TECHNICKÁ UNIVERZITA V KOŠICIACH

Fakulta elektrotechniky a informatiky

KATEDRA ELEKTRONIKY A MULTIMEDIÁLNYCH TELEKOMUNIKÁCIÍ

ING. MILOŠ DRUTAROVSKÝ, CSC.

SIGNÁLOVÉ PROCESORY V ČÍSLICOVOM SPRACOVANÍ SIGNÁLOV

HABILITAČNÁ PRÁCA

KOŠICE, NOVEMBER 1999

(REV.1)

OBSAH

ZOZNAM POUŽITÝCH SKRATIEK

AAU	**A**ddress **A**rithmetic **U**nit, adresová aritmetická jednotka
AC3	metóda kompresie audio signálov
A/D	**A**nalog to **D**igital, analógovo-číslicový (prevodník)
ALU	**A**rithmetic **L**ogic **U**nit, aritmeticko-logická jednotka
ANOVIS	*Acoustic NOise and VIbration Signal Analyser*, systém na akustickú a vibračnú analýzu signálov
API	**A**pplication **P**rogramming **I**nterface, aplikačné programové rozhranie
ASCII	*American Standard Code for Information Interchange*, štandardný americký kód na výmenu informácií
ASIC	*Application Specific Integrated Circuit*, zákaznícky obvod
ASK	**A**mplitude **S**hift **K**eying, amplitúdové kľúčovanie
AWGN	**A**dditive **W**hite **G**aussian **N**oise, aditívny biely (nekorelovaný) Gaussov šum
BDTI	*Berkeley Design Technology, Inc.*
BER	**B**ite **E**rror **R**ate, bitová chybovosť
BFU	**B**it **F**ield **U**nit, jednotka pre bitové operácie
BIOS	**B**asic **I**nput **O**utput **S**ystem, základný vstupno-výstupný systém
BMU	**B**it **M**anipulation **U**nit, jednotka pre bitové operácie
BU	**B**ranch **U**nit, jednotka pre vetvenie
CCOP	**C**yclic Code **CO**-**P**rocessor, koprocesor pre cyklické kódy
CELP	**C**ode **E**xcited **L**inear **P**rediction, súbor metód kompresie (reči) na báze lineárnej predikcie
CLIW	**C**onfigurable **L**ong **I**nstruction **W**ord, konfigurovateľné dlhé inštrukčné slovo
CMOS	*Complementary Metal Oxide Semiconductor*, komplementárny MOS
COFF	**C**ommon **O**bject **F**ile **F**ormat, štandardný formát relatívnych súborov
CPLD	**C**omplex **P**rogrammable **L**ogic **D**evices, zložité programovateľné obvody
CPM	**C**ontinuos **P**hase **M**odulation, modulácia so spojitou fázou
CPU	**C**entral **P**rocessing **U**nit, centrálna procesorová jednotka
CRC	**C**yclic **R**edundancy **C**heck, metóda detekcie chýb s využitím syndrómu cyklických blokových kódov
ČSS	**Č**íslicové **S**pracovanie **S**ignálov
DA	**D**istributed **A**rithmetic, distribuovaná aritmetika
D/A	**D**igital to **A**nalog, číslicovo-analógový (prevodník)
DCT	**D**iscrete **C**osine **T**ransform, diskrétna kosínusová transformácia
DFT	**D**iscrete **F**ourier **T**ransform, diskrétna Fourierova transformácia
DIF	**D**ecimation **I**n **F**requency, decimácia vo frekvenčnej oblasti
DIT	**D**ecimation **I**n **T**ime, decimácia v časovej oblasti
DM	**D**ata **M**emory, dátová pamäť

DMA	**D**irect **M**emory **A**ccess, priamy prístup do pamäte
DP	**D**ata **P**rocessor, dátový procesor
DRAM	**D**ynamic **RAM**, dynamická RAM
DSP	**D**igital **S**ignal **P**rocessor, číslicový signálový procesor
DVD	**D**igital **V**ideo **D**isc, digitálny video disk
EFCOP	**E**nhanced **F**ilter **CO-P**rocessor, rozšírený filtračný koprocesor
ENIAC	***E**lectronic **N**umerical **I**ntegrator **A**nd **C**alculator*
EPROM	***E**rasable **P**rogrammable **R**ead-**O**nly **M**emory*, pamäť na čítanie mazateľná UV žiarením
ETSI	***E**uropean **T**elecommunications **S**tandard **I**nstitute*, Európsky telekomunikačný štandardizačný úrad
EU	**E**urópska **Ú**nia
FCOP	**F**ilter **CO-P**rocessor, filtračný koprocesor
FFA	**F**inite **F**ield **A**rithmetic, aritmetika v konečných (Galoisových) poliach
FFT	**F**ast **F**ourier **T**ransform, rýchla Fourierova transformácia
FIR	**F**inite **I**mpulse **R**esponse, konečná impulzová odpoveď
FLASH	elelektricky prepisovateľná pamäť na čítanie
FPGA	**F**ield **P**rogrammable **G**ate **A**rray, užívateľsky programovateľné hradlové pole
FSK	**F**requency **S**hift **K**eying, frekvenčné kľúčovanie
G	**G**iga, miliarda (prípadne 2^{30} =1073741824 pre veľkosť pamäte)
GMSK	**G**aussian **M**inimum **S**hift **K**eying, Gaussove kľúčovanie s minimálnym frekvenčným posunom
GOPS	**G**iga **O**perations **P**er **S**econd, miliárd operácií za sekundu
GSM	**G**roup **S**pécial **M**obiles, digitálny mobilný systém
IEEE 754	štandard pre reprezentáciu čísel v pohyblivej rádovej čiarke
IDFT	**I**nverse **DFT**, inverzná DFT
IFFT	**I**nverse **FFT**, inverzná FFT
IIR	**I**nfinite **I**mpulse **R**esponse, nekonečná impulzová odpoveď
ILP	**I**nstruction **L**evel **P**arallelism, paralelizmus na úrovni inštrukcií
IM	**I**nstruction **M**emory, inštrukčná pamäť
IP	**I**nstruction **P**rocessor, inštrukčný procesor
I/O	vstupno-výstupný
JTAG	**J**oin European **T**est **A**ction **G**roup, skupina, ktorá vytvorila testovací štandard IEEE 1149.1-1990
K	**K**ilo, tisíc (prípadne 2^{10} =1024 pre veľkosť pamäte)
LIW	**L**ong **I**nstruction **W**ord, dlhé inštrukčné slovo
LSB	**L**east **S**ignificant **B**it, najmenej významový bit
LSI	**L**arge **S**cale **I**ntegration, vysoký stupeň integrácie
M	**M**ega, milión (prípadne 2^{20} =1048576 pre veľkosť pamäte)
MAC	**M**ultiply and **AC**cumulate, operácia násobenia a akumulácie
MF	**M**atched **F**ilter, prispôsobený filter
MFOPS	**M**illion **F**loating **O**perations **P**er **S**econd, milióny operácií v pohyblivej rádovej čiarke za sekundu
MIMD	***M**ultiple **I**nstructions **M**ultiple **D**ata*
MISD	***M**ultiple **I**nstructions **S**ingle **D**ata*
MIPS	**M**illion **I**nstructions **P**er **S**econd, milióny inštrukcií za sekundu
MLSE	***M**aximum **L**ikelihood **S**equence **E**stimation*, odhad postupnosti

	s maximálnou vierohodnosťou
MOPS	**M**illion **O**perations **P**er **S**econd, milióny operácií za sekundu
MOS	**M**etal **O**xide **S**emiconductor, polovodič s kovovým oxidom
MOSFET	**M**etal **O**xide **S**emiconductor **F**ield **E**ffect **T**ransistor, poľom riadený tranzistor s hradlom izolovaným kovovým oxidom
MPEG	**M**oving **P**icture **E**xperts **G**roup, tiež metóda kompresie audio a video signálov
NCEG	**N**umeric **C** **E**xtensions **G**roup, skupina pre numerické rozšírenie jazyka C
NMOS	**N** channel **MOS**, štruktúra MOS s kanálom typu N
NOP	**N**o **OP**eration, prázdna operácia
OFDM	***O**rthogonal **F**requency **D**ivision **M**ultiplex*, otogonálny frekvenčne delený multiplex
OnCE	**On C**hip **E**mulation, emulácia s využitím obvodov na čipe procesora
OS	**O**peračný **S**ystém
$O(N^x)$	zložitosť úmerná mocnine N^x
PLL	**P**hase **L**ock(ed) **L**oop, slučka fázového závesu
PROM	**P**rogrammable **R**ead **O**nly **M**emory, trvalo programovateľná pamäť
RAM	**R**andom **A**ccess **M**emory, pamäť s ľubovoľným prístupom
RFFT	**R**eal **FFT**, FFT s čisto reálnym vstupom
RISC	**R**educed **I**nstruction **S**et **C**omputer, počítač (procesor) s redukovaným súborom inštrukcií
RNS	**R**esidue **N**umber **S**ystem, zvyškový číselný systém
SA	**S**ystolic **A**rray, synchrónne systolické pole
SCI	**S**erial **C**ommunication **I**nterface, sériové komunikačné rozhranie
SIM	**S**ubscriber **I**dentification **M**odule, (čipový) identifikačný modul účastníka
SIMD	***S**ingle **I**nstruction **M**ultiple **D**ata*
SMD	**S**urface **M**ount **D**evice, súčiastka pre povrchovú montáž
SRAM	**S**tatic **RAM**, statická RAM
SSI	**S**erial **S**ynchronous **I**nterface, sériové synchrónne rozhranie
TDMA	**T**ime **D**ivision **M**ultiple **A**ccess, viacnásobný prístup s časovým delením
TI	***T**exas **I**nstruments*
ULSI	**U**ltra **L**arge **S**cale **I**ntegration, ultravysoký stupeň integrácie
VA	**V**iterbi **A**lgorithm, Viterbiho algoritmus
VCOP	**V**iterbi **CO-P**rocessor, Viterbiho koprocesor
VelociTI	VLIW modifikácia firmy Texas Instruments
VLES	**V**ariable **L**ength **E**xecution **S**et, inštrukčná sada s premenlivou dĺžkou
VLIW	**V**ery **L**arge **I**nstruction **W**ord, inštrukčné slovo s veľkou dĺžkou
VLSI	**V**ery **L**arge **S**cale **I**ntegration, veľmi vysoký stupeň integrácie
VMOS	technológia N**MOS** v žliabku **V**
VSELP	*Vector **S**um **E**xcited **L**inear **P**redictive Coding*, variant CELP
WA	**W**avefront **A**rray, asynchrónne systolické pole
xDSL	(asymetric) **D**igital **S**ubscriber **L**ine, normy pre (asymetrické) modemy na prenos digitálnych dát pomocou klasických prístupových sietí (medené dvojlinky) s rôznou rýchlosťou (x) prenosu
3BH	3-bodové Blackmanovo-Harrisovo okno
3D	3 **D**imensional, 3-rozmerný

4BH	4-bodové Blackmanovo-Harrisovo okno
μP	mikroprocesor

1 ÚVOD

Číslicové spracovanie signálov (ČSS) sa počas uplynulého desaťročia zmenilo z vedeckej disciplíny, ktorá bola známa len úzkemu okruhu pracovníkov z univerzít a výskumných ústavov na všeobecne známy pojem, ktorý významným spôsobom ovplyvňuje náš každodenný život. Nárast využívania elektronických súčiastok využívajúcich princípy číslicového spracovania signálov v *spotrebnej elektronike*, *telekomunikačných zariadeniach* a *počítačovej technike* je tak výrazný, že ČSS je v súčasnosti jednou z hlavných síl rozvoja polovodičového priemyslu. Tento výrazný nárast v rozsahu využitia a výkonnosti týchto súčiastok má pôvod v rozvoji obvodov veľmi vysokej hustoty integrácie (VLSI –Very Large Scale Integration) a exponenciálnom náraste úrovne integrácie známom ako Moorov zákon.

Počas predchádzajúcich troch desaťročí sa úroveň integrácie VLSI obvodov zvyšovala štvornásobne každé tri roky, pričom hodinová frekvencia týchto obvodov taktiež neustále rastie. Tieto faktory priamo ovplyvňovali a stále ovplyvňujú výkonnosť číslicových integrovaných obvodov a je predpoklad, že tento nárast sa udrží minimálne aj v nasledujúcom desaťročí. Po dosiahnutí určitej hustoty integrácie bolo koncom 70-tych rokov možné vytvoriť monolitický čip – *číslicový signálový procesor* (DSP – Digital Signal Processor), ktorý významným spôsobom ovplyvnil prenikanie metód číslicového spracovania do praxe. Možnosť monolitickej integrácie bola hlavným faktorom, ktorý umožnil významné zníženie ceny, príkonu a poruchovosti zariadení využívajúcich číslicové spracovanie signálov a tým aj ich masové nasadenie.

Úroveň mikroelektroniky však nie je jediným faktorom, ktorý ovplyvňuje rozvoj súčiastok a zariadení na báze ČSS. Číslicové spracovanie signálov je v súčasnosti už samostatnou vedeckou disciplínou a naďalej sa veľmi rýchlo rozvíja. Bola rozpracovaná samostatná teória algoritmov ČSS a nájdené efektívne postupy pre ich výpočet. Pre súčasné obdobie je charakteristická vzájomná súvislosť *teórie algoritmov* ČSS a *architektúr VLSI obvodov*. Tieto dve oblasti sa vzájomne podnecujú a doplňujú, pričom výsledky tejto vzájomnej symbiózy sa najviac prejavujú pri rozpracovaní paralelných algoritmov ČSS na pravidelne sa opakujúce základné elementy, ktoré sa dajú pomocou VLSI čipov s využitím dostupnej technológie jednoducho realizovať.

Teória paralelných počítačových systémov je známa a dobre rozpracovaná už niekoľko desaťročí. Veľmi populárne sú napr. *paralelné architektúry* SIMD, MIMD a MISD. Teória ČSS tiež významným spôsobom prispela do oblasti paralelných systémov rozpracovaním *systolických architektúr* už na začiatku 80-tych rokov.

Počas dvadsaťročného vývoja sa monolitické DSP vyvíjali spočiatku klasickým evolučným vývojom, pričom vývoj vychádzal predovšetkým z technologických zlepšení dostupnej CMOS technológie a prejavoval sa zmenšovaním dĺžky inštrukčného cyklu a zväčšovaním interných pamätí. V posledných troch-štyroch rokov začína vývoj DSP nadobúdať (minimálne z hľadiska integrácie nových prvkov) znaky revolučného rozvoja a je charakteristický predovšetkým väčším využívaním paralelizmu s cieľom zvýšiť priepustnosť systémov na báze DSP. Existuje niekoľko základných spôsobov využívania

paralelizmu, ktoré sa v súčasnosti v oblasti DSP uplatňujú. Je to jednak zvýšenie paralelizmu integrovaním viacerých funkčných jednotiek na jeden čip ako aj trend využívania predovšetkým SIMD a systolických architektúr, čo sa prejavuje predovšetkým v oblasti nových tzv. *multimediálnych procesorov* a *multimediálnych rozšírení* štandardných procesorov. Výpočtová výkonnosť programovateľných monolitických DSP sa tak posúva na úroveň, ktorá bola pred niekoľkými rokmi nepredstaviteľná.

Využívanie paralelizmu je z pohľadu počítačových architektúr značne prepracovaná metóda zvyšovania výpočtového výkonu, v oblasti monolitických čipov je však potrebné splniť veľmi špecifické požiadavky. Medzi najkritickejšie patria minimalizácia príkonu a množstva využívaných pamätí (umožňuje zníženie ceny čipu), čo vyžaduje výraznú modifikáciu už existujúcich princípov, prípadne aj vývoj úplne nových riešení.

Neoddeliteľnou súčasťou programovateľných DSP je ich *programové vybavenie*, ktoré je prostriedkom na *transformáciu* algoritmov ČSS do vykonateľného programu pre programovateľné DSP. Aj napriek pokroku v architektúrach a vývojových prostriedkoch je v odbornej verejnosti stále rozšírený názor, že programovanie DSP je náročná činnosť, často vyžadujúca programovanie v asembleri. Jedným z cieľov nových paralelných architektúr je aj umožnenie širšieho využívania vyšších programovacích jazykov pre programovanie DSP.

Základným cieľom pri využívaní DSP je *efektívne mapovanie algoritmov* ČSS do architektúry DSP a programové prostriedky sú prostriedkom na dosiahnutie tohto cieľa. Vzhľadom na špecifickú architektúru týchto technických prostriedkov je často potrebné pôvodné algoritmy ČSS optimalizovať pre cieľovú architektúru DSP. Úspešné praktické využitie DSP tak vyžaduje zvládnutie troch relatívne samostatných oblastí:

- architektúr DSP,
- ich programového vybavenia,
- teórie ČSS.

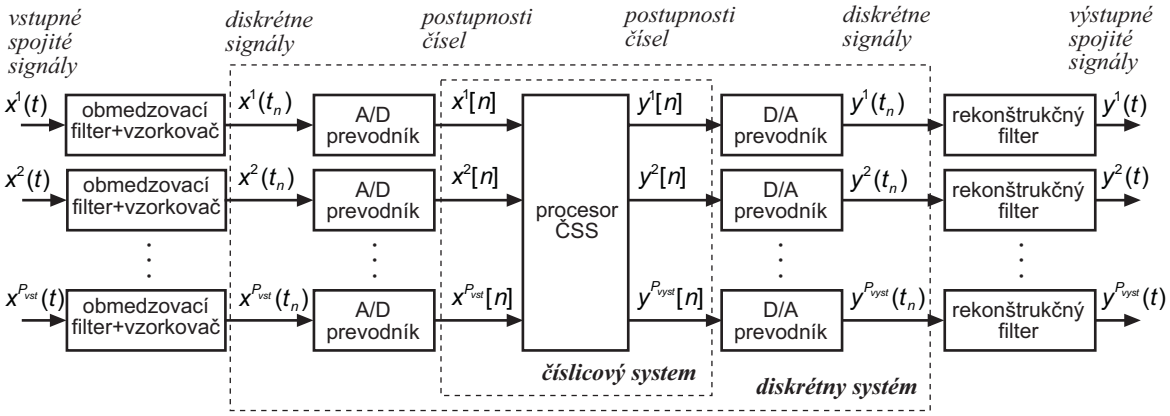
Predkladaná habilitačná práca je pokusom o analýzu a naznačenie trendov vývoja v týchto oblastiach a zhrnutie výsledkov a vedomostí, ktoré autor dosiahol počas pôsobenia na Katedre elektroniky a multimediálnych telekomunikácii v Košiciach. Práca je členená do piatich kapitol. Druhá kapitola opisuje základnú architektúru všeobecného procesora ČSS, niektoré charakteristické parametre týchto procesorov, vplyv technológie VLSI na ich vývoj a ich základnú klasifikáciu. Historický vývoj programovateľných DSP a stručný opis ich základných stavebných blokov je uvedený v tretej kapitole. V tejto kapitole je uvedený aj súbor testov používaný na hodnotenie výkonnosti rôznych (nie len DSP) procesorov z pohľadu algoritmov ČSS. Štvrtá kapitola opisuje možnosti využitia paralelizmu na rôznych úrovniach architektúry v súčasných komerčne dostupných ako aj v najnovších ohlásených DSP. Vývojovým prostriedkom pre tvorbu aplikácií na báze DSP je venovaná piata kapitola. Šiesta kapitola opisuje vybrané algoritmy ČSS, ktoré autor práce v uplynulých rokoch realizoval pomocou DSP a ich základným cieľom je poukázať na skutočnosť, že úspešné nasadenie DSP často vyžaduje optimalizáciu implementovaných algoritmov, ktorá vyžaduje okrem dobrej znalosti architektúry cieľového DSP aj využitie poznatkov z teórie ČSS.

2 SYSTÉMY ČÍSLICOVÉHO SPRACOVANIA SIGNÁLOV

Číslicové spracovanie signálov hlboko preniklo a intenzívne preniká do najrôznejších odborov ľudskej činnosti. Tomu výrazne napomáha technologický rozvoj súčiastkovej základne a vývoj nových návrhových postupov. Napríklad klasické číslicové obvody sú nahradzované štandardnými hradlovými poliami. Sú zlepšované parametre používaných štandardných polovodičových technológií (CMOS) [1] a zavádzané nové materiály pre výrobu veľmi rýchlych obvodov (GaAs) [2]. Vzniká rad nových zákazníckych, polozákazníckych a špeciálnych obvodov ako sú napr. grafické a multimediálne signálové procesory, FFT procesory, programovateľné číslicové filtre a pod. Cieľom tejto kapitoly je definovať základné pojmy a charakteristiky systémov na báze ČSS, analyzovať vplyvy CMOS VLSI technológie na tieto procesory a klasifikovať technické prostriedky pre stavbu procesorov ČSS.

