard:* Synthesis and Comparison of Two' e Complement Parallel Multipliers; IEEE Trans. Comp., October 1975 pp. 1020—1027

[6] *G. R. Baugh, B. A. Wooley:* A Two' s Complement Parallel Array Multiplioation Algorithm; IEEE Trans. Comp. vol. C—22 No. 12, December 1973 pp. 1045—1047

[7] *Szőke Sándor:* 16 bites szorzó áramkör logikai tervezése a részegységek analóg szimulációjával; Híradástechnika XXXVI. évf. 1985. 7. szám 324—330 oldal

[8] *Jean Vuillemin:* A Very Fast Multiplioation Algorithm for VLSI Implementation; Integration, The VLSI Journal 1983 pp. 39—52

[9] TRW TMC2010, Preliminary Information; TRW Inc. 1984