2.1 ZÁKLADNÝ MODEL ČÍSLICOVÉHO SPRACOVANIA SIGNÁLOV

Aj keď systémy ČSS môžu mať rôznorodé stvárnenie, model znázornený na obr. 2.1 je dostatočne všeobecný na opísanie širokej triedy algoritmov ČSS.



Obr. 2.1 Model číslicového spracovania signálov

Analógové signály[1] $x^i(t),\ \ i=1,2,\ldots,P_{vst}$ vstupujú do *obmedzovacích filtrov*, ktoré predstavujú analógové dolné priepusty s medznou frekvenciou F_m^i a ich úlohou je obmedziť frekvenčný rozsah spracovávaných analógových signálov tak, aby po ich vzorkovaní nedošlo k prekrytiu spektra (tzv. aliasing). Filtrované signály sú v blokoch

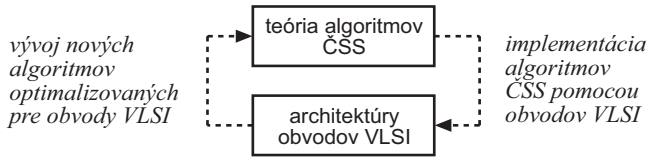
[1] Počet vstupných analógových signálov je závislý na konkrétnej aplikácii a môže byť aj nulový (napr. v prípade digitálnej syntézy).

analógovo-číslicových (A/D) *prevodníkov* konvertované na číslicové vzorky $x^i[n]$. Rýchlosť vzorkovania je riadená frekvenciou vzorkovania F_{vz}^i, ktorá musí podľa Shanon-Koteľnikovho teorému splňovať podmienku:

$$F_{vz}^i > 2F_m^i, \qquad i = 1,\ldots,P_{vst} \tag{2.1}$$

Číslicové vzorky vstupujú do *procesora* ČSS, ktorý realizuje algoritmus číslicového spracovania signálov, ktorý môže byť lineárny alebo nelineárny, časovo invariantný alebo závislý od času a generuje výstupné číslicové vzorky $y^j[n], \quad j = 1,\ldots,P_{vyst}$. Procesor ČSS je možné definovať ako *číslicový procesor*, ktorý dokáže vykonávať všetky matematické operácie využívané realizovaným algoritmom ako napr. sčítania, odčítania, násobenia a pod. V prípade potreby[2] je možné konvertovať číslicové signály $y^j[n]$ na zodpovedajúce analógové signály $y^j(t)$ pomocou *číslicovo-analógových* (D/A) *prevodníkov* a *rekonštrukčných filtrov*, ktorých úlohou je potlačiť zrkadlové kmitočty nad medznými frekvenciami jednotlivých kanálov.

V teórii algoritmov ČSS, napr. pri tvorbe nových algoritmov ČSS je často možné pracovať s procesorom ČSS ako s abstraktným číslicovým procesorom. Typickým príkladom je algoritmus výpočtu rýchlej Fourierovej transformácie (FFT), ktorý bol v podstate objavením rýchleho matematického algoritmu na výpočet diskrétnej Fourierovej transformácie (DFT). Praktické nasadenie metód ČSS však výrazne ovplyvňujú praktické možnosti reálnych technických prostriedkov. Pre súčasné obdobie je dokonca charakteristická vzájomná súvislosť teórie algoritmov ČSS a architektúr VLSI obvodov, ktorá je znázornená na obr. 2.2 .



Obr. 2.2 Vzájomné ovplyvňovanie teórie algoritmov ČSS a architektúr obvodov VLSI

Táto vzájomná súvislosť sa prejavuje napr. tvorbou nových algoritmov ČSS, ktoré pre dostupné technické prostriedky minimalizujú počet matematických operácií a prístupov do pamäti. Na druhej strane požiadavky technológie VLSI vytvorili aj nové smery v oblasti teórie algoritmov ČSS. Typickým predstaviteľom je napr. vývoj algoritmov pre systolické polia. Základné vlastnosti reálnych systémov ČSS, ktoré umožňujú ich široké praktické využitie ako aj ich základné charakteristiky sú uvedené v nasledujúcej časti.

2.2 VLASTNOSTI A CHARAKTERISTIKY REÁLNYCH SYSTÉMOV ČSS

Kľúčovým predpokladom pre úspešné (masové) nasadenie systémov na báze ČSS je poskytnutie vlastností, ktoré nie je možné dosiahnuť s použitím analógových technológií, resp. poskytnutím riešenia, ktorého cena je nižšia ako cena porovnateľného analógového riešenia. Medzi základné výhodné vlastnosti systémov na báze ČSS patria:

[2] Existujú aj systémy, ktoré využívajú len číslicový výstup (napr. systém rozpoznávania reči s textovým výstupom).

- ***Stabilita a necitlivosť na vplyv prostredia***. Číslicové systémy sú principiálne menej citlivé na vplyv prostredia, čo sa plne prejavuje aj v systémoch ČSS. Naviac využitie metód ČSS umožňuje znížiť vplyv prostredia aj na klasicky analógové bloky systému ČSS ako sú obmedzovacie a rekonštrukčné filtre a A/D a D/A prevodníky.
- ***Predikovateľnosť a opakovateľnosť***. Použitím systémov ČSS je možné eliminovať vplyv tolerancií súčiastok na cieľovú funkciu zariadenia. Najjednoduchším príkladom sú rôzne typy frekvenčne selektívnych filtrov, ktorých analógová implementácia je extrémne náročná.
- ***(Re)programovateľnosť***. Niektoré konštrukčné riešenia procesorov ČSS umožňujú definovanie, resp. zmenu funkcie systému ČSS zmenou *programu*. Táto vlastnosť má významný vplyv na efektívne využitie systémov ČSS. Je napr. možné využiť jeden systém ČSS na implementáciu značne odlišných zariadení, ktoré v optimálnom prípade vyžadujú len zmenu programu. Táto vlastnosť sa ukazuje ako *komerčne veľmi výhodná* a umožňuje napr. distribuovať zariadenia na báze ČSS k zákazníkom ešte pred prijatím noriem, podľa ktorých majú tieto zariadenia pracovať. Po prijatí noriem je možné realizovať potrebné zmeny priamo u zákazníka (alebo samotným zákazníkom). V súčasnosti, keď je doba prijímania nových (napr. telekomunikačných) štandardov rýchlejšia, než doba životnosti zariadení, je táto možnosť veľmi výhodná.
- ***Nízky príkon***. Pokrok v technológii VLSI umožňuje realizovať vzrastajúci počet systémov na báze ČSS, ktoré poskytujú minimálne rovnaké funkčné možnosti ako porovnateľné analógové systémy, pri podstatnom znížení príkonu. Táto vlastnosť umožňuje napr. u prenosných zariadení predlžovať napr. dobu medzi výmenou batérií a tým zvyšuje ich *úžitkovú hodnotu*. Z globálneho hľadiska nezanedbateľným efektom je aj *úspora energie*.
- ***Spoľahlivosť***. Zvyšovanie hustoty integrácie číslicových obvodov umožňuje znižovanie počtu súčiastok (integrovaných obvodov), ktoré sú potrebné na realizáciu cieľového zariadenia. Cieľom (v mnohých prípadoch realistickým) je realizovať *jednočipove riešenie* celého systému, čo zvyšuje spoľahlivosť výsledného zariadenia.
- ***Nízka cena***. Všeobecný trend znižovania ceny číslicových obvodov umožňuje zvyšovanie úžitkovej hodnoty zariadení na báze ČSS pri zachovaní ceny a často ju dokonca umožňuje znižovať. Podstatný vplyv tu zohráva aj znižovanie výrobných nákladov využívaním univerzálnych (často reprogramovateľných) súčiastok.
- ***Nové algoritmy***. Algoritmy ČSS umožňujú realizovať aj zariadenia, ktoré nemajú analógový ekvivalent. Ako príklad je možné uviesť filtre s konečnou impulzovou odpoveďou (FIR – Finite Impulse Response), metódy kompresie rečových, audio a video signálov, prípadne algoritmy z oblasti zabezpečovacích kanálových kódov.

Existuje niekoľko základných charakteristík, ktoré sú spoločné pre všetky systémy ČSS:

a) Algoritmus

Systémy ČSS sú často charakterizované použitým *algoritmom*. Algoritmus špecifikuje vykonávané aritmetické operácie, často však nešpecifikuje ako sú tieto operácie implementované. Tieto môžu byť implementované napr. pomocou štandardných mikroprocesorov, programovateľných signálových procesorov, alebo pomocou zákazníckych obvodov. Výber implementačnej technológie je do značnej miery určovaný potrebnou rýchlosťou aritmetických operácií a potrebnou *aritmetickou presnosťou*.

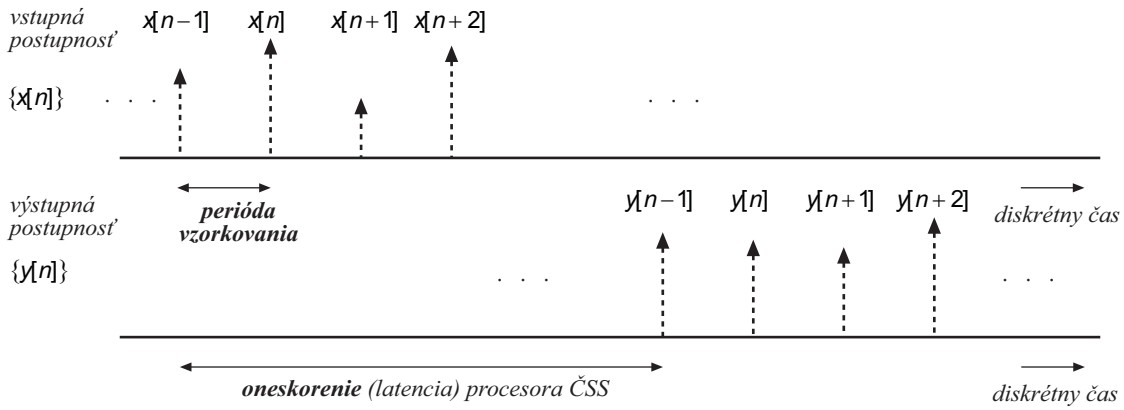
Medzi základné algoritmy ČSS s významným praktickým využitím patria:

- číslicová filtrácia,
- rýchla Fourierova transformácia,
- diskrétna konvolúcia a korelácia.

Teória týchto základných algoritmov je dobre prepracovaná a existujú efektívne postupy pre ich výpočet. Oblasť algoritmov ČSS je v súčasnosti veľmi rozsiahla a výrazne presahuje rámec tejto práce. Analýza niektorých vybraných algoritmov ČSS z oblasti spektrálnej analýzy a ekvalizácie prenosového kanálu a ich implementácia pomocou programovateľných signálových procesoroch je uvedená v kapitole 6.

b) Frekvencia vzorkovania

Jednou z kľúčových charakteristík systémov ČSS je *frekvencia vzorkovania* – t.j. rýchlosť s akou prichádzajú vstupné vzorky, resp. s akou sú generované výstupné vzorky. Spolu so zložitosťou implementovaného algoritmu, určuje frekvencia vzorkovania rýchlosť potrebnej implementačnej technológie. Na obr. 2.3 sú znázornené vstupné a výstupné vzorky procesora ČSS s jedným vstupom $x[n]$ a jedným výstupom $y[n]$.



Obr. 2.3 Frekvencia vzorkovania a oneskorenie procesora ČSS

Časový interval medzi príchodom za sebou idúcich vzoriek vstupného signálu sa nazýva *perióda vzorkovania* T_{vz}, pričom pre frekvenciu vzorkovania F_{vz} platí vzťah

$$F_{vz} = \frac{1}{T_{vz}} \tag{2.2}$$

Časový interval medzi príchodom vstupnej vzorky a zodpovedajúcou výstupnou vzorkou je definovaný ako *výpočtové oneskorenie* (computational latency). V typických aplikáciách je minimálna dosiahnuteľná perióda vzorkovania zvyčajne dôležitejším faktorom ako oneskorenie procesora ČSS. Samozrejme systém ČSS môže využívať (a v praxi aj veľmi často využíva) viac ako jednu frekvenciu vzorkovania a takýto systém sa nazýva *viacrýchlostný* (multirate) systém ČSS.

Z hľadiska praktických aplikácií je často veľmi dôležité, aby spracovanie signálov bolo realizované v *reálnom čase*. Tento pojem vyjadruje takú činnosť procesora ČSS, pri ktorej sa z postupnosti vstupných vzoriek spracuje každá vzorka. Táto definícia má rôzne dôsledky pre *rekurzívne* a *blokové algoritmy* výpočtu.

Medzi rekurzívne algoritmy výpočtu patria algoritmy, ktoré pracujú spôsobom:

- čítanie vstupnej vzorky $x[n]$ z A/D prevodníka,
- výpočet výstupnej vzorky $y[n]$ v procesore ČSS,
- zápis výstupnej vzorky $y[n]$ do D/A prevodníka,

a ich typickým predstaviteľom je napr. číslicová filtrácia[3]. Pokiaľ trvá realizácia výpočtu v procesore ČSS kratšiu dobu, než je perióda vzorkovania T_{vz}, potom je výpočet realizovaný v reálnom čase. V opačnom prípade sa niektoré vzorky vstupnej postupnosti nespracujú, sú vynechané a systém nepracuje v reálnom čase.

Blokové algoritmy, ako vyplýva z názvu, spracovávajú vstupné vzorky po blokoch a nie jednotlivo. Typickým predstaviteľom je napr. algoritmus FFT, ktorý zahrňuje v blokovom intervale $[b]$ tri operácie:

- zozbieranie bloku $[b]$ s dĺžkou N vstupných vzoriek,
- výpočet FFT z predchádzajúceho bloku $[b-1]$,
- výstup výsledkov výpočtu FFT z bloku $[b-2]$.

Pokiaľ trvá výpočet jedného bloku s N vstupnými vzorkami dobu kratšiu, než je jeho načítanie, je výpočet realizovaný v reálnom čase.

Rozsah frekvencií vzorkovania, ktoré sa používajú v systémoch ČSS je obrovský a presahuje 12 rádov. Samozrejme algoritmy pre najvyššie (aktuálne realizovateľné) frekvencie vzorkovania patria z hľadiska zložitosti medzi najjednoduchšie, pretože cena a zložitosť technickej realizácie s rastúcou frekvenciou vzorkovania prudko narastajú. Požiadavka po stále vyššej výpočtovej výkonnosti procesorov ČSS je jedným z faktorov, ktoré vytvárajú tlak na okamžité využívanie pokroku v technológii obvodov VLSI na zvyšovanie výkonnosti procesorov ČSS.

c) Hodinová (taktovacia) frekvencia

Číslicové elektronické systémy sú charakterizované ich hodinovou frekvenciou, ktorá obvykle vyjadruje rýchlosť, s akou číslicový systém realizuje najzákladnejšiu operáciu v systéme. Pre systémy ČSS pomer

$$P_{frek} = \frac{hodinová\ \ frekvencia\ systému}{vzorkovacia\ \ frekvencia} \tag{2.3}$$

charakterizuje, ako je možné systém ČSS implementovať. Táto hodnota charakterizuje predovšetkým množstvo a typ technických prostriedkov potrebných na realizáciu algoritmu ČSS s určitou zložitosťou, ktorý by pracoval v reálnom čase. Pre hodnoty $P_{frek} \to 1$ je jedinou možnosťou využitie paralelizmu technických prostriedkov. Naopak pre hodnoty $P_{frek} >> 1$ je možné využiť napr. štandardné programovateľné číslicové procesory.

d) Číslicová reprezentácia

Jadrom algoritmov ČSS sú predovšetkým operácie sčítania, odčítania a násobenia. Veľké množstvo algoritmov je možné efektívne realizovať pomocou tzv. MAC (Multiply and ACcumulate) operácie, ktorá realizuje matematickú operáciu [3]:

[3] Číslicovú filtráciu je však možné realizovať aj blokovými algoritmami napr. s využitím rýchlej konvolúcie, prípadne využitím špeciálnych paralelných štruktúr.

$$A = B + C * D \tag{2.4}$$

Medzi základné číslicové reprezentácie v oblasti procesorov ČSS patrí reprezentácia *v pevnej rádovej čiarke* (fixed point representation), pričom v oblasti ČSS je preferovaná jej tzv. *zlomková reprezentácia* (fractional representation) v tvare M bitov (dĺžka slova) $b_m \in \{0,1\}, \quad m = 0,1,2,\ldots,M-1$, ktorá vyjadrujú číslo $x_{pevná} \in \langle -1,1)$ v tvare

$$x_{pevná} = (-1)b_0 + \sum_{m=1}^{M-1} b_m 2^{-m} \tag{2.5}$$

ktorý reprezentuje záporné čísla v druhom doplnku. Čísla mimo intervalu $\langle -1,1)$ nie je možné v zlomkovej reprezentácií reprezentovať a prípadné výsledky mimo intervalu $\langle -1,1)$ sú v procesoroch ČSS najčastejšie *obmedzené* (saturated), t.j. nadobúdajú najväčšiu kladnú $(1-2^{-M})$, alebo najmenšiu zápornú hodnotu (-1). Menej časté je potlačenie dodatočných bitov, ktoré vzniknú pri pretečení aritmetických operácií (wrap around mode), ktoré je typické pre systémy ČSS na báze štandardných univerzálnych procesorov.

Reprezentácia v *pohyblivej rádovej čiarke* (floating point representation) výrazne zvyšuje dynamický rozsah (t.j. pomer medzi najväčším a najmenším reprezentovateľným číslom) použitím mantisy a exponentu a vyjadrením čísla v tvare

$$x_{pohyblivá} = mantisa \times 2^{\exp onent} \tag{2.6}$$

Tento formát bol vzhľadom na široké využitie štandardizovaný v norme IEEE 754 [4]. Táto reprezentácia výrazne znižuje pravdepodobnosť *pretečení*, *podtečení* a *potrebu zmeny mierky*, ktoré sú typické pre reprezentáciu v pevnej rádovej čiarke, čo sa odráža v zjednodušení návrhu algoritmov a ich realizácii pomocou programovateľných procesorov ČSS. Nevýhodou tejto reprezentácie je zložitejšia konštrukcia aritmetickej jednotky. V praktických aplikáciách je často možné využiť reprezentáciu v tzv. *blokovej pohyblivej čiarke*, ktorá využíva spoločný exponent pre blok (vektor) dát a využíva výhodné vlastnosti obidvoch predchádzajúcich systémov [5].

Okrem týchto klasických reprezentácií sa v procesoroch ČSS často využívajú aj menej tradičné číselné reprezentácie a aritmetiky a to predovšetkým

- distribuovaná aritmetika (DA – Distributed Arithmetic),
- číselný systém zvyškových tried (RNS – Residue Number System),
- aritmetika v konečných poliach (FFA – Finite Field Arithmetic).

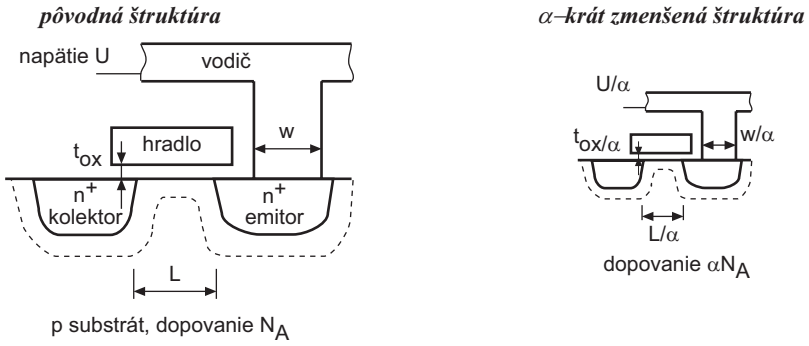
Podstatou DA [6] a RNS [7] je odstránenie využitia zložitej paralelnej násobičky a možnosť využitia paralelizmu, ktoré tieto netradičné aritmetiky poskytujú. Tieto prístupy nachádzajú uplatnenie predovšetkým v oblasti špecializovaných jednoúčelových procesorov ČSS a umožňujú dosahovať rádovo vyššie rýchlosti ako klasické aritmetiky, ktoré využívajú zložité paralelné násobičky, pri relatívne malej zložitosti a ploche čipu procesora ČSS. Ukazuje sa, že DA a RNS je možné s úspechom uplatniť napr. v procesoroch ČSS na báze (programovateľných) hradlových polí.

Uplatnenie FFA v procesoroch ČSS je v súčasnosti aktuálne predovšetkým pri implementácii blokových zabezpečovacích kódov [8] ako aj vybraných kryptografických algoritmov, pričom je však možné realizovať v tejto aritmetike aj klasické algoritmy ČSS [9], [10]. Tieto typy algoritmov sa v súčasných, predovšetkým telekomunikačných aplikáciách široko využívajú.

2.3 TECHNOLÓGIA VLSI A JEJ VPLYV NA SYSTÉMY ČSS

Kremíková technológia CMOS sa počas uplynulých 25 rokov stala dominantnou technológiou mikroelektronického priemyslu. Úroveň dosiahnutej integrácie je všeobecne známa pod skratkou VLSI a vzhľadom na neustály rast úrovne integrácie sa stále častejšie používa označenie ULSI (Ultra Large Scale Integration) [11], [12], predovšetkým v súvislosti prechodu do sub-mikrónovej oblasti. Trendy vo vývoji tejto technológie majú zásadný vplyv aj na parametre a architektúry systémov ČSS. Analýzy a prognózy vývoja VLSI technológie umožňujú pochopiť a zdôvodniť mnohé trendy v oblasti monolitických systémov ČSS.

Exponenciálny nárast hustoty integrácie VLSI obvodov (známy ako Moorov zákon) vychádza z tzv. princípu *zmeny mierky* (principle of scaling), t.j. faktu, že bolo (a stále je) možné zmenšovať rozmery štruktúr MOSFET tranzistora, ktorý je základným stavebným prvkom CMOS obvodov. Princíp zmeny mierky je znázornený na obr. 2.4 [1], [13].



Obr. 2.4 Princíp zmeny mierky kremíkovej technológie o faktor α

Zmenšením rozmerov štruktúry vľavo o faktor $\alpha > 1$ je možné získať zmenšený prvok. Pokiaľ sa o faktor α zmenší aj úroveň napätia a zvýši úroveň dopovania na hodnotu αN_A bude elektrostatické pole v obidvoch štruktúrach rovnaké. Plocha štruktúry sa takto zmenší o faktor α^{-2} a na rovnakú plochu čipu je možné integrovať α^2-viac tranzistorov, pričom pri zachovaní rovnakej *intenzity elektrického poľa* je v obidvoch štruktúrach rovnaká výkonová hustota a teda celkový príkon čipu sa nezmení, t.j. čip s α^2 krát väčším počtom tranzistorov vyžaduje ten istý príkon. Naviac oneskorenie menšej štruktúry je α-krát menšie.

Tento optimistický model v praxi obmedzujú viaceré faktory, z ktorých najzávažnejší je, že *prahové napätie* sa so zmenou mierky nemení (je závislé na šírke zakázaného pásma použitého polovodiča) a spôsobuje nárast zvyškových prúdov, čo sa začína výrazne prejavovať pokiaľ sa napájacie napätie U/α blíži k hodnote 1 V. Toto obmedzenie sa v praxi obchádza tzv. *selektívnou zmenou mierky*, ktorá zabezpečí zmenšenie napájacieho napätia v inom (menšom) pomere ako rozmery štruktúr. Nevýhodou tohto prístupu je zvýšenie intenzity elektrostatického poľa v čipe, ktoré má za následok zníženie spoľahlivosti polovodičového čipu. Principiálne je teda nevyhnutné znižovať napájacie napätie obvodov VLSI aj v prípadoch, keď otázka príkonu nie je kritická (aby sa zabezpečila spoľahlivosť VLSI čipu).

Na základe princípu zmeny mierky a predchádzajúceho vývoja je možné extrapolovať ďalší vývoj VLSI technológie. Jedna z najznámejších prognóz, tzv. Semiconductor Industry Association Roadmap [14], je uvedená v tab. 2.1 [1].

Tab. 2.1 Prognózy vývoja VLSI CMOS technológie

Rok uvedenia	1995	1997	1999	2001	2003	2006	2009	2012
DRAM (bity/čip)	64 M	256 M	1 G		4 G	16 G	64 G	256 G
Veľkosť čipov DRAM (mm^2)	190	280	400	445	560	790	1120	1580
μP (tranzistory/cm^2)		3,7 M	6,2 M	10 M	18 M	39 M	84 M	180 M
μP – veľkosť čipu (mm^2)	250	300	340	385	430	520	620	750
Litografia (μm)	0,35	0,25	0,18	0,15	0,13	0,10	0,07	0,05
Hrúbka oxidu (nm)	7-12	4-5	3-4	2,4-3,2	2-3	1,5-2	<1,5	<1,0
Napájacie napätie (V)	3,3	1,8-2,5	1,5-1,8	1,2-1,5	1,2-1,5	0,9-1,2	0,6-0,9	0,5-0,6
Hodinová frekvencia (MHz)	300	750	1200	1400	1600	2000	2500	3000

Na základe týchto údajov je možné očakávať v oblasti procesorov ČSS na báze technológie VLSI tieto trendy:

a) Výkonnosť

Ďalšie zvyšovanie výpočtovej výkonnosti procesorov ČSS bude realizované predovšetkým rozsiahlejším využívaním *paralelizmu*, čo je spôsobené rýchlejším nárastom hustoty integrácie oproti nárastu hodinovej frekvencie obvodov VLSI.

b) Príkon

Príkon procesorov ČSS sa bude znižovať, čo sa prejaví zníženou úrovňou napájacieho napätia vo všeobecnosti a samozrejme aj celkového príkonu potrebného na implementáciu konkrétneho algoritmu. Rýchlosť obvodov bude taktiež kontinuálne narastať. Tento trend bude možné pozorovať v obidvoch, z pohľadu optimalizácie CMOS VLSI obvodov (dosiahnutej rozdielnou selektívnou zmenou mierky) odlišných oblastiach:

- ***vysokovýkonné*** (high performance) súčiastky,
- ***nízkopríkonové*** (low power) súčiastky,

čo je znázornené na obr. 2.5 . Typickým predstaviteľom nízkopríkonových súčiastok CMOS sú práve monolitické DSP optimalizované pre telekomunikačné aplikácie a spotrebnú elektroniku. Do kategórie vysokovýkonných súčiastok patria napr. procesory pre personálne počítače.

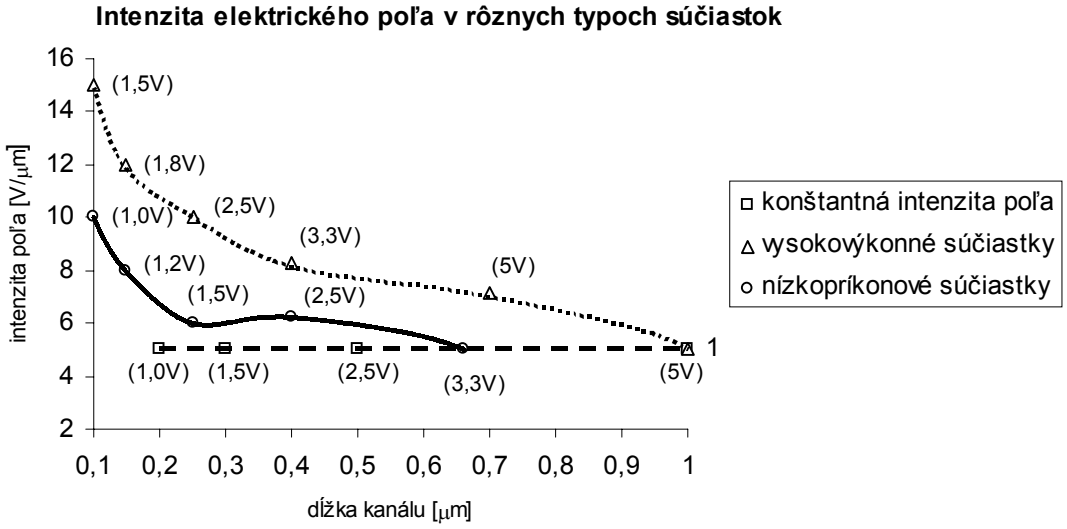
Na porovnanie rôznych procesorov ČSS z pohľadu príkonu sa veľmi často používa charakteristika

$$\frac{príkon}{operácia} \qquad \{\text{mW/MIPS, mW/MOPS, mW/MFOPS}\dots\} \tag{2.7}$$

prípadne vyjadrenie spotreby prúdu na operáciu pri udanej hodnote napájacieho napätia, čo bude zapisované v tvare $mA/operáciu@napätie$. Skratky MIPS, MOPS, MFOPS znamenajú milióny inštrukcií, operácií resp. operácií v pohyblivej rádovej čiarke za sekundu. Táto charakteristika síce do značnej miery odráža úroveň použitej technológie zahrňuje však v sebe aj vplyv použitej architektúry a je preto vhodná predovšetkým na porovnanie procesorov ČSS s rovnakou architektúrou. Z pohľadu implementovaného algoritmu je často používaná charakteristika

$$\frac{príkon}{aloritmus} \qquad \{\text{mW/FFT, mW/CELP}\dots\} \tag{2.8}$$

pričom algoritmus musí byť presne špecifikovaný (napr. rozmer FFT, typ CELP kodéra a pod.).



Obr. 2.5 Nárast intenzity elektrického poľa, U / L, v závislosti na dĺžke kanálu pre rôzne typy súčiastok CMOS

c) Ekonomický aspekt

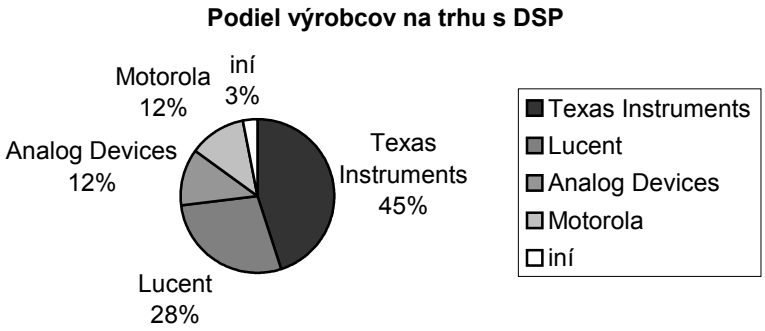
Analýza dlhodobých údajov ukazuje, že cena kremíkových čipov na jednotku plochy (samozrejme pri použití aktuálnej technológie) vyjadrená v $\$/cm^2$ mierne narastá a dosahuje hodnoty 5 – 6 % za rok, čo zabezpečuje neustály pokles ceny čipov vykonávajúcich ekvivalentnú funkciu [15]. Na porovnanie ceny rôznych čipov sa používa charakteristika

$$\frac{operácia}{cena} \qquad \{\text{MIPS/\$, MOPS/\$, MFOPS/\$…}\} \tag{2.9}$$

Táto charakteristika, spolu s predchádzajúcimi umožňuje zhodnotiť vhodnosť konkrétnych technických prostriedkov pre využitie v konkrétnom zariadení a v praxi sa veľmi často využívajú.

Cenový aspekt má aj inú stránku, ktorá má vplyv na správanie sa výrobcov súčiastok VLSI vo všeobecnosti a uplatňujú sa v poslednej dobe aj u kľúčových výrobcov monolitických DSP. Zvládnutie masovej produkcie obvodov VLSI so špičkovou (v danom období) úrovňou technológie vyžaduje enormne *vysoké ekonomické náklady*. V oblasti polovodičového priemyslu je možné pozorovať *vytváranie strategických aliancií* medzi najväčšími výrobcami polovodičových súčiastok, ktorých motív je ekonomický. V oblasti výroby programovateľných signálových procesorov je napr. možné poukázať na vytvorenie aliancie medzi firmami Motorola a Lucent Technology (ďalej len Lucent - predtým časť AT&T) ako aj Analog Devices a Intel, pričom v týchto prípadoch ide o jedných z najväčších výrobcov integrovaných obvodov. Ich podiel na trhu programovateľných jednočipových signálových procesorov je dokumentovaný na obr. 2.6 [80]. Od takýchto aliancií je možné očakávať moderné riešenia (a to nie len z hľadiska úrovne integrácie), ale aj v oblasti vytvorenia nových architektúr resp. filozofie programovateľných signálových

procesorov. Najnovšie výsledky tohto vývoja s dôrazom na uvedených výrobcov budú uvedené v kapitole 4.



Obr. 2.6 Podiel jednotlivých výrobcov na trhu programovateľných signálových procesorov

d) Integrácia

Exponenciálny nárast v počte integrovaných prvkov v čipoch VLSI umožní realizovať kompletné systémy ČSS na jednom monolitickom čipe, pričom takéto jednočipové systémy budú obsahovať na čipe aj A/D a D/A prevodníky a vstupno-výstupné senzory. V ďalšej fáze budú do monolitickej formy integrované kompletné systémy, ktoré budú využívať systémy ČSS na zvýšenie úžitkovej hodnoty a tým aj zvýšenie ich podielu na trhu. Ako možno trochu futuristické aplikácie, ale z hľadiska úrovne integrácie a príkonu v budúcnosti realistické (pokiaľ sa extrapolované údaje vývoja VLSI technológie zrealizujú), je možné uviesť systémy na preklad hovorenej reči v reálnom čase, prípadne videokonferenčné systémy integrované do veľkosti náramkových hodiniek [1]. Aspekt masovosti je pre rozvoj technológie VLSI extrémne dôležitý, pretože veľké výrobné náklady je nutné amortizovať do veľkého počtu vyrobených čipov. Takúto masovosť nasadenia môžu zabezpečiť predovšetkým *spotrebná elektronika*, *počítačová* a *telekomunikačná technika*, pričom je možné očakávať zvyšujúci sa podiel prenosných zariadení, ktoré budú založené na nízkopríkonových modifikáciách VLSI technológie.

2.4 KLASIFIKÁCIA TECHNICKÝCH PROSTRIEDKOV PRE ČSS

Implementačná báza systémov pre ČSS je veľmi široká a ponúka možnosti využitia technických prostriedkov v rôznych kvalitatívnych, kvantitatívnych a cenových oblastiach. Neexistujú jednoznačné kritéria pre klasifikáciu procesorov ČSS. Tieto by sa dali klasifikovať napr. podľa stupňa integrácie použitých technických prostriedkov, konštrukcie aritmetickej jednotky, použitej aritmetiky, výkonnosti, ceny, príkonu a pod. Žiadne z týchto hľadísk však nevystihuje to podstatné, t.j. spôsob riadenia procesora ČSS [16]. Spôsob riadenia určuje predovšetkým flexibilitu procesora ČSS a ukazuje sa ako dostatočne univerzálne hľadisko pre klasifikáciu procesorov ČSS. Podľa spôsobu riadenia sa procesory ČSS delia do troch skupín:

1. Procesory s pevnou logikou

Do tejto skupiny patria procesory ČSS u ktorých je riadenie algoritmu spracovania signálov realizované *pevným obvodovým zapojením* jednotlivých stavebných prvkov. Podľa technologickej úrovne môžu tieto prvky predstavovať diskrétne obvody na báze technológie LSI alebo VLSI, jednočipové VLSI procesory, prípadne štandardné bunky obvodov ASIC. Nevýhodou procesorov tejto skupiny je *malá flexibilita*, pretože zmena algoritmu sa realizuje zmenou obvodového zapojenia. Ich výhodou je však vysoká rýchlosť a relatívna jednoduchosť, pretože obsahujú iba prvky nevyhnutné pre realizáciu vybraného algoritmu ČSS. Do tejto skupiny patria špecializované, jednoúčelové zákaznícke a polozákaznícke procesory ako napr. FFT procesory, FIR filtre, MPEG dekodéry, modemové čipové sady a pod.

2. Programovateľné procesory

Tieto procesory majú program realizovaného algoritmu ČSS uložený *v reprogramovateľnej inštrukčnej pamäti* a sú *značne flexibilné*. Zmena implementovaného algoritmu ČSS sa realizuje preprogramovaním inštrukčnej pamäte. Programovateľné procesory sa delia do troch skupín:

- ***Procesory z diskrétnych funkčných blokov***. Medzi diskrétne funkčné bloky patria násobičky, sčítačky, mikroprocesorové rezy, viacportové pamäte, radiče programov a pod. Vzhľadom na pokrok technológie VLSI sú tieto procesory využívané len vo veľmi špeciálnych aplikáciách, kde je požadovaná vysoká rýchlosť, pričom otázka ceny a príkonu je druhoradá.
- ***Jednočipové signálové procesory***. Obsahujú na čipe všetky obvody potrebné na implementáciu veľkej triedy algoritmov ČSS. Procesory tejto skupiny sú vzhľadom na ich univerzálnosť značne rozšírené. Ich relatívne vysoká rýchlosť je dosiahnutá pomocou *optimalizovanej architektúry* a *optimalizovanej redukovanej inštrukčnej sady*. Vlastností a možnosti využitia jednočipových signálových procesorov sú podrobne analyzované v ďalších častiach práce.
- ***Univerzálne procesory***. Vzhľadom na svoju univerzálnosť môžu tieto procesory realizovať jednoduchšie algoritmy ČSS (pojem jednoduchší algoritmus sa s technologickým pokrokom samozrejme mení). Výkonnosti jednočipových signálových procesorov môžu konkurovať len pri rádovo vyššom príkone a vyššej cene.

3. Iné procesory

Do tejto skupiny je možné zaradiť procesory ČSS v ktorých je vykonávanie algoritmov vykonávané netradičným spôsobom. Patria sem napr. procesory riadené tokom dát (data flow processors), systolické, celoparalelné a neurónové procesory, ktoré dosahujú vysokú rýchlosť spracovania predovšetkým s využitím masívneho paralelizmu. S pokrokom

technológie VLSI je možné očakávať aj implementáciu tejto skupiny procesorov v monolitickej forme, pričom už v súčasnosti existujú špecializované čipy využívajúce niektoré z uvedených princípov. Pre širšie nasadenie týchto systémov bude veľmi dôležité *umožnenie programovateľnosti* týchto procesorov, prípadne zahrnutie týchto procesorov ako špecializovaných koprocesorov v rámci moderných programovateľných procesorov.

2.5 ZHRNUTIE

Vzhľadom na fakty uvedené v tejto kapitole je možné skonštatovať, že programovateľné monolitické jednočipové signálové procesory (v ďalšej časti bude na označenie tejto triedy procesorov používaná skratka **DSP**) budú v budúcnosti nachádzať stále širšie využitie predovšetkým v telekomunikačných [17], [18], [19], [20] a multimediálnych aplikáciách [21] a ich úlohu v systémoch ČSS je možné porovnať s úlohou mikroprocesorov v klasických číslicových systémoch. Samozrejme DSP nepokryjú celý segment systémov ČSS, kde predovšetkým v oblasti vysokovýkonných systémov ČSS sa budú uplatňovať procesory s pevnou logikou (predovšetkým na báze technológie ASIC), prípadne iné procesory.

Technológia použitá v oblasti DSP však má veľmi úzku súvislosť aj s obvodmi ASIC a univerzálnymi mikroprocesormi a mikropočítačmi [22]. Už s využitím súčasnej technológie VLSI je možné na báze polozákaznických obvodov ASIC na báze štandardných buniek umiestňovať na čipy ASIC celé jadrá DSP. Podobne mnohé obvody ASIC sú v podstate maskou programovateľné DSP a existujú dokonca firmy, ktoré sa špecializujú len na vývoj DSP, ktoré predávajú ako jadra DSP, prípadne predávajú ASIC obvody na báze týchto jadier.

V oblasti univerzálnych mikroprocesorov sa vplyv DSP technológie prejavuje predovšetkým v multimediálnych rozšíreniach mikroprocesorov pre oblasť počítačovej techniky ako aj vznikom novej triedy výkonných DSP, *multimediálnych procesorov* (media processors).

Paralelizmus je naopak oblasťou, ktorá má vplyv na samotné DSP a mnohé princípy z oblasti paralelných počítačových systémov sa začínajú uplatňovať aj v DSP.

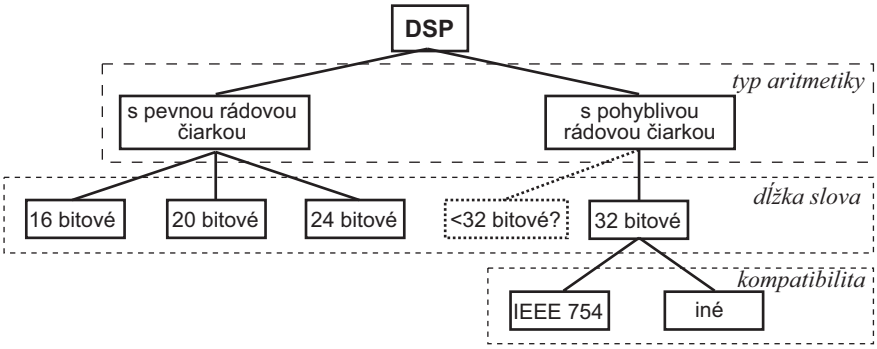
Ďalšie kapitoly práce sa pokúsia analyzovať vývoj DSP pre nízkopríkonové aplikácie, opísať základné stavebné bloky týchto DSP a na základe dostupných informácii naznačiť trendy vývoja tejto skupiny DSP ako z pohľadu vývoja architektúr, tak aj z pohľadu vývoja programových prostriedkov.

3 KLASICKÉ SIGNÁLOVÉ PROCESORY

Na určitom stupni vývoja integrácie polovodičovej technológie (na konci 70-tych rokov) bolo možné integrovať do monolitického čipu všetky základné stavebné bloky procesora ČSS a vznikol programovateľný monolitický signálový procesor – DSP, ktorého charakteristické vlastnosti:

- špecializovaná architektúra,
- krátky inštrukčný cyklus,
- malý ale výkonný inštrukčný súbor,
- nízka spotreba,
- zbernicová orientácia,
- malé rozmery,
- vysoká spoľahlivosť,

mu získali veľmi rýchlo významnú pozíciu v oblasti technických prostriedkov pre ČSS. Architektúra DSP sa síce postupne vylepšovala a prenikla aj do architektúr štandardných procesorov [78], približne do polovice 90-tych rokov sa však stále používali DSP s jednou aritmetickou jednotkou. Takéto DSP budú označované v ďalšom texte ako *klasické* DSP, prípadne len DSP. Z hľadiska použitej aritmetiky, ktorá výrazne ovplyvňuje vhodnosť využitia DSP pre konkrétnu aplikáciu je možné komerčne dostupné DSP rozdeliť do hierarchickej štruktúry zobrazenej na obr. 3.1 [23].



Obr. 3.1 Rozdelenie komerčných DSP na základe numerickej reprezentácie

V tejto kapitole je opísaný historický vývoj, základné stavebné bloky a modifikácie architektúr klasických DSP s orientáciou na segment DSP s pevnou rádovou čiarkou, ktoré sú dominantné pre aplikácie v telekomunikačnej technike a spotrebnej elektronike. Súčasťou tejto kapitoly je aj uvedenie základných pojmov a princípov využívaných v architektúrach DSP, ktoré budú využívané v ďalších častiach práce.

3.1 HISTORICKÝ VÝVOJ

DSP boli logickým vyústením dvoch trendov v oblasti procesorov ČSS, ktoré boli využívané koncom 70-tych rokov. Prvý trend bol vývoj špeciálnych jednoúčelových jednočipových procesorov ČSS určených pre špecifické aplikácie. Druhý trend bol vývoj multičipových systémov ČSS predovšetkým pre vysokovýkonné vojenské aplikácie, ktoré boli zvyčajne založené na technológii bitových rezov. Tieto dva trendy spolu s pokrokom v oblasti mikroprocesorovej techniky viedli k vytvoreniu DSP.

Prvý DSP – AMI S2811 bol ohlásený v roku 1978, ale nebol komerčne dostupný počas niekoľkých nasledujúcich rokov vzhľadom na problémy s VMOS technológiou [29]. Prvým komerčne dostupným DSP sa tak stal v roku 1979 procesor Intel i2920. Koncepčne zaujímavou vlastnosťou na tomto procesore bola integrácia jedného vstupného 9-bitového A/D a jedného výstupného D/A prevodníka spolu so vstupnými a výstupnými analógovými multiplexormi pre 4 vstupné a 8 výstupných kanálov, ktoré umožňovali principiálnu realizáciu viackanálového spracovania analógových signálov na obr. 2.1 v jednočipovom vyhotovení. Novšie DSP s integrovanými A/D a D/A prevodníkmi (aj keď na kvalitatívne vyššej úrovni s využitím prevodníkov na báze *sigma-delta modulácie*) sa objavili až o niekoľko rokov neskôr. Intel i2920 umožňoval realizovať jednočipové riešenie kompletného systému ČSS pomocou čipu s 28 vývodmi. Procesor však nemal *hardvérovú násobičku*, ktorá sa stala typickým znakom všetkých novších DSP, čo sa odrazilo v jeho rýchlom nahradení novšími typmi DSP. Najväčším prínosom i2920 bolo, že ukázal na medzeru v technických prostriedkoch, ktorú rýchlo vyplnili iné firmy generáciou DSP, ktoré už mali architektúry podstatne lepšie prispôsobené k požiadavkám systémov ČSS. Zaujímavým faktom je tiež skutočnosť, že firma Intel po uvedení i2920 svoje aktivity v oblasti DSP na nasledujúcich 20 rokov zastavila.

Prvým komerčne dostupným DSP s hardvérovou násobičkou bol japonský NEC7720 uvedený na trh v roku 1980 [30]. Čip využíval *harvardskú architektúru*, ktorá sa stala ďalšou charakteristickou vlastnosťou klasických DSP, jeho ALU však ešte nepodporovala saturačnú aritmetiku. Procesor NEC7720 umožňoval vykonávanie inštrukcií len z vnútornej PROM (EPROM) pamäte čo bolo pre zložitejšie aplikácie značným obmedzením.

V roku 1980 firma AT&T vyvinula maskou programovateľný DSP1, vychádzajúci ešte z klasickej v*on Neumanovej architektúry*, ktorý však bol určený len pre interné účely.

Prvý DSP umožňujúci vykonávanie inštrukcií z externe pripojenej pamäte bol TMS32010 od firmy Texas Instruments (TI) vyrobený v roku 1982. Možnosť pripojenia externej programovej pamäte posunula DSP z hľadiska *programovacieho modelu* bližšie k univerzálnym mikroprocesorom. Táto vlastnosť spolu s veľkým dôrazom kladeným na vývojové prostriedky a optimalizované knižnice, ktorými je firma TI známa, viedli k masovému rozšíreniu DSP.

Vyššie uvedené DSP využívali aritmetiku v pevnej rádovej čiarke. Ďalším logickým krokom bolo využitie aritmetiky s pohyblivou rádovou čiarkou v procesore Hitachi HD61810 z roku 1982, ktorý využíval 12-bitovú mantisu a 4-bitový exponent [31]. Aj keď niektoré úvahy naznačujú, že využitie aritmetiky s pohyblivou rádovou čiarkou, ktorá by využívala skrátenú mantisu a exponent by mohlo byť zaujímavé [5], dĺžka slova v tejto triede DSP sa veľmi rýchlo ustálila na hodnote 32 bitov. V roku 1984 firma AT&T uviedla na externý trh svoj prvý komerčne dostupný DSP s pohyblivou rádovou čiarkou využívajúci 24-bitovú mantisu a 8-bitový exponent. Prvý DSP, ktorý podporoval aritmetiku IEEE 754 bol v roku 1987 procesor MB86232 od firmy Fujitsu.

V druhej polovici 80-tych rokov vstúpili na trh s DSP aj firmy Motorola a Analog Devices so svojimi DSP DSP56001 a ADSP2100.

Z hľadiska naznačeného historického vývoja sú DSP často klasifikované do *troch generácií* [32], [33], pričom vybraní predstavitelia jednotlivých generácií sú uvedení v tab. 3.1 [32], [33], pričom podrobnejšie informácie o týchto DSP je možné nájsť napr. v [32], [33], [34], [35], [36]. DSP z prvej a druhej generácie pracovali s pevnou rádovou čiarkou, pričom DSP z druhej generácie mali nižšiu spotrebu (využívali už technológiu CMOS), boli rýchlejšie a mali väčšiu kapacitu vnútorných a vonkajších pamätí. DSP z tretej generácie používali aritmetiku s pohyblivou rádovou čiarkou, často obsahovali interné radiče DMA a boli prispôsobené pre prácu v multiprocesorových systémoch.

Tab. 3.1 Klasifikácia vybraných DSP

Generácia	Výrobca - typ (rok oznámenia)			
1. generácia	INTEL i2920 (1979)	NEC μPD7720 (1980)	TI TMS32010 (1982)	NEC μPD7725 (1988)
2. generácia	TI TMS320C25 (1985)	NEC μPD77220 (1985)	AT&T DSP16A (1988)	MOTOROLA DSP56001 (1988)
3. generácia	NEC μPD77230 (1986)	TI TMS320C30 (1987)	AT&T DSP32A (1988)	MOTOROLA DSP96002 (1988)

Tento, v staršej literatúre veľmi často používaný spôsob klasifikácie DSP, však môže viesť k chybným záverom. Skutočne, koncom 80-tych rokov bol často uvádzaný názor, že s postupom úrovne integrácie obvodov VLSI budú DSP s pevnou rádovou čiarkou v mnohých praktických aplikáciách nahradené DSP z tretej generácie, t.j. procesormi s pohyblivou rádovou čiarkou. To malo predovšetkým uľahčiť tvorbu programov minimalizovaním problémov s pretečením. Tento predpoklad sa však nesplnil a DSP s pevnou rádovou čiarkou sa vyvíjajú rýchlejšie (minimálne z hľadiska počtu typov a vyrobených kusov) ako DSP s pohyblivou rádovou čiarkou. Napr. firma Motorola, jeden z popredných svetových výrobcov DSP po uvedení DSP 96002 [37] – svojho prvého DSP s pohyblivou rádovou čiarkou (bol dokonca jedným z prvých DSP, ktorý pracoval so štandardom IEEE 754) už počas ďalších desiatich rokov nevyrobila žiadny nový DSP s pohyblivou rádovú čiarkou. Počas tejto doby sústredila svoj výrobný a vývojový potenciál do oblasti DSP s pevnou rádovou čiarkou a vyvinula a vyrobila viac ako dve desiatky rôznych typov DSP tejto kategórie. Podobne aj ostatní svetoví výrobcovia DSP, ktorí tvoria tzv. veľkú štvorku (okrem Motoroly ešte Lucent, TI a Analog Devices) vyrábajú nepomerne viac kusov a typov DSP s pevnou rádovou čiarkou. Toto platí aj v súčasnosti viac ako 15 rokov po uvedení prvého DSP s pohyblivou rádovou čiarkou. Tento stav je možné *zdôvodniť požiadavkami trhu po stále lacnejších a energeticky efektívnejších DSP*, ktoré sa používajú v produktoch s veľkou sériovosťou. Menšia zložitosť a vo všeobecnosti *kratšia dĺžka slova* týchto procesorov, ktoré priamo vplývajú na *nižšiu cenu a príkon* DSP, hrajú dominantnú úlohu. Otázka vyšších nákladov na programové prostriedky, ktoré sa amortizujú do veľkého počtu výrobkov je až druhoradou otázkou (aj keď stále dôležitou).

DSP s pohyblivou rádovou čiarkou nachádzajú uplatnenie predovšetkým v oblasti špecializovaných vysokovýkonných paralelných systémov ČSS [38] ako aj pri nízkosériovej výrobe zariadení na báze ČSS, kde otázka nákladov na vývoj programových prostriedkov je kritickým faktorom.

Klasické DSP v súčasnosti vyrába viac ako desiatka svetových výrobcov [39], [40], pričom počet komerčne dostupných DSP presahuje číslo 100.

3.2 ARCHITEKTÚRY PROGRAMOVATEĽNÝCH DSP

Medzi najčastejšie používané algoritmy ČSS patria číslicová filtrácia a výpočet FFT. Základná architektúra DSP bola vytvorená s cieľom dosiahnuť vysokú rýchlosť implementácie práve pre tieto dva typy algoritmov. Najjednoduchším číslicovým filtrom je FIR filter, ktorý je možné opísať matematickým vzťahom

$$y[n] = \sum_{k=0}^{N-1} h_k x[n-k] \tag{3.1}$$

pričom N je rád FIR filtra, h_k sú koeficienty filtra a $x[n]$, $y[n]$ sú vstupné resp. výstupné vzorky FIR filtra v diskrétnom čase n. Je zrejmé, že FIR filter je možné efektívne implementovať pomocou operácie MAC (2.4). Podobne je možné pomocou tejto operácie a špeciálneho rozkladu implementovať aj algoritmus FFT.

Vzhľadom na potrebu maximálnej výpočtovej rýchlosti, prakticky všetky významné klasické DSP využívajú harvardskú architektúru s oddeleným spracovaním inštrukcií a dát, prípadne jej modifikácie, umožňujúce zvýšenie rýchlosti s využitím paralelizmu, ktoré táto architektúra poskytuje.

3.2.1 SÚBEŽNÉ SPRACOVANIE

Súbežné spracovanie (concurrent processing) je všeobecnou základnou metódou umožňujúcou výrazné zvýšenie výkonnosti technických prostriedkov a má v DSP dve základné formy – *paralelizmus* a *zreťazenie* [41], [42], ktoré sú schematicky znázornené na obr. 3.2 a charakterizované nasledujúcimi vlastnosťami:

a) Paralelizmus

Ako paralelný sa označuje systém, v ktorom môže prebiehať niekoľko procesov súčasne (paralelne)[4]. Dôvod na uplatnenie paralelizmu v číslicových systémoch je treba hľadať predovšetkým v snahe zvyšovať ich výkonnosť nad hranicu, ktorá je realizovateľná súčasnou úrovňou technológie VLSI obvodov. Vzhľadom na trendy vývoja technológie VLSI je možné očakávať prudký nárast využitia paralelizmu aj v DSP.

b) Zreťazenie

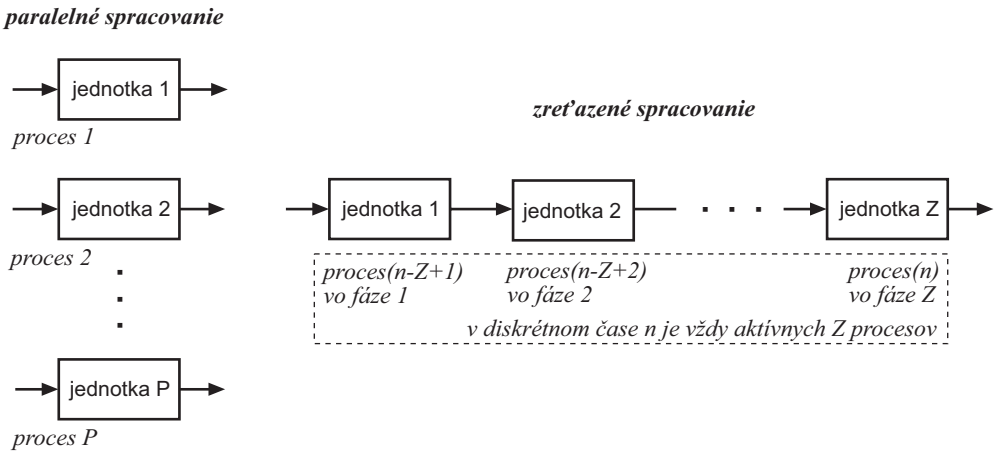
Spracovávanie informácie je obvykle realizované ako postupnosť jednoduchých úkonov – elementárnych operácií. Tieto úkony je možné zoradiť tak, aby na seba nadväzovali a vzájomne sa doplňovali v súlade s požadovaným algoritmom spracovania. Určitý operand[5] (inštrukcia alebo dáta) tak postupne prechádza postupnosťou transformácií, pričom každá operácia musí čakať na ukončenie všetkých operácií, ktoré ju predchádzajú. Pokiaľ sa jeden operand postupne spracováva v Z rôznych blokoch, je v optimálnom prípade možné súčasné spracovanie pomocou všetkých Z blokov tak, že v dobe, keď bol prvý operand spracovaný v prvom bloku a prechádza do druhého bloku, začne prvý blok spracovávať druhý operand, atď. V určitom okamihu sa tak bude v každom bloku spracovávať iný operand, ktorý bude po ukončení spracovania odovzdaný

[4] Táto definícia paralelného systému je veľmi všeobecná a závisí na pojme proces. V tejto kapitole bude proces vyjadrovať spracovávanie inštrukcie prípadne dát v DSP. V ďalších kapitolách však procesom bude napr. aj časť úlohy ako napr. číslicová filtrácia vo filtračnom koprocesore.

[5] Zreťazeným spôsobom môže byť spracovávaná inštrukcia programu v inštrukčnom procesore tak aj dáta v dátovom procesore. Tieto procesory sú základnými blokmi harvardskej architektúry.

na ďalšie spracovanie do bloku s vyšším poradovým číslom. Takýto spôsob spracovania sa označuje termínom *zreťazené spracovanie* (pipelining).

Zreťazené spracovanie sa v procesorovej technike široko využíva, vzhľadom na možnosť zvýšenia výkonnosti pri relatívne nízkych nákladoch (obyčajne umiestnením vhodných vyrovnávacích registrov). Zreťazenie však môže výrazne skomplikovať programovanie procesora a pri konštrukcii je snaha tento vplyv eliminovať [43], [44]. Podrobnejšie je táto problematika z pohľadu DSP analyzovaná v podkapitole 3.3.

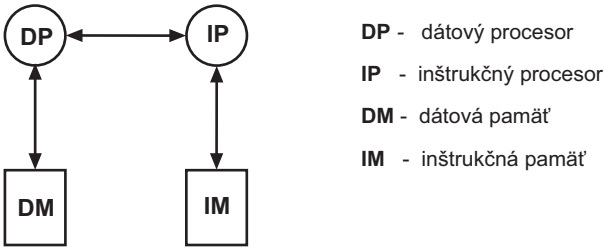


Obr. 3.2 Dve formy súbežného spracovania

3.2.2 HARVARDSKÁ ARCHITEKTÚRA

Je veľmi častým javom, že niektoré princípy boli známe omnoho skôr ako došlo k ich širokému využitiu. Vývoj harvardskej architektúry patrí do tejto kategórie [45]. Architektúra prvého významného elektromechanického počítača mala oddelené pamäte pre program a dáta, čím bol umožnený súčasný prístup k týmto pamätiam. Táto architektúra bola vyvinutá na konci 30-tych rokov Howardom Aikenom z Harvardskej univerzity a prvý počítač Harvard Mark 1 bol skonštruovaný v roku 1944. Na báze harvardskej architektúry pracoval aj známy počítač ENIAC (Electronic Numerical Integrator and Calculator) postavený v rokoch 1943 – 1946 na Pennsylvanskej univerzite [45]. Základná štruktúra harvardskej architektúry je znázornená na obr. 3.3 [4]. *Inštrukčný procesor* (IP) je funkčná jednotka, ktorá interpretuje inštrukcie a riadi *dátový procesor* (DP), ktorý modifikuje príslušné dáta. Obe tieto funkčné jednotky majú samostatné pamäte. *Dátová pamäť* (DM) uchováva dáta pre DP *a inštrukčná pamäť* (IM) uchováva inštrukcie pre IP.

Vzhľadom na zložitosť systému s oddelenými zbernicami však získala omnoho väčší význam *von Neumannova architektúra*, ktorá dostala názov po jednom z konzultantov ENIAC projektu, matematikovi maďarského pôvodu, ktorého meno bolo John von Neumann. Táto architektúra s unifikovanou programovou a dátovou pamäťou, aj keď principiálne pomalšia, sa stala štandardom vo vývoji najrozšírenejších počítačových architektúr na viac ako 40 nasledujúcich rokov po jej publikovaní v roku 1946.



Obr. 3.3 Základná harvardská architektúra

DSP však vzhľadom na potrebu vysokej výpočtovej výkonnosti využili potenciálne rýchlejšiu, aj keď zložitejšiu, harvardskú architektúru, pričom v oblasti DSP je možné identifikovať niekoľko jej modifikácií.

3.2.3 MODIFIKÁCIE HARVARDSKEJ ARCHITEKTÚRY

V oblasti DSP využívané modifikácie harvardskej architektúry je možné klasifikovať predovšetkým z pohľadu organizácie pamätí [4] [46], [47]. Cieľom jednotlivých modifikácií je *zvýšenie priepustnosti* DSP pri súčasnej snahe *optimalizovať* ďalšie parametre ako je *plocha čipu, príkon a zložitosť riadenia*. Základným cieľom je umožniť viacoperandové[6] inštrukcie pri minimalizácii nárokov na rýchlosť resp. počet pamätí v DSP. Hlavný dôraz je kladený na umožnenie dvojoperandových inštrukcií, čo vyplýva z faktu, že napr. pri implementácii FIR filtra (3.1) (základného algoritmu pre ktorý sú DSP optimalizované) je potrebné čítať dva vstupné operandy (koeficient h_k a vstupnú vzorku $x[n-k]$ N-krát častejšie ako zápis výstupnej vzorky). Podobné úvahy je možné využiť aj pre FFT. Samozrejme možnosť realizácie troj a viacoperandových inštrukcií je vítaná, je však zvyčajne spojená s dodatočnými nárokmi na rýchlosť resp. zložitosť DSP.

V praxi sa využíva niekoľko modifikácií harvardskej architektúry:

a) Základná harvardská architektúra

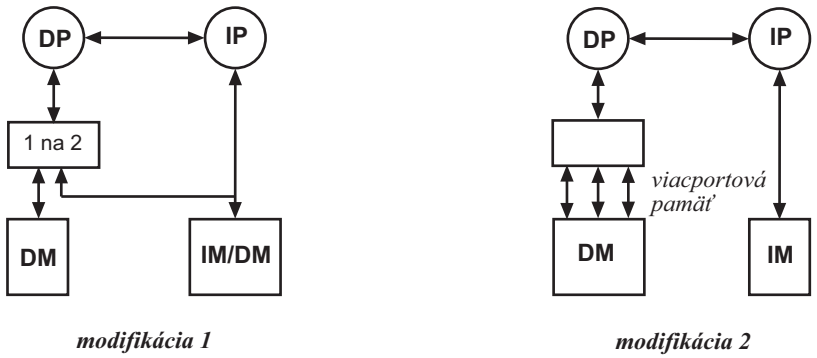
Táto architektúra (obr. 3.3) využíva jednu DM a jednu IM. Každá z pamätí využíva samostatné adresové a dátové zbernice. Výber inštrukcie z IM a dát z DM je realizovaný paralelne. Pre jednoslovné inštrukcie je dĺžka inštrukčného cyklu *rovná* dobe prístupu do pamäte. Príkladom DSP, ktorý využíva túto architektúru je TMS320C10 od TI.

b) Modifikácia 1

Modifikácia 1 zobrazená na obr. 3.4 umožňuje uloženie dát a inštrukcií v IM. Inštrukcie a dáta nemôžu byť v prípade dvojoperandových inštrukcií získané paralelne. Táto nevýhoda je kompenzovaná využitím pamätí, ktorých doba prístupu je *rovná polovici inštrukčného cyklu*. Na základe tejto vlastnosti je možné za optimálnych podmienok realizovať dokonca v jednom inštrukčnom cykle aj trojoperandové inštrukcie. Túto modifikáciu využíva napr. procesor DSP32C od AT&T, ktorý obsahuje dve banky RAM pamätí a dátovú zbernicu, ktorá v jenom inštrukčnom cykle umožňuje 4 prístupy (po 2 do/z každej pamäte RAM) a teda DSP32C umožňuje realizovať výber dvoch operandov, MAC

[6] Pri úvahách v tejto podkapitole sú uvažované operandy umiestnené v pamätiach DSP a nie v jeho registroch, pretože práve tieto operandy vyžadujú zvýšenú priepustnosť zbernicového systému.

operáciu a zápis výsledku[7] v jednom inštrukčnom cykle, pokiaľ operandy, inštrukcia a výsledok sú vhodne umiestnené v RAM pamätiach.



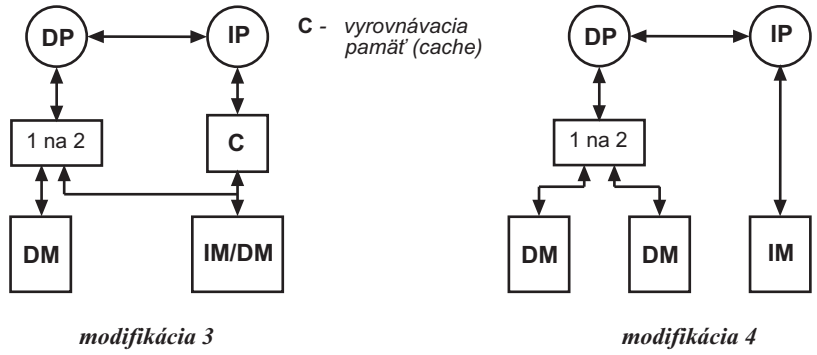
Obr. 3.4 Harvardská architektúra – modifikácia 1 a modifikácia 2

c) Modifikácia 2

V prípade, že DM je *viacportová pamäť* (multi-port memory), je možné realizovať niekoľko prístupov v rámci jedného inštrukčného cyklu. Táto modifikácia je znázornená na obr. 3.4 . Príkladom DSP, ktorý využíva trojportovú DM je MB86232 od Fujitsu, čo umožňuje tomuto procesoru realizovať trojoperandové inštrukcie v jednom inštrukčnom cykle. Vo všeobecnosti sú viacportové pamäte drahé a sú, vzhľadom na obmedzenia počtu vývodov, prakticky realizovateľné len v rámci interných pamätí.

d) Modifikácia 3

V prípade modifikácie 1 nastáva konflikt, pokiaľ je potrebné pristúpiť súčasne k dátam a inštrukcii, ktoré sú umiestnené v IP. Dodatočná *vyrovnávacia pamäť* (cache) použitá v modifikácii 3 na obr. 3.5 umožňuje uloženie inštrukcií a realizáciu *opakovaných krátkych slučiek* (veľmi typických pre algoritmy ČSS) bez opakovaného čítania inštrukcií z IM a teda IM môže byť využitá pre druhý operand. Túto modifikáciu využívajú napr. DSP16 od AT&T a ADSP2100 od Analog Devices, ktoré využívajú vyrovnávacie pamäte pre 15 resp. 16 inštrukcií, čo potvrdzuje optimalizáciu pre veľmi krátke segmenty programov.



Obr. 3.5 Harvardská architektúra – modifikácia 3 a modifikácia 4

[7] Vzhľadom na využívanie zreťazenia v IP je to výsledok niektorej (v závislosti na úrovni zreťazenia) z predchádzajúcej operácie.

e) Modifikácia 4

Namiesto využitia vyrovnávacej pamäte niektoré DSP využívajú oddelenú IM a dve samostatné DM. Táto modifikácia je využitá napr. v DSP5600x a DSP96002 od Motoroly. DSP využívajúce túto modifikáciu môžu realizovať prístup k dvom údajom a jednej inštrukcii aj v prípade, ak je prístup k pamäti rovný inštrukčnému cyklu. Táto modifikácia, znázornená na obr. 3.5 sa niekedy zvykne označovať ako *duálna harvardská architektúra*.

3.3 ZÁKLADNÉ BLOKY SIGNÁLOVÝCH PROCESOROV

Architektúry typických DSP obsahujú niekoľko paralelných blokov, ktoré síce vzájomne veľmi úzko spolupracujú, realizujú však výrazne odlišné činnosti. Pri podrobnejšej analýze je výhodné tieto bloky opísať samostatne.

3.3.1 DÁTOVÉ CESTY

Dátové cesty (data paths) sú miestom, kde sa realizujú základné aritmetické a logické operácie s dátami. Dátové cesty typických DSP obsahujú paralelnú násobičku, *aritmeticko-logickú jednotku*[8] (ALU – Arithmetic Logic Unit), jeden alebo viac posúvačov (shifters), operačné registre, akumulátory a prípadné ďalšie špecializované jednotky.

a) Násobička

Je základným stavebným blokom DSP. Prakticky všetky typy DSP môžu poskytnúť výsledok násobenia v každom inštrukčnom cykle. Niektoré DSP však využívajú interné zreťazenie, čo spôsobuje, že výsledok je dostupný až o niekoľko cyklov neskôr a teda dosahujú výkonnosť jedného násobenia/inštrukciu len pre dlhú opakovanú postupnosť násobení. Niektoré typy DSP majú násobičku integrovanú spolu so sčítačkou a realizujú priamo MAC operáciu (napr. DSP5600x), iné typy (napr. DSP16xx od AT&T) využívajú oddelenú násobičku a sčítačku a tak výsledok MAC operácie je oneskorený o jednu inštrukciu. Násobičky typicky poskytujú výsledok, ktorý má dvojnásobnú dĺžku ako vstupné operandy a v prípade operácie MAC je výsledok akumulovaný v *akumulátoroch*, ktorých dĺžka je zväčšená o tzv. *ochranné bity* (typicky 8 bitov, menej často 4 bity), čo vedie k typickej dĺžke akumulátorov 40 bitov (16-bitové DSP) a 56 bitov (24-bitové DSP).

b) ALU

Táto jednotka vykonáva základné aritmetické operácie (sčítanie, odčítanie, inkrementovanie, dekrementovanie, negovanie) a logické operácie (logický súčin, súčet a negáciu). Dĺžka operandov pre aritmetické a logické operácie môže byť rovná dĺžke akumulátora (napr. DSP16xx), alebo môže byť len časťou akumulátora (napr. DSP5600x). Ako už bolo uvedené, niektoré typy DSP používajú ALU aj na realizáciu MAC operácie, iné typy DSP pre tento účel využívajú oddelenú sčítačku (napr. ADSP21xx).

c) Posúvač (shifter)

Posúvač je možné typicky nájsť bezprostredne za násobičkou a ALU, niektoré typy DSP umožňujú posúvať aj vstupné dáta týchto jednotiek. Typická je možnosť realizovať posun výsledku o 1 bit doľava, doprava alebo posun nerealizovať, pričom je operácia

[8] Niektorí výrobcovia označujú termínom ALU všetky časti dátových ciest, v rámci tejto práce, podobne ako napr. v [23] je termínom ALU označovaná aritmetická jednotka na realizáciu sčítania, odčítania a logických operácií.

posunu realizovaná paralelne s operáciami ALU a násobičky. Klasické viacbitové posuny vyžadujú väčší počet inštrukcií čo niektoré DSP nahradzujú využívaním špeciálnej jednotky *viacbitového posúvača* (barrel shifter).

3.3.2 RIADIACA JEDNOTKA

Typické riadiace jednotky DSP využívajú viacúrovňové zreťazenie, najčastejšie *trojúrovňové* (výber inštrukcie, dekódovanie a vykonanie), prípadne *štvorúrovňové* (výber inštrukcie, dekódovanie, výber operandu a vykonanie). Existujú však aj DSP s dvoj a päťúrovňovým zreťazením. Vo všeobecnosti čím vyššia úroveň zreťazenia, tým je možné dosiahnuť vyšší výkon DSP, výrazne sa však komplikuje programovanie DSP. V DSP sú využívané nasledujúce modely riadenia zreťazenia [4], [23]:

a) Časovo stacionárny (time stationary) model

Inštrukcie procesora *špecifikujú činnosť výkonných jednotiek* (násobička, ALU, adresové jednotky…) v každom inštrukčnom cykle. Typickým príkladom je DSP16xx, pre ktorý má MAC inštrukcia tvar

```
a0=a0+p     p=x*y        y=*r0++      x=*pt++
```

a vykoná akumuláciu predchádzajúceho súčinu (p), vynásobí obsahy registrov x, y a uloží výsledok do registra p. Zároveň načíta obsahy pamätí na adresách r0 a pt do registrov x,y a inkrementuje registre r0 a pt. Každá časť inštrukcie pracuje s odlišnými operandami.

b) Dátovo stacionárny (data stationary) model

Inštrukcie procesora *špecifikujú operácie, ktoré sa majú vykonať*, ale nešpecifikuje čas, kedy sa tieto operácie vykonajú. Typickým príkladom je kód pre DSP32xx

```
a1=a1+(*r5++ = *r4++) * *r3++
```

ktorý zabezpečí vynásobenie dát v pamäti na adrese r3 a r4, uloží obsah pamäťového miesta na adrese r4 na adresu r5, inkrementuje registre r3, r4, r5 a výsledok súčinu pripočíta k akumulátoru a1. V rámci tohto modelu riadenia inštrukcia *opisuje prechod množiny dát sekvenciou operácií*.

c) Model s využitím blokovania (interlocking)

Technické prostriedky v rámci riadiacej jednotky *sledujú konflikty a zabezpečujú korektné vykonávanie* operácií (zvyčajne oneskorením konfliktných inštrukcií). Tento spôsob je pre programátora plne transparentný a je pomocou neho možné výrazne zjednodušiť programovanie DSP. Typickým príkladom je DSP563xx od firmy Motorola, ktorý vzhľadom na využitie blokovania umožňuje binárnu kompatibilitu s procesormi DSP560xx, ktoré využívajú časovo stacionárny model riadenia. Maximálne využitie DSP však vyžaduje minimalizáciu konfliktov dosahovanú pomocou preusporiadania konfliktných inštrukcií.

3.3.3 ADRESOVÉ GENERÁTORY A PAMÄŤOVÝ SYSTÉM

Vysoká výkonnosť DSP je umožnená okrem výkonných dátových ciest aj veľkou priepustnosťou pamäťového systému. Typická realizácia FIR filtra (3.1) vyžaduje prenos dvoch údajov (koeficient a vzorka) v jednom inštrukčnom cykle. Zápis výsledku je realizovaný N-krát pomalšie a je teda z pohľadu prístupu k pamäti menej kritický. Typické

DSP obsahujú dve (existujú však aj DSP len s jednou jednotkou) špeciálne jednotky – *adresové aritmetické jednotky*, (AAU – Address Arithmetic Units) ktoré umožňujú generovať dve adresy pre prístup k dvom dátovým pamätiam v jednom inštrukčnom cykle. Tieto jednotky môžu pracovať paralelne s dátovými cestami pričom typicky podporujú nasledujúce spôsoby adresovania [4], [23]:

- ***lineárne*** – klasické adresovanie,
- ***modulo*** – vhodné pre vytváranie napr. oneskorovacích liniek číslicových filtrov,
- ***bitovo reverzné*** – vhodné na efektívne preusporiadanie výstupu FFT algoritmu.

Vysoká priepustnosť pamäťového systému je v rámci DSP čipu umožnená *integráciou* IM a DM spolu s podpornými adresovými a dátovými zbernicami *priamo do čipu DSP*, čo umožňuje realizáciu jednočipových systémov ČSS. Novšie DSP umožňujú aj využitie externých IM a DM, pričom interné zbernice sú typicky *multiplexované* do jednej externej adresovej a jednej externej dátovej zbernice, čo výrazne znižuje priepustnosť pri práci s externými pamäťami.

3.3.4 PERIFÉRNE OBVODY

Periférne obvody v DSP zabezpečujú predovšetkým komunikáciu s externými zariadeniami. Prakticky od počiatku vývoja DSP sú súčasťou DSP *sériové rozhrania* pre pripojenie A/D a D/A prevodníkov, čo umožňuje zníženie počtu vodičov potrebných na pripojenie externých prevodníkov. Tieto rozhrania je často možné využiť aj na prepojenie viacerých procesorov a vytvorenie jednoduchších sietí DSP.

Medzi ďalšie štandardné periférne obvody patria paralelné rozhrania pre pripojenie k nadradeným systémom, typicky k štandardným 8 a 16-bitovým mikroprocesorom. Aj keď základné vlastnosti týchto obvodov sú prakticky rovnaké u všetkých typov DSP, detailné vlastnosti sú špecifické pre každý typ DSP a ich opis v manuáloch DSP často zaberá viac priestoru ako opis jadra DSP.

3.3.5 PREVODNÍKY NA BÁZE SIGMA-DELTA MODULÁCIE

Aj keď A/D a D/A prevodníky by bolo možné zahrnúť medzi periférne obvody, ich postavenie z pohľadu systému ČSS na obr. 2.1 je veľmi špecifické, pretože umožňuje realizovať z číslicového procesora (t.j. číslicového systému) systém diskrétny. Existuje niekoľko základných princípov využívaných v A/D a D/A prevodníkoch. V oblasti spracovania rečových a audio signálov (teda oblastí, kde majú v súčasnosti DSP dominantné postavenie) však v poslednom desaťročí dominuje použitie prevodníkov na báze sigma-delta modulácie [24], [27]. Tieto prevodníky využívajú princíp *prevzorkovania* a *spektrálneho tvarovania* (noise shaping) kvantizačného šumu, ktorý sa realizuje sigma-delta modulátorom a.z pohľadu DSP majú tieto výhodné vlastnosti:

- vyžadujú minimum presných analógových prvkov, čo výrazne znižuje ich cenu,
- využívajú princípy ČSS, ich veľká časť môže byť implementovaná lacnou CMOS VLSI technológiou a integrovaná na jednom čipe spolu s DSP,
- dosahujú vysoké rozlíšenie (typicky 16 bitov, často až 24 bitov pri frekvencii vzorkovania 8 až 100 KHz,
- princíp prevodu zabezpečuje vysokú linearitu prevodu,
- umožňujú znížiť nároky na extrený obmedzovací a rekonštrukčný analógový filter.

Výhodné vlastnosti tejto triedy prevodníkov boli využité v rámci niektorých DSP (napr. DSP56156 od Motoroly, ADSP 2159 od Analog Devices) na integráciu A/D a D/A prevodníkov priamo do čipu DSP.

Určitou nevýhodou sigma-delta prevodníkov je ich relatívne nízky rozsah vzorkovacích frekvencií, čo je dané vysokým prevzorkovaním (často 256 a viac) s ktorým tieto prevodníky interne pracujú. Aj v tejto oblasti je však možné očakávať výrazný pokrok predovšetkým pri spracovaní úzkopásmových vysokofrekvenčných signálov [25], [26] ktoré je potrebné pri realizácii tzv. *softvérového rádia* [28], čo je z pohľadu DSP jeden z najbližších technologických cieľov.

3.4 POROVNÁVANIE VÝKONNOSTI – BDTIMARK

Porovnávanie výpočtovej výkonnosti rôznych procesorov je úloha značne komplikovaná [48]. Napr. v oblasti univerzálnych mikroprocesorov sa používajú rôzne typy skúšobných úloh (tzv. benchmarks) ako napr. Wheatstone, Dhrystone alebo Linpack [41]. Jednoduché charakteristiky, ktoré výrobcovia DSP typicky uvádzajú vo forme MIPS, MOPS, MFOPS a pod. sú pre prax často nepostačujúce, pretože výrobcovia často interpretujú ich význam rôzne. Naviac napríklad výkonnosť MIPS značne závisí od architektúry DSP, veľkostí interných pamätí alebo mechanizmu riadenia vyrovnávacích pamätí a tak často DSP s nižším parametrom MIPS môže v prípade praktických aplikácií dosiahnuť vyšší výpočtový výkon. Druhým extrémom je porovnávanie výkonnosti na základe efektívnosti implementácie *kompletných aplikácií* ako sú napr. implementácia CELP vokodéra, V34 modemu a pod. Kompletné algoritmy často odrážajú skôr efektivitu a zručnosť programátora a sú ťažko reprodukovateľné. Ako pre prax optimálne kompromisné riešenie sa ukazuje porovnávanie výkonnosti pomocou súboru *jadier základných algoritmov*, prevažne z oblasti ČSS. V oblasti DSP je najznámejší systém testovacích úloh firmy Berkeley Design Technology, Inc. (www.bdti.com), ktorý zavádza *testovaciu metriku*, ktorej jednotkou je tzv. *BDTImark*, pričom tento testovací systém má nasledujúce vlastnosti:

- ***relevantnosť*** – táto metrika odráža výkonnosť procesorov pre typicky používané jadrá algoritmov ČSS, ktorých zoznam so stručnou charakteristikou je uvedený v tab. 3.2 ,
- ***jednoduchosť*** – vyjadrenie hodnoty metriky jediným číslom umožňuje rýchle a ľahké porovnanie rôznych procesorov, pričom hodnoty BDTImark vyjadrujú výkonnosť v lineárnej mierke a vyššia hodnota vyjadruje vyšší výpočtový výkon,
- ***aplikovateľnosť*** – metrika je aplikovateľná pre ľubovoľný typ programovateľného procesora a je nezávislá na jeho architektúre,
- ***nezávislosť*** – metrika je vyhodnocovaná *nezávislou firmou*, čo zaručuje, že výsledky sú vypočítavané korektne,
- ***dostupnosť*** – hodnoty metriky sú bezplatne verejne dostupné na WWW stránke firmy.

Samozrejme tento výkonnostný test má v praxi aj obmedzenia. Ako hodnota závislá na množine testovacích algoritmov, hodnota BDTImark neodráža napr. špecializáciu procesora pre konkrétne aplikácie (pre ktoré napr. DSP môže obsahovať špeciálne koprocesory). Hodnoty BDTImark odrážajú rýchlosť testovaného procesora pre všeobecne používané typy algoritmov ČSS a na ich hodnoty nemajú vplyv ďalšie pre prax často dôležité vlastnosti ako je cena, príkon a pod. Všetky testovacie algoritmy sú optimalizované v asembleri príslušného procesora a využívajú prirodzenú dĺžku slova

testovaného procesora. Pri interpretácii hodnôt BDTImark je preto potrebné vziať do úvahy, že procesory s pohyblivou rádovou čiarkou poskytujú presnejší výsledok.

Tab. 3.2 Zoznam algoritmov použitých pri vyhodnocovaní hodnoty BDTImark

Funkcia	Význam	Príklad aplikácie
Blokový reálny FIR	Bloková realizácia filtra s konečnou impulzovou odpoveďou pre reálne dáta	Spracovanie reči (napr. algoritmus G728)
Blokový komplexný FIR	Bloková realizácia FIR filtra pre komplexné dáta	Ekvalizácia kanálu v modemoch
Reálny FIR	FIR filter pre spracovanie jednej reálnej vzorky	Spracovanie reči, všeobecná filtrácia
LMS adaptívny FIR	Gradientný stochastický adaptívny algoritmus pre spracovanie jednej reálnej vzorky	Ekvalizácia kanálov, riadenie servopohonov, lineárne predikčné kódovanie
IIR	Rekurzívny filter s nekonečnou impulzovou odpoveďou pre jednu reálnu vzorku	Spracovanie audiosignálov, všeobecná filtrácia
Vektorový súčin	Súčet výsledkov komponentného násobenia zložiek dvoch vektorov	Konvolúcia, korelácia, násobenie matíc, viacrozmerné spracovanie signálov
Vektorový súčet	Komponentný súčet dvoch vektorov, výsledkom je tretí vektor	Grafika, kombinovanie audio signálov a obrazov, vektorové prehľadávanie
Maximum vo vektore	Nájdenie maximálnej hodnoty a jeho polohy vo vektore	Algoritmy protichybového zabezpečenia, aritmetika s blokovou pohyblivou čiarkou
Konvolučný kodér	Aplikácia konvolučných zabezpečovacích kódov na blok bitov	Americký štandard digitálnej mobilnej telefónnej siete (štandard IS 54)
Konečný automat	Klasické riadenie (testovanie, vetvenie) a bitové manipulácie	Prakticky všetky DSP aplikácie obsahujúce určitú formu riadenia
256 bodová, radix 2 FFT	Rýchla Fourierova transformácia	Radary, sonary, MPEG audio kompresia, spektrálna analýza

3.5 ZHRNUTIE

Táto kapitola stručne opísala modifikácie harvardskej architektúry a základné stavebné bloky používané v oblasti klasických DSP. Táto oblasť je v literatúre dobre rozpracovaná [4], [23] a odráža stav DSP do polovice 90-tych rokov. Mnohé z princípov použitých v DSP sa dokonca uplatňujú aj v oblasti jednočipových mikropočítačov [49] a univerzálnych mikroprocesorov [48]. Klasické DSP sa v 90-tych rokoch vyvíjali ďalej predovšetkým na základe technologického pokroku CMOS VLSI technológie, pričom časť nových DSP začala výrazne využívať nové formy paralelizmu, čo viedlo k vytvoreniu skupiny DSP s úplne novými vlastnosťami. Tieto DSP budú v ďalšej časti označované termínom *moderné* DSP.

4 MODERNÉ SIGNÁLOVÉ PROCESORY

Z predchádzajúcich úvah je zrejmé, že aj keď taktovacie frekvencie DSP sa stále zvyšujú a teda rastie aj ich výpočtový výkon, výraznejšie zvýšenie výpočtového výkonu je možné len zvýšeným využitím *paralelizmu*. V oblasti DSP sa využívajú dva základné spôsoby zvýšenia paralelizmu

- zväčšenie počtu operácií vykonaných každou inštrukciou,
- zväčšenie počtu inštrukcií vykonaných v každom cykle.

Prvý spôsob sa realizuje zahrnutím nových vlastnosti do dátových ciest a je typický predovšetkým u výrobcov DSP, ktorí sa snažia zvýšiť výkonnosť DSP bez radikálnej zmeny celkovej architektúry. Tieto DSP budú v ďalšej časti nazývané *rozšírené klasické* DSP (enhanced conventional DSP) [51]. Naopak druhý spôsob, paralelné vykonávanie viacerých inštrukcií je typické pre najnovšie architektúry DSP – *DSP s paralelným vykonávaním inštrukcií*, pričom v tejto triede DSP sa často využívajú aj vylepšenia, ktoré využívajú obidva spôsoby zvýšenia paralelizmu. Cieľom tejto kapitoly je na základe naznačenej klasifikácie opísať najnovšie trendy v rozvoji moderných DSP.

4.1 ROZŠÍRENÉ KLASICKÉ SIGNÁLOVÉ PROCESORY

Približne v polovici 90-tych rokov sa objavili nové varianty DSP, ktoré na základe evolučného vývoja architektúr klasických DSP poskytli oproti klasickým DSP zvýšený výpočtový výkon. Toto zlepšenie bolo dosiahnuté okrem technologického pokroku, predovšetkým *aplikačne orientovanými vylepšeniami*. Do tejto kategórie DSP patria napr. TMS320C54x od TI, DSP563xx a DSP566xx od Motoroly, DSP16xxx od firmy Lucent a čiastočne aj AD218x od Analog Devices. Z pohľadu klasifikácie je ich možné zaradiť do 3. generácie DSP s pevnou rádovou čiarkou.

4.1.1 TECHNOLOGICKÉ ZLEPŠENIA

Technologický pokrok CMOS VLSI technológie sa prejavil (a v tejto kategórii DSP sa stále prejavuje) predovšetkým v týchto oblastiach:

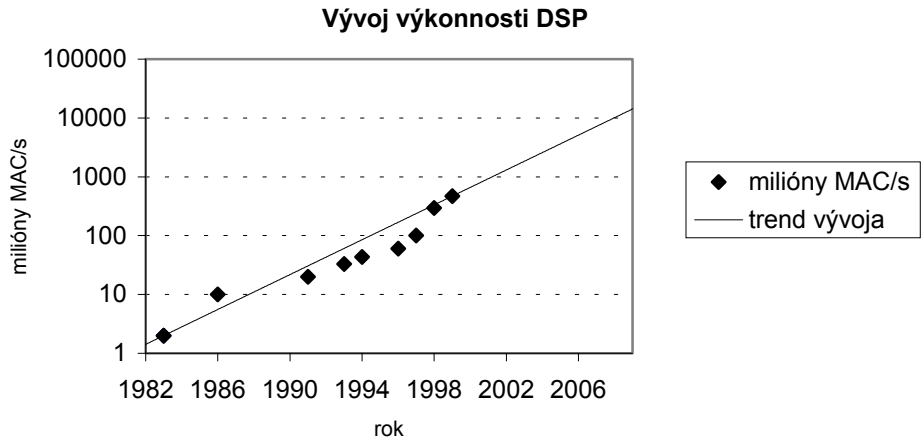
a) Zvyšovanie taktovacej frekvencie

Najvyššia taktovacia frekvencia procesorov tejto kategórie v súčasnosti presahuje[9] 100 MHz a sú ohlasované stále vyššie taktovacie frekvencie (napr. 150 MHz pre DSP56311). Naviac oproti DSP z druhej generácie je veľmi časté, že počet vykonaných inštrukcií za sekundu je zhodný s taktovacou frekvenciou (takzvaná *1X architektúra*).

[9] Zaostáva však za taktovacími frekvenciami najvýkonnejších mikroprocesorov. Dôvodom je predovšetkým orientácia DSP na nízkopríkonové aplikácie.

Prakticky ako štandard sa využíva generovanie taktovacej frekvencie priamym násobením na čipe buď pomocou násobičiek alebo, stále častejšie, konfigurovateľnými obvodmi PLL.

Nárast výpočtového výkonu DSP vyjadrené v miliónoch MAC operácii za sekundu a predpokladané trendy[10] sú znázornené na obr. 4.1 [29].



Obr. 4.1 Nárast výkonnosti DSP vyjadrený v miliónoch MAC operácií za sekundu

b) Zvyšovania úrovne zreťazenia

Zreťazenie základných operácii riadiacej jednotky (výber, dekódovanie a vykonanie) bolo využívané už u 1. a 2. generácie DSP. Využitie zreťazených dátových ciest však využívali len niektorí výrobcovia DSP a napr. Motorola u radu DSP5600x a Analog Devices u radu AD218x vystačili s nezreťazenými jednotkami MAC. Architektúra 1X však vzhľadom na technologickú úroveň už vyžaduje použitie zreťazenia aj v rámci dátových ciest (jednotkách MAC, ALU…). DSP typicky využívajú v riadiacich jednotkách väčšiu úroveň zreťazenia (napr. 7 úrovní v DSP563xx ako v MAC jednotkách (napr. 2 úrovne v DSP563xx), čo je však stále podstatne menej ako napr. v oblasti mikroprocesorov pre pracovné stanice, kde sa typicky využíva zreťazenie s niekoľkými desiatkami úrovni zreťazenia.

Relatívne nízka úroveň zreťazenia sa pozitívne prejavuje v zachovaní malej zložitosti programovacieho modelu aj pre túto kategóriu DSP[11]. Vzhľadom na nutnosť použitia ručnej optimalizácie kódu (minimálne pre najkritickejšie časti kódu) pre túto triedu DSP je udržanie nízkej úrovne zreťazenia prakticky nutnosťou. Aj keď zavedením zreťazenia do jednotky MAC vznikajú nové obmedzenia (typu výsledok je dostupný až o x cyklov neskôr), existuje často snaha tento problém eliminovať na úrovni technických prostriedkov využitím modelu zreťazenia s využitím vzájomného blokovania (napr. DSP563xx). Takéto riešenie môže dokonca zabezpečiť kompatibilitu na úrovni binárneho kódu s procesormi predchádzajúcej generácie. Objektívne však treba povedať, že v prípade dosiahnutia

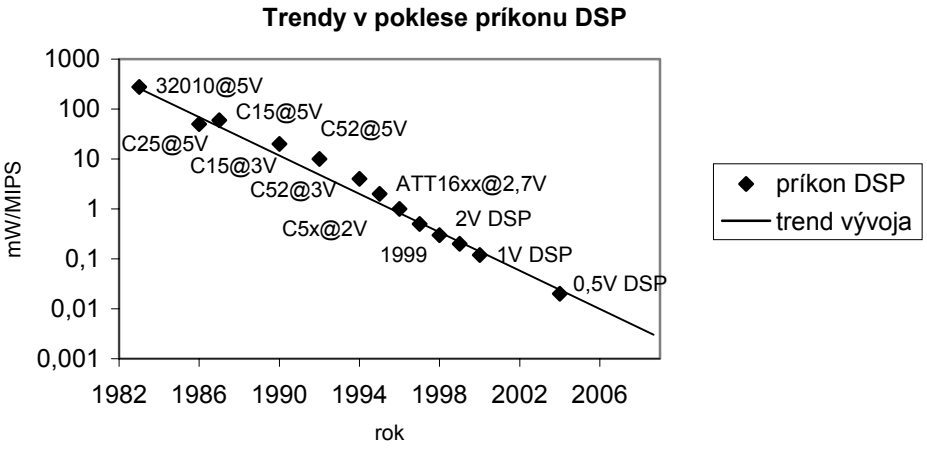
[10] Nárast výpočtovej výkonnosti od roku 1998 je už však zabezpečovaný zvýšeným využitím paralelizmu.

[11] Aj keď to znie paradoxne, pretože od 1. generácie DSP je rozšírený názor, že programovanie DSP je veľmi zložité. Ručná optimalizácia kódu napr. pre procesor Intel Pentium je však v súčasnosti podstatne zložitejšia čo je dané jeho zložitejším programovacím modelom.

maximálneho výpočtového výkonu je potrebné pri optimalizácii kódu všetky obmedzenia vziať do úvahy a minimalizovať prípady zablokovania.

c) Znižovanie napájacieho napätia a príkonu

Napájacie napätie tejto triedy DSP už výrazne pokleslo pod hodnotu 3.3 V a vzhľadom na potrebu zabezpečenia kompatibility s externými obvodmi dochádza k rozdeleniu napájacieho napätia pre jadro DSP (v súčasnosti okolo 1,8 V) a externé obvody DSP (typicky 3,3 V, prípadne 2,5 V). Tento trend je všeobecný a je ho možné pozorovať napr. aj v oblasti mikroprocesorov pre pracovné stanice a v oblasti zložitých programovateľných hradlových polí. V oblasti DSP pre mobilné zariadenia (napr. GSM telefóny) sa napájacie napätie blíži k hranici 1 V [52], [53], čo je dosahované využívaním najnovších nízkopríkonových CMOS technológií [173] (v súčasnosti sú dostupné vzorky ultra-nízkopríkonových DSP TMS320UVC5402 a TMS320UVC5409 s napájacím napätím jadra 1,2 V a napätím I/O obvodov 1,2 V alebo 1,8 V). Pokles napájacieho napätia a príkonu DSP spolu s vylepšeniami v architektúre vo všeobecnosti vedie k poklesu parametrov príkon/algoritmus. Samozrejme parametre mA/MIPS a mW/MIPS taktiež pre nové procesory klesajú (napr. 0,54 mW/MIPS pre TMS320UVC5402 pri napájacom napätí 1,2 V). Tieto trendy sú naznačené na obr. 4.2 [29]. Znižovanie príkonu a napájacieho napätia DSP vytvára v súčasnosti výrazný tlak predovšetkým na optimalizáciu externých analógových obvodov využívaných v telekomunikačnej technike [54].



Obr. 4.2 Energetické trendy v oblasti DSP

d) Zväčšovania veľkostí interných pamätí

Interné pamäte majú z hľadiska výkonnosti pre DSP veľký význam. Takmer všetky dostupné DSP dosahujú pri práci s internými pamäťami podstatne vyšší výkon, pretože prístup k externým pamäťovým bankám harvardskej architektúry je z dôvodu zníženia počtu I/O vývodov *multiplexovaný*. Súčasný prístup k dvom alebo až trom externým pamäťovým bankám tak vyžaduje dva až tri inštrukčné cykly. Naviac pri zvyšujúcej sa taktovacej frekvencii a využívaní 1X architektúry požiadavky na rýchlosť prístupu k externým (typicky SRAM) pamätiam stále narastajú a pri taktovacích frekvenciách viac ako 100 MHz sú potrebné pamäte s dobou prístupu pod 10ns, čo je v prípade externých pamätí značný problém. V praxi tak musí byť prístup k pomalším externým pamätiam často realizovaný s využitím dodatočných čakacích stavov, čo ďalej znižuje výpočtový

výkon DSP. V minulosti bolo potrebné venovať značné úsilie vývoju optimalizovaných algoritmov ČSS pre DSP s malými internými pamäťami a príklady takýchto implementácií sú uvedené v kapitole 6.

V súčasnosti majú DSP s najväčšími internými pamäťami na čipe implementované SRAM pamäte s veľkosťou niekoľko Mbitov[12] (napr. 3,2 Mbitu v TMS320VC5420, 3 Mbity v DSP56311, 1,5 Mbitu v ADSP2189). Najnovšie DSP majú zodpovedajúcim spôsobom rozšírené aj adresové zbernice a umožňujú adresovať typicky 16 a viac Mslov[13] pamäte.

e) Integrácia nových typov pamätí

Okrem štandardných SRAM pamätí a PROM pamätí u maskou programovateľných verzií DSP sa začínajú objavovať verzie s FLASH pamäťami (napr. DSP1609F [50]). Tento trend, ktorý je výrazný predovšetkým v oblasti univerzálnych jednočipových mikropočítačov, je logickým dôsledkom o zníženie počtu čipov pri realizácii kompletného systému DSP na jednom čipe, čo vyžaduje umiestnenie programu do vnútra čipu DSP. Vzhľadom na rýchlosť dostupných FLASH pamätí je to stále len suboptimálne riešenie. Veľké nádeje sú vkladané do oblastí feroelektrických pamätí [55], [173] ktoré by mali poskytnúť výraznejšie kratšiu dobu prístupu než je doba prístupu FLASH pamätí, zatiaľ však tento typ technológie nebol v DSP použitý.

f) Integrácia radičov DMA

Pôvodne boli radiče DMA umiestňované do vysokovýkonných DSP s pohyblivou rádovou čiarkou a určené predovšetkým na podporu multiprocesorovej komunikácie. Vzhľadom na výrazne zväčšenie interných a externých pamäťových priestorov ako aj umiestňovanie výkonných periférií priamo na čipoch DSP (napr. štandardné sériové a paralelné kanály, ale aj špecializované DSP koprocesory a pod.) je výhodné na presuny veľkého objemu dát využívať mechanizmus kanálov DMA a tým zvýšiť priepustnosť celého DSP. Naviac, oproti klasickým DSP bez radičov DMA nie je potrebné na prenos (realizovaný napr. v klasických DSP pomocou prerušení) alokovať časť registrov DSP a tieto potom môžu byť využité na efektívnejšiu implementáciu algoritmov ČSS v samotnom jadre DSP. Počet DMA kanálov v moderných DSP je rôzny (napr. ADSP218x má 1-kanálový radič DMA, DSP563xx má 6-kanálový radič DMA) a najmodernejšie implementácie radičov DMA môžu pracovať v prípade interných DSP zdrojov paralelne s DSP jadrom bez vzájomného ovplyvňovania[14].

DMA kanály predstavujú unifikovaný spôsob prístupu k periférnym obvodom DSP a úspešne tak nahradzujú rôzne špecializované prístupy spolupráce s periférnymi obvodmi klasických DSP.

g) Integrácia ladiacich obvodov

Pri stále sa zvyšujúcej taktovacej frekvencii a využívaní SMD technológie je využívanie klasických emulačných techník, kedy sa cieľový procesor nahradil emulačnou hlavicou a jeho činnosť prevzal emulátor, prakticky nerealizovateľné. V praxi sa tento

[12] Uvádzanie veľkostí pamätí v Mbitoch je zvyčajne reklamnou záležitosťou a objektívnejším údajom je počet slov, čo je napr. pre 16-bitový procesor 16-krát menej.

[13] Existujú aj výnimky, keď sú síce na čipe umiestnené relatívne veľké pamäte, prístup k nim je však vzhľadom na malý rozsah adresových zberníc stránkovaný (napr. ADSP2189), čo znižuje ich užitočnosť.

[14] Napr. interná pamäť DSP563xx je rozdelená do baniek s veľkosťou 256 slov. Pokiaľ jadro DSP a radič DMA pristupujú súčasne do rôznych pamäťových baniek, jadro DSP a radič DMA pracujú úplne nezávisle.

problém rieši integráciou špeciálnych ladiacich obvodov priamo do každého čipu DSP, čo umožňuje realizovať emuláciu bez nutnosti vyberať čip z testovanej dosky plošného spoja. Na pripojenie je vyhradených niekoľko vývodov. V súčasnosti sú tieto vývody často zároveň využívané aj ako vývody štandardizovaného rozhrania JTAG [58], ktoré umožňuje predovšetkým testovanie správneho osadenia a porúch plošných spojov.

h) Vylepšené vyrovnávacie pamäte

DSP založené na modifikácii 3 harvardskej architektúry (str. 21) využívali vyrovnávacie pamäte veľmi malej veľkosti (typicky okolo 16 slov), pričom využitie tejto pamäte bolo automaticky zabezpečené pre opakované vykonávanie tej istej inštrukcie (zvyčajne po zadaní inštrukcie REP), prípadne v tzv. hardvérových slučkách (zvyčajne inštrukcia DO). Z hľadiska veľkosti vyrovnávacej pamäte je modifikácia 4 podstatne vhodnejšia, pretože veľkosť internej programovej pamäte bola minimálne niekoľko sto slov (napr. 512 slov v DSP5600x). Niektoré typy najnovších DSP na báze modifikácie 4 (napr. DSP563xx) aj keď využívajú podstatne zväčšené interné pamäte, majú naviac možnosť vyhradiť časť internej programovej pamäte (veľkosti 1 až 2 Kslov) ako vyrovnávaciu pamäť. Existuje niekoľko režimov činnosti tejto pamäte od režimu, ktorý je plne transparentný pre programátora, až po možnosť *riadenia uzamknutia* prípadne *uvoľnenia* jednotlivých sektorov vyrovnávacej pamäte pomocou inštrukcií vo vykonávanom programe. Tento nový (v oblasti DSP) prístup je výhodný predovšetkým v prípade, že sú pripojené pomalé externé programové pamäte a v prípade veľkých programov minimalizuje potrebu využívania techniky *dynamických prekryvných modulov* (dynamic overlays). Tieto fakty naznačujú, že zložitosť programov, pre ktoré sú najnovšie typy DSP *určené výrazne presahuje zložitosť programov* typicky implementovaných pomocou klasických DSP.

i) Multičipové moduly

Priamym spôsobom využitia paralelizmu, ktorý poskytuje technológia VLSI je umiestnenie viacerých čipov DSP do jedného puzdra – *multičipového modulu*. Z hľadiska koncového užívateľa je prakticky nepodstatné, či sa jedná o multičipový modul alebo jednočipovú integráciu viacerých procesorov. Logické je využitie multičipových modulov v oblasti vysokovýkonných DSP, ktoré sú originálne navrhnuté pre multiprocesorové moduly a multičipový modul môže výrazným spôsobom zlepšiť technické parametre koncových zariadení. Typickým príkladom je čip AD14060 od Analog Devices [59], ktorý integruje 4 čipy s pohyblivou rádovou čiarkou – AD21060 spolu s ich vzájomným prepojením alebo TMS320C8x, ktorý integruje 2 až 4 DSP s pevnou rádovou čiarkou a riadiaci RISC procesor.

Aj keď naznačený prístup je predovšetkým logický v oblasti vysokovýkonných DSP systémov, na trhu existuje[15] napr. TMS320VC5420 - variant nízkopríkonového DSP s výkonom 200 MIPS, ktorý obsahuje dva samostatné čipy DSP jadier s výkonom 100 MIPS. Aj keď multičipové moduly reprezentujú určitú formu využitia paralelizmu, z hľadiska výkonnosti je podstatne výhodnejšie využitie paralelizmu v rámci samotného DSP. Tieto spôsoby sú opísané v nasledujúcich častiach tejto kapitoly.

[15] Analog Devices nedávno ohlásil, že jeho najnovšií rad procesorov ADSP219x s pevnou rádovou čiarkou bude obsahovať aj multičipove moduly so 4 čipmi ADSP219x s celkovou výkonnosťou 1,2 miliardy MACs a 16 Mbitmi SRAM pamätí.

4.1.2 OPTIMALIZOVANÉ DÁTOVÉ CESTY

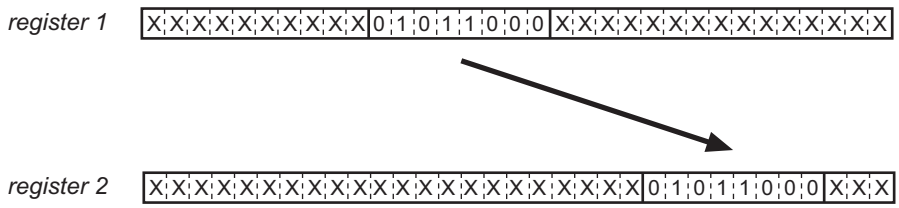
Klasické DSP mali architektúru výrazne optimalizovanú pre štandardné všeobecné algoritmy ČSS – číslicovú filtráciu a FFT. Ukázalo sa, že pre tieto typy algoritmov je potrebné zrýchliť MAC operáciu a tak sa MAC inštrukcia (podporovaná technickými prostriedkami priamo v dátových cestách) stala integrálnou súčasťou DSP. Podobne potreba špecializovaného prístupu k dátovým štruktúram v prípade číslicových filtrov (modulo adresovanie) a FFT (reverzné adresovanie) viedli k priamej podpore týchto režimov v adresových aritmetických jednotkách.

Široké nasadenie DSP predovšetkým v telekomunikačnej oblasti prispelo k rozšíreniu triedy algoritmov, pre ktoré výrobcovia DSP integrovali do dátových ciest nové bloky a zaviedli nové inštrukcie.

a) Jednotky pre bitovú manipuláciu (BFU - Bit Field Units)

Bitové manipulácie sú v telekomunikačnej oblasti extrémne často využívané, pretože pri prenose informácie sa často pracuje s jednotlivými bitmi. Typickým príkladom je realizácia presusporiadania jednotlivých bitov v bloku interlívera [179], CRC zabezpečenie [179] a pod. Už staršie DSP mali často možnosť realizovať základné bitové operácie ako napr. testovanie jednotlivých bitov v pamäti resp. v registroch. Novšie DSP (napr. ADSP21xx, DSP16xx) integrovali špeciálne viacbitové posúvače, ktoré realizovali viacbitové posuny v jednom inštrukčnom takte. Tieto operácie sú veľmi časté predovšetkým pri výpočte exponentu prípadne pri *normovaní čísel*, ktoré je využívané napr. pri implementácií rečových kodekov na báze CELP algoritmov. Podpora normovania sa objavila najskôr u 16-bitových DSP, kde pre mnohé algoritmy ČSS je potrebné realizovať výpočty v blokovej pohyblivej rádovej čiarke a eliminovať tak relatívne malú dĺžku slova tejto triedy DSP.

Ďalším krokom bola integrácia nových jednotiek pre bitové manipulácie, ktoré už okrem uvedených bitových operácii umožnili realizovať v jednom cykle aj operácie vloženia a extrakcie bitových reťazcov do/z iných bitových reťazcov. Na obr. 4.3 je tento typ operácie znázornený na príklade extrakcie bitového reťazca z registra 1 do registra 2. Zrýchlenie týchto typov operácii prispelo k výraznému zvýšeniu rýchlosti veľkej triedy telekomunikačných algoritmov.



Obr. 4.3 Princíp vloženia a extrakcie bitového reťazca

b) Podpora pre Viterbiho algoritmus

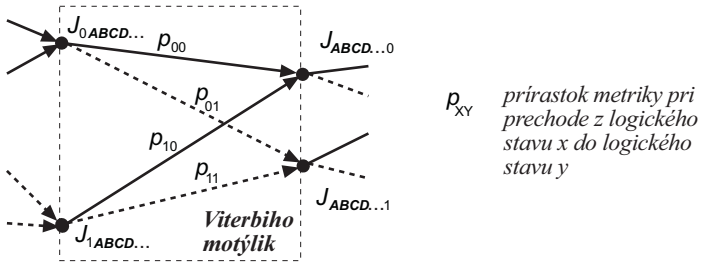
Viterbiho algoritmus (VA) je rýchly dekódovací algoritmus originálne navrhnutý v roku 1967 na optimálne dekódovanie konvolučných kódov [60]. Konvolučné kódy [61] patria v súčasnosti medzi najčastejšie používané kódy pre protichybové zabezpečenie a predovšetkým v oblasti mobilnej telekomunikačnej techniky tvoria prakticky štandardný algoritmus. Ďalšou typickou aplikáciou VA je ekvalizácia prenosového kanálu, ktorá sa taktiež široko využíva v telekomunikačnej technike, ale aj v menej tradičných oblastiach

ako napr. v riadiacich jednotkách pevných diskov. Význam VA v týchto oblastiach je porovnateľný s významom FFT v spektrálnej analýze. Tento rekurzívny algoritmus vyhľadáva minimálnu cestu v tzv. *trellise*[16], pričom základnou operáciou na najnižšej úrovni algoritmu je operácia tzv. *Viterbiho motýlika* (Viterbi butterfly) [62] znázornená na obr. 4.4 , ktorá realizuje operáciu

$$J_{ABCD\ldots 0} = \min\left(J_{0\,ABCD\ldots} + p_{00}, J_{1\,ABCD\ldots} + p_{10}\right) \tag{4.1}$$

$$J_{ABCD\ldots 1} = \min\left\{J_{0\,ABCD\ldots} + p_{01},\ J_{1\,ABCD\ldots} + p_{11}\right\} \tag{4.2}$$

pričom $J_{ABCD\ldots X}$ a $J_{YABCD\ldots}$ sú tzv. *akumulované metriky* v stavoch $ABCD\ldots X$, $YABCD\ldots$, $X, A, B, C, D\ldots, Y \in \{0,1\}$ a motýlik tvoria stavy, ktoré majú rovnaké hodnoty $ABCD\ldots$



Obr. 4.4 Štruktúra základného bloku Viterbiho algoritmu – Viterbiho motýlika

Výrobcovia DSP pridali do inštrukčnej sady nové inštrukcie (napr. VSL pre DSP563xx a DSP566xx [62]), ktoré umožnili zrýchliť spracovanie Viterbiho motýlika. Niektorí výrobcovia (napr. TI v TMS320C54x [63], Lucent v DSP16xxx [66]) pridali nové bloky priamo do dátových ciest. Ich význam je závislý na type procesora, cieľ je však rovnaký – zrýchlenie VA. Využitím týchto optimalizovaných inštrukcií resp. rozšírení dátových ciest je možné dosiahnuť zrýchlenie najčastejšie sa opakujúcich segmentov kódu (napr. na TMS320C54x je možné realizovať algoritmus Viterbiho motýlika počas 4 taktov [64]).

c) Špeciálne aritmetické režimy

Okrem podpory blokovej pohyblivej rádovej čiarky, ktorá sa využíva na zvýšenie presnosti výpočtu predovšetkým FFT na 16 a dokonca aj 24-bitových DSP a dátové cesty ich podporujú už od 2. generácie DSP, nové DSP vylepšujú aj podporu výpočtov *v dvojnásobnej presnosti*. Vo všeobecnosti platí, že tieto výpočty vyžadujú zvýšený počet inštrukčných cyklov. Cieľom nových inštrukcií je tento počet znížiť. U 16-bitových DSP bola táto podpora od počiatku značne prepracovaná, ukazuje sa však, že aj v oblasti 24-bitových DSP sa táto podpora stále vylepšuje. Napr. u radu DSP5600x bolo potrebné prepnúť dátové cesty do špeciálneho režimu dvojnásobnej presnosti (čo samozrejme vyžaduje dodatočné inštrukcie) u novšieho radu DSP563xx je síce z dôvodu spätnej kompatibility tento režim zachovaný, boli však dodané nové inštrukcie pre aritmetiku v dvojnásobnej presnosti, ktoré pracujú bez nutnosti prepnutia do režimu s dvojnásobnou presnosťou.

Aj keď saturačná aritmetika je stále podporovaná ako základný aritmetický režim nových DSP, v oblasti telekomunikačnej techniky sa veľmi často *používa klasická*

[16] Jeden zo spôsobov popisu konvolučných kódov.

aritmetika s pretečením. Napríklad v oblasti zdrojového kódovania rečových signálov existujú štandardné kompresné algoritmy (rôzne varianty CELP kodekov), ktorých implementácie musia poskytovať rovnaké výsledky (s bitovou presnosťou) ako referenčný model podľa normy. Keďže normy typicky používajú štandardnú aritmetiku, čo výrazne komplikuje programovanie klasických DSP, výrobcovia DSP začali v najnovších DSP podporovať nové aritmetické režimy, ktoré využívajú štandardnú 16-bitovú aritmetiku s pretečením (v prípade 24-bitových DSP teda dochádza dokonca k strate ôsmych najmenej významových bitov).

d) Pridanie druhej jednotky MAC

Keďže MAC operácia je najvýkonnejšia a najčastejšie využívaná inštrukcia v DSP, niektorí výrobcovia sa snažia zvýšiť výkon DSP integrovaním *sekundárnej* jednotky MAC. Napr. firma DSPGroup v rozšírenom jadre Teak DSP využilo sekundárnu jednotku MAC [65]. Podobne DSP16xxx od firmy Lucent využíva duálnu MAC architektúru [66]. Tento DSP (bol prvým po oddelení firmy Lucent od AT&T) však už využíva aj ďalšie výkonné rozšírenia a naznačuje orientáciu na širšie využívanie paralelizmu. Pridanie sekundárnej MAC jednotky je tak možné považovať len za *prechodný stupeň* k vývoju výkonnejších DSP, z komerčného hľadiska však môže v čase uvedenia zabezpečiť výrobcom DSP určitú výhodu na trhu.

e) Ďalšie možné trendy

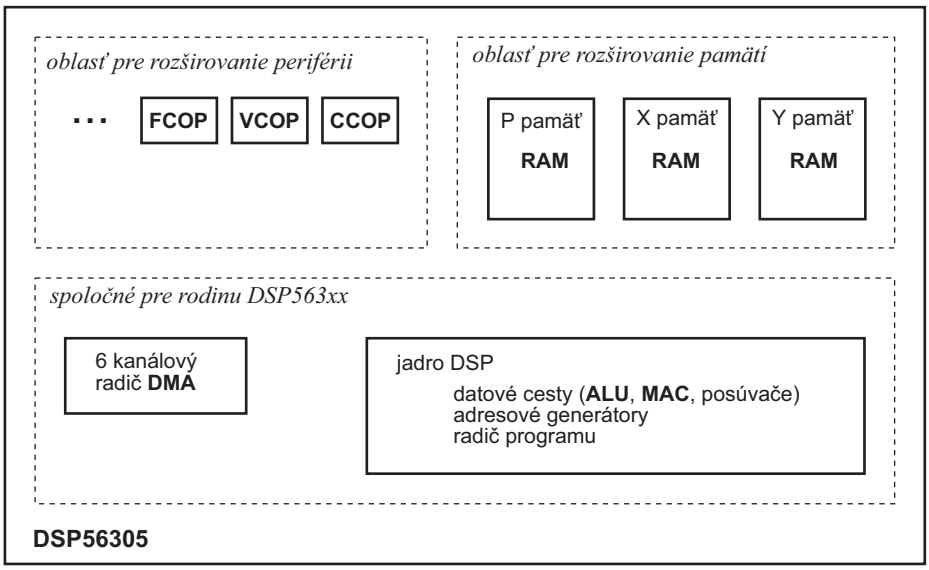
Jedným z možných aritmetických režimov by mohla byť podpora *výpočtov v konečných (Galoisových) poliach*. Táto aritmetika sa využíva predovšetkým v blokových zabezpečovacích kódoch (napr. Read-Solomonove kódy, BCH kódy…), ktoré sú často využívané v moderných telekomunikačných aplikáciách. Výpočty v konečných poliach je principiálne možné realizovať pomocou ľubovoľných programovateľných procesorových architektúr, pokiaľ však neexistuje podpora na úrovni technických prostriedkov, je rýchlosť výpočtu nízka.

Jedným z aktuálnych štandardov, ktoré využívajú túto technológiu a mohli by v budúcnosti ovplyvniť architektúry komerčných DSP sú modemy xDSL [67]. Pre implementáciu týchto modemov na programovateľných DSP je však potrebný podstatne vyšší výpočtový výkon ako poskytujú súčasné rozšírené DSP. Existujú prototypy experimentálnych programovateľných architektúr DSP optimalizovaných pre implementáciu modemov xDSL, ktoré dosahujú potrebnú výkonnosť s využitím paralelizmu. Jedna z prognóz autorov čipu je zahrnutie podpory aritmetiky v Galoisových poliach priamo do aritmetických režimov [68], ktoré by mali dátové cesty podporovať. Aj keď zatiaľ nie sú komerčne dostupné univerzálne DSP s podporou aritmetiky v Galoisových poliach, je zahrnutie tejto podpory v budúcnosti veľmi pravdepodobné.

4.1.3 DSP KOPROCESORY

Využitie *koprocesorov* v mikroprocesorovej technike je veľmi časté. Typickým príkladom sú numerické koprocesory v oblasti personálnych počítačov (napr. Intel x87), jednočipových mikropočítačov (napr. Siemens 8xC537) alebo šifračných koprocesorov v čipoch pre čipové karty [69]. Z principiálneho hľadiska nie je podstatné, či je koprocesor umiestnený na čipe, alebo je tvorený samostatným čipom. Jeho hlavnou úlohou je poskytnutie zvýšeného výpočtového výkonu v oblasti, v ktorej hlavný procesor (alebo jeho jadro) nie je dostatočne výkonný. Optimálne je, pokiaľ koprocesor tvorí samostatnú časť, ktorá môže pracovať paralelne s procesorom (alebo jeho jadrom).

Vzhľadom na uvedené trendy bolo len otázkou času, kedy sa objavia v čipoch DSP samostatné DSP koprocesory. Značným prekvapením však bolo, že medzi prvými bol tzv. *filtračný koprocesor*, ktorý umožňoval realizovať algoritmus FIR filtra (t.j. algoritmus, pre ktorý boli DSP optimalizované od počiatku ich vývoja). DSP koprocesory široko využíva v DSP predovšetkým firma Motorola a jej procesor DSP5305[17] patrí v súčasnosti z pohľadu koprocesorov k najvýkonnejším DSP. DSP56305 obsahuje tri koprocesory – filtračný koprocesor (FCOP), *Viterbiho koprocesor* (VCOP) a *cyklický koprocesor* (CCOP). Bloková štruktúra DSP56305 je zobrazená na obr. 4.5 .



Obr. 4.5 DSP koprocesory v štruktúre DSP56305

Aj keď DSP56305 je optimalizovaný pre implementáciu algoritmov, ktoré sú používané v systéme GSM, jednotlivé koprocesory sú značne univerzálne a môžu byť využité aj v iných algoritmoch ČSS. Vysokú výkonnosť DSP56305 je možné využiť predovšetkým v bázových staniciach mobilných telekomunikačných sieti, kde je možné v jednom procesore súbežne spracovať niekoľko samostatných kanálov.

Uvedené koprocesory sú z hľadiska programovacieho modelu externé periférie, ktorých riadiace, stavové a dátové registre sú mapované do dátovej pamäťovej oblasti harvardskej architektúry a môžu plne využívať vysokú výkonnosť kanálov DMA ako aj štandardný prerušovací mechanizmus, ktorými sú procesory DSP563xx vybavené.

Jednotlivé DSP koprocesory majú nasledujúce vlastnosti [56]:

a) Filtračný koprocesor

- plne programovateľný komplexný FIR filter so 16-bitovou presnosťou[18],
- nezávislé interné pamäťové banky pre dáta (84×16 bitov) a koeficienty (42×16 bitov),
- štyri optimalizované režimy činnosti:
  režim 0 – *reálny FIR filter* s maximálne 42 reálnymi koeficientmi,
  režim 1 – *komplexný FIR filter* s maximálne 21 komplexnými koeficientmi,

[17] Príklady implementácii vybraných telekomunikačných algoritmov implementovaných na tomto type DSP sú uvedené v kapitole 6 a preto sú vlastnosti koprocesorov popísané podrobnejšie.

[18] Filtračné koprocesory na čipoch DSP56307 a DSP56311 majú vylepšené (enhanced) FCOP – EFCOP s 24-bitovou presnosťou a podporou pre IIR filtre [57].

režim 2 – *komplexný FIR filter* generujúci alternujúco čisto reálne a imaginárne výstupné vzorky, v systéme GSM využívaný na realizáciu *prispôsobeného filtra* (MF – Matched Filter),
režim 3 – *komplexná korelácia* komplexnej postupnosti s postupnosťou obsahujúcou iba čisto reálne alebo čisto imaginárne koeficienty, v systéme GSM využívaná na výpočet vzájomnej korelácie medzi prijatou postupnosťou a preddefinovanou trénovacou postupnosťou,

- dva výstupné decimačné režimy:
  bez decimácie,
  decimácia s faktorom 2.

b) Viterbiho koprocesor

- podpora VA pre kanálové kódovanie a ekvalizáciu kanálu,
- využitie tzv. Ungerboeckovej metriky na ekvalizáciu kanálu [155],
- podpora trellisov s 8, 16, 32 a 64 stavmi,
- pevná hĺbka trellisu 36 bitov,
- podpora konvolučných kódov s kódovým pomerom 1/2, 1/3, 1/4 a 1/6,
- možnosť špecifikovať počiatočný a koncový stav trellisu ako aj počiatočnú metriku,
- 8-bitový (tzv. soft) vstup pre kanálové dekódovanie,
- podpora tzv. *dierovaných* (punctured) konvolučných kódov,
- možnosť počas dekódovania určiť počet opravených chýb,
- možnosť prístupu k okamžitým hodnotám metriky (vhodné pre adaptívne algoritmy).

c) Cyklický koprocesor

- obsahuje 4 posuvné registre s lineárnou spätnou väzbou a umožňuje ich vhodným prepojením a riadením generovať pseudonáhodné postupnosti pre prúdové šifry,
- podporuje kódovanie a dekódovanie tzv. *Fire kódov* [179], optimalizovaných pre opravy zhlukových chýb s možnosťou voľby generačného polynómu až do stupňa 48,
- podporuje výpočet CRC syndrómu pre generačný polynóm až do stupňa 48.

Tieto vlastnosti naznačujú, že koprocesory poskytujú podporu pre širokú triedu algoritmov ČSS využívaných v telekomunikačnej technike. Pri ich nasadení je samozrejme potrebné preštudovať detailnú dokumentáciu, avšak bez solídnych základov z oblasti teórie ČSS je ich využitie v praxi veľmi problematické. Tejto problematike je venovaná časť kapitoly 6.

4.2 DSP S PARALELNÝM VYKONÁVANÍM INŠTRUKCIÍ

Do tejto kategórie patria v súčasnosti najvýkonnejšie jednočipové DSP a všetci hlavní výrobcovia DSP už ohlásili, prípadne vyrobili produkty založené na tomto princípe. Tieto DSP využívajú *paralelizmus na úrovni inštrukcií* (ILP – Instruction Level Parallelism).

Technické prostriedky môžu využívať ILP rôznymi spôsobmi:

- niekoľko *rôznych* funkčných jednotiek v procesore môže pracovať paralelne,
- zvýšením počtu *rovnakých* funkčných jednotiek je možné ďalšie zvýšenie paralelizmu,
- funkčné jednotky môže využívať *zreťazenie.*

4.2.1 ZÁKLADNÁ KLASIFIKÁCIA

Z pohľadu riadenia procesorov, ktoré využívajú ILP je možné procesory rozdeliť do dvoch základných skupín:

a) Superskalárne procesory

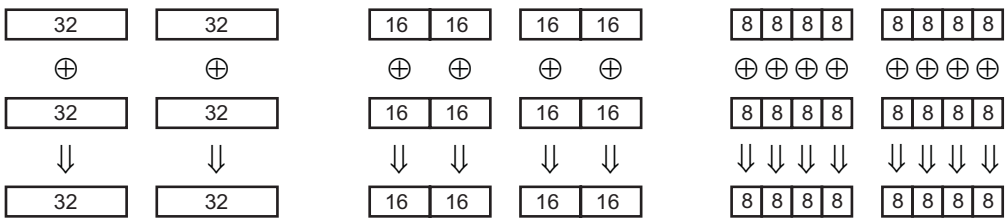
Do tejto kategórie patrí väčšina moderných výkonných mikroprocesorov pre všeobecné použitie. Riadenie poradia a paralelizmu vykonávaných inštrukcií je u týchto procesorov realizované technickými prostriedkami umiestnenými priamo na čipe vo forme *plánovacej jednotky* (scheduling hardware) a je realizované počas behu programu. Tento princíp umožňuje spracovanie pôvodne sekvenčného kódu a teda zachovanie *spätnej kompatibility* s procesormi nižších generácii, čo je napr. v oblasti osobných počítačov kľúčová požiadavka. Nevýhodou tejto triedy procesorov je zložitá konštrukcia plánovacej jednotky, ktorá často využíva napr. *špekulatívne vykonávanie inštrukcií*, ktoré môže viesť z hľadiska systémov pracujúcich v reálnom čase k neakceptovateľným oneskoreniam.

b) Procesory s veľkou dĺžkou inštrukčného slova (VLIW processors)

V tejto triede procesorov je plánovanie poradia a paralelizmu inštrukcií realizované počas prekladu a riadiace jednotky na čipe sú oproti superskalárnym procesorom podstatne jednoduchšie. Oproti superskalárnym procesorom je ich hlavnou nevýhodou nezabezpečenie spätnej kompatibility (kód je potrebné pre rôzne VLIW procesory prekladať) a nemožnosť dynamickej zmeny plánovania poradia a paralelizmu inštrukcií v závislosti na aktuálnych podmienkach, čo napr. umožňuje v niektorých prípadoch superskalárnym procesorom využiť skrytý paralelizmus algoritmov.

Z pohľadu algoritmov ČSS, ktoré využívajú relatívne *statické riadenie programu* je aplikácia VLIW architektúry v DSP procesoroch veľmi výhodná. Naviac otázka spätnej kompatibility binárneho kódu má v oblasti DSP malý význam a optimalizácia a preklad kódu sa stali v tejto triede procesorov prakticky štandardným vývojovým postupom.

Ďalšou perspektívnou modifikáciou, ktorá je využívaná v oblasti superskalárnych aj VLIW procesorov je využitie paralelizmu na báze *architektúry SIMD*, ktoré je v moderných DSP realizované predovšetkým vo forme *SIMD inštrukcií* [70]. Tieto inštrukcie sú efektívnym a kompaktným spôsobom reprezentácie inštrukcií, ktoré realizujú rovnaké operácie nad rôznymi dátami. V minulosti boli využívané predovšetkým v oblasti špeciálnych vektorových počítačov. V súčasnosti sa SIMD inštrukcie využívajú aj priamo v rámci procesorov predovšetkým pri spracovaní multimediálnych dát s menšou dĺžkou slova (typicky 8 alebo 16 bitov), ktoré sú zoskupené do slov väčších rozmerov (niekedy označovaných ako *kontajnery*) a využitím špeciálneho režimu aritmetickej jednotky, ktorá realizuje paralelne požadované operácie nad celým slovom, čo je znázornené na obr. 4.6 .



Obr. 4.6 Princíp využitia SIMD inštrukcií v aritmetických operáciách

Dĺžka slova (kontajnera) je typicky 32 resp. 64 bitov, pričom aritmetické jednotky a registre príslušnej dĺžky sú už zvyčajne v rámci existujúcich dátových ciest obsiahnuté napr. v rámci podpory aritmetiky s dvojnásobnou presnosťou. Tieto SIMD rozšírenia sa zvyknú označovať aj pojmom *multimediálne rozšírenia* prípadne tiež *zbalená aritmetika* (packed arithmetic) a sú využívané napr. v procesoroch INTEL (MMX inštrukcie), SUN (VIS inštrukcie) a HP (PA-RISC multimedia extensions).

Z pohľadu DSP je zaujímavé porovnanie *klasickej* VLIW architektúry s *klasickými* SIMD rozšíreniami.

Nevýhody VLIW architektúry oproti SIMD rozšíreniam:

- ***Funkčné jednotky VLIW procesora sú drahšie***. SIMD rozšírenia využívajú štruktúru existujúcich dátových ciest (predovšetkým v rámci aritmetickej jednotky), ktoré vyžadujú len malú modifikáciu. VLIW architektúra vyžaduje na dosiahnutie tej istej úrovne ILP niekoľko funkčných jednotiek s plnou presnosťou, čo vyžaduje väčšiu plochu čipu.
- ***Kódovanie VLIW inštrukcií je náročnejšie na pamäť***. Pomocou SIMD rozšírenia je možné reprezentovať jedinou inštrukciou *S* nezávislých operácii, pričom *S* je počet zbalených slov. V klasickej VLIW architektúre je na vykonanie rovnakého počtu operácií potrebných minimálne *S* inštrukcií, ktoré sú v optimálnom prípade umiestnené v jedinej VLIW inštrukcii (pokiaľ má VLIW procesor dostatočný počet funkčných jednotiek). To vyžaduje podstatne väčšiu inštrukčnú pamäť, čo je pre jednočipové aplikácie značná nevýhoda. Techniky *kompresie inštrukčnej pamäte* používané vo VLIW DSP sú jednou z metód, ktoré sa snažia túto základnú nevýhodu klasickej VLIW architektúry eliminovať.

Výhody VLIW architektúry oproti SIMD rozšíreniam:

- ***VLIW nevyžaduje explicitné určenie oblastí ILP***. V prípade SIMD rozšírení je naopak potrebné identifikovať tieto oblasti (často ručne) a realizovať volanie SIMD inštrukcií.
- ***VLIW inštrukcie sú flexibilnejšie***. Za cenu zvýšených nárokov na dekódovacie obvody a programové pamäte VLIW architektúry nekladú obmedzenia na typy operácií, ktoré môžu byť realizované paralelne, čo je naopak kritické v prípade SIMD rozšírení, kde je možné paralelne vykonávať len určité typy operácií. Táto vlastnosť VLIW architektúry je výhodná predovšetkým v prípade algoritmov, ktoré nemajú regulárnu štruktúru.

Superskalárna architektúra je používaná v oblasti DSP len vo výnimočných prípadoch (napr. ZSP16400 od firmy ZSP [71]) a v ďalšej časti nebude podrobnejšie analyzovaná. V súčasnosti najprogresívnejší smer v oblasti programovateľných DSP je využitie VLIW architektúry, modifikovanej s cieľom dosiahnuť úsporu programovej pamäte. SIMD rozšírenia sú typicky často využívané na zvýšenie výkonnosti klasických mikroprocesorov prípadne aj VLIW DSP.

4.2.2 ARCHITEKTÚRA VLIW

Táto architektúra bola pôvodne navrhnutá a využívaná v oblasti superpočítačov. V oblasti jednočipových procesorov bola VLIW architektúra prvýkrát využitá v roku 1995 v multimediálnych procesoroch Trimedia [72] a MPACT [73] od firiem Philips resp. Chromatics. Aj keď multimediálne procesory neboli pre trh klasických a rozšírených DSP priamym konkurentom (boli orientované na odlišný segment trhu), uvedenie TMS320C62xx firmy TI v roku 1997 znamenalo rádový výkonnostný skok v oblasti

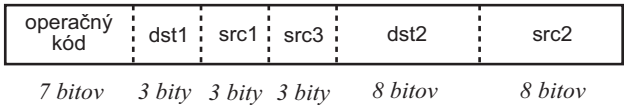
univerzálnych DSP a naznačilo trend, ktorý nasledovali (a stále nasledujú) aj ďalší výrobcovia.

4.2.2.1 ZÁKLADNÝ PRINCÍP VLIW ARCHITEKTÚRY

Architektúry DSP boli medzi prvými programovateľnými súčiastkami, ktoré využívali *dlhé inštrukčné slovo* (LIW – Long Instruction Word) [74]. Tento prístup vychádzal z horizontálneho mikroprogramovania [75], ktoré sa používalo v oblasti mikroprocesorových rezov. V architektúrach, ktoré využívajú LIW má procesor *jeden programový čítač*, pričom *inštrukcia riadi niekoľko* funkčných jednotiek. V jednom inštrukčnom cykle klasický DSP umožňuje prístup k dvom pamäťovým miestam, vykonáva MAC operáciu a modifikuje obsahy registrov v adresových aritmetických jednotkách, pričom napr. LIW inštrukcia 32-bitového procesora TMS320C30

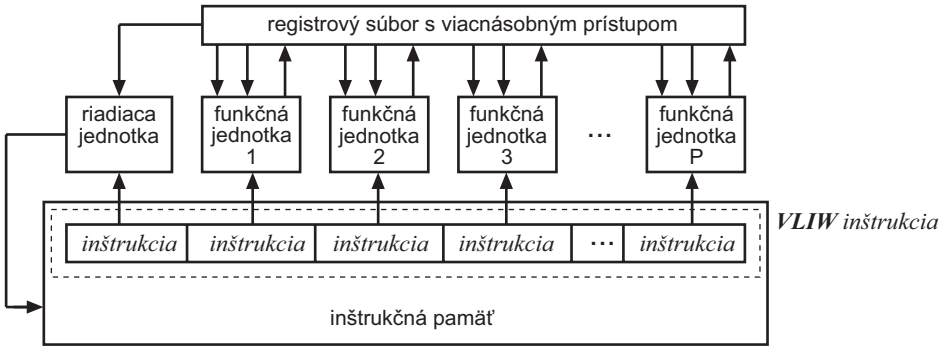
```
        MPYI3     src2, src1, dst1
||      STI       src2, dst2
```

má formát zobrazený na obr. 4.7 [74], pričom dĺžka jednotlivých polí inštrukcie súvisí s počtom a typom registrov procesora ako aj s jeho architektúrou. Táto inštrukcia realizuje paralelné vykonanie (symbolicky označované znakom ||) celočíselného násobenia (MPYI3) a celočíselného presunu (STI) a v závislosti na forme parametrov src a dst aj operácie v adresových aritmetických jednotkách a teda jedna inštrukcia LIW je využívaná viacerými funkčnými jednotkami.



Obr. 4.7 Formát inštrukcie MPYI3 || STI procesora TMS320C30

Pokroky v technológii VLSI umožnili zvýšiť počet funkčných jednotiek nad rámec využiteľný v tradičných DSP. VLIW DSP zvyšujú využívanie paralelizmu priradením samostatnej časti (slotu) v celkovej inštrukcii pre každú funkčnú jednotku, čo je principiálne znázornené na obr. 4.8 .



Obr. 4.8 Princíp architektúry VLIW

Klasická architektúra VLIW je z pohľadu DSP zaujímavá predovšetkým z týchto dôvodov [70]:

a) Využívanie paralelizmu

Inštrukcie v jednotlivých slotoch VLIW inštrukcie sú vyberané, dekódované a vykonávané paralelne, čo umožňuje výrazné zvýšenie výkonnosti VLIW DSP. Inštrukcie v jednotlivých slotoch sú vykonávané v samostatných funkčných jednotkách. Počet funkčných jednotiek je v súčasnosti typicky okolo 10 (TMS320C62x má 8), pričom existujú aj multimediálne procesory s 27 funkčnými jednotkami (Trimedia TM-1000 [74]). Dĺžka VLIW inštrukcie súčasných jednočipových VLIW DSP – rádovo stovky bitov (napr. $32 \times 8 = 256$ bitov pre TMS320C62x) je síce z pohľadu klasických jednočipových procesorov vysoká, je však potrebné zdôrazniť, že v oblasti superpočítačov je bežne využívaná dĺžka inštrukčného slova niekoľko tisíc bitov, takže je možné očakávať, že v budúcnosti by sa dĺžka inštrukčného slova v prípade jednočipových VLIW DSP ešte mohla zvýšiť.

b) Využívanie zreťazenia

Taktovacie frekvencie VLIW DSP majú zvyčajne hodnotu[19] niekoľko stoviek MHz, čo vyžaduje široké využívanie zreťazenia. Naviac VLIW DSP procesory typicky využívajú architektúru 1X podobne ako rozšírené DSP.

c) Riadenie (plánovanie) súbežnosti

V prípade architektúry VLIW je analýza a plánovanie paralelizmu realizované *počas prekladu*, čo výrazne zjednodušuje technickú konštrukciu VLIW DSP a umožňuje dosahovať vysoký výpočtový výkon pri relatívne malých nárokoch na zložitosť čipu.

d) RISC inštrukcie

Inštrukcie v jednotlivých slotoch *klasickej architektúry* VLIW predstavujú jednoduché a nezávislé inštrukcie, ktoré vychádzajú z princípu *redukovanej inštrukčnej sady* [76], [77]. Tento typ inštrukcií uľahčuje preklad programov z vyšších programovacích jazykov, čo je vzhľadom na zložitosť programovania veľmi dôležité.

Podobne ako v klasických RISC procesoroch sú aj vo VLIW DSP aritmetické operácie realizované ako operácie typu register-register nad *registrami zo súboru registrov* a prístup do pamäte je realizovaný samostatnými inštrukciami.

e) Ortogonalita architektúry

Ľubovoľný register zo súboru registrov môže byť operandom v ľubovoľnej inštrukcii. Aj keď funkčné jednotky v architektúre VLIW DSP nie sú identické, ich štruktúra vychádza z princípu, že najfrekventovanejšie inštrukcie sa môžu realizovať vo väčšine funkčných jednotiek. Veľmi často sú funkčné jednotky zoskupené do dvoch, prípadne viacerých *identických* dátových ciest, čo zvyšuje pravdepodobnosť výskytu voľnej funkčnej jednotky, ktorá môže danú inštrukciu vykonať.

f) Využitie prekladačov

Aj keď to znie na prvý pohľad paradoxne, jedným z cieľov VLIW DSP je umožniť písanie (efektívnych) programov vo vyšších programovacích jazykoch. Aj keď je architektúra VLIW DSP zložitejšia ako architektúra klasických DSP, využívanie RISC inštrukcií a ortogonalita architektúry umožňuje prekladačom (typicky z jazyka C, prípadne

[19] Najnovšia ohlásená verzia TMS320C6203 využíva taktovaciu frekvenciu 300 MHz.

C++) založených na najnovších *optimalizačných algoritmoch*, dosahovať *vysokú účinnosť* generovaného kódu, ktorá je v rámci klasických DSP nedosiahnuteľná.

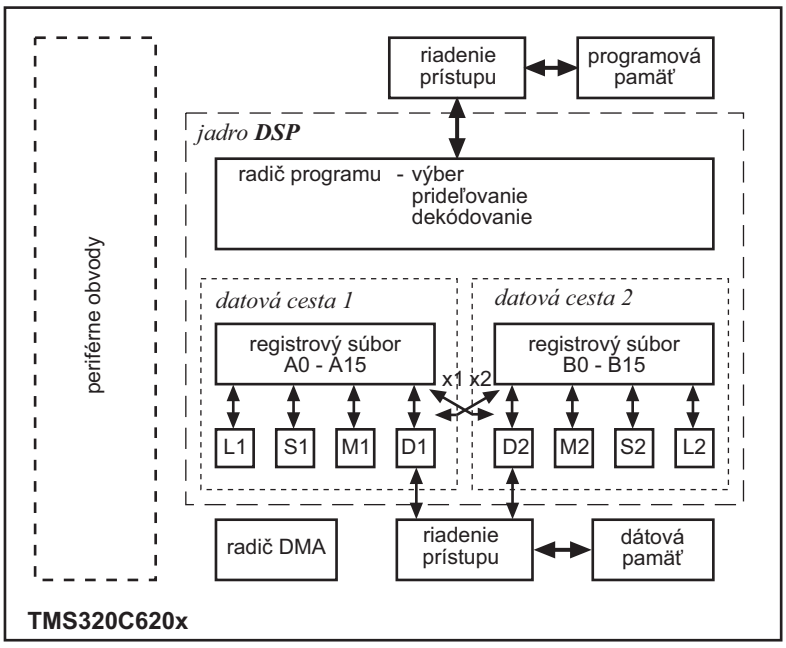
4.2.2.2 PRÍKLADY ARCHITEKTÚR VLIW DSP

Aj keď jednotliví výrobcovia VLIW DSP využívajú základný princíp architektúry VLIW, snažia sa eliminovať predovšetkým jej základnú nevýhodu – výrazné zvýšenie pamäťových nárokov, čo je v oblasti architektúr jednočipových DSP stále jeden z kritických parametrov. Ako najnovšie architektúry DSP, VLIW DSP odrážajú aj najnovšie technologické pokroky a naznačujú zaujímavé koncepčné trendy, ktoré majú pre ďalší vývoj DSP zásadný význam. V tejto podkapitole sú opísané jadrá komerčne dostupných alebo ohlásených DSP založených na architektúre VLIW s cieľom tieto trendy naznačiť.

4.2.2.2.1 VELOCITI – TEXAS INSTRUMENTS

Do kategórie VLIW DSP, ktoré TI označuje termínom *VelociTI architektúra*, patria DSP z rodiny TMS320C6x, ktoré z pohľadu klasických DSP (už) nemajú štandardnú MAC jednotku ani špecializované adresové aritmetické jednotky [79]. Ako prvý bol dostupný DSP s pevnou rádovou čiarkou – TMS320C6201, ktorý má 8 nezávislých funkčných jednotiek a pri taktovacej frekvencii 200 MHz umožňuje realizovať 1600 MIPS (alebo 200 VLIW MIPS). Ďalším členom bol DSP s pohyblivou rádovou čiarkou TMS320C6701, ktorý je objektovo aj pinovo kompatibilný s TMS320C6201 a obsahuje rovnakú množinu 8 funkčných jednotiek z ktorých je 6 rozšírených o podporu aritmetiky v pohyblivej rádovej čiarke.

Štruktúra dátových ciest procesora TMS320C62x je znázornená na obr. 4.9 [79]. Je zložená z dvoch identických dátových ciest, z ktorých každá je tvorená funkčnými jednotkami L, S, M a D. Každá dátová cesta obsahuje šestnásť 32-bitových registrov. Registrový súbor má 16 portov (10 čítacích a 6 zapisovacích), pričom každá z dátových ciest má prístup k operandom zo susedného registrového súboru (porty 1X, 2X).



Obr. 4.9 Štruktúra dátových ciest procesora TMS320C62x od firmy Texas Instruments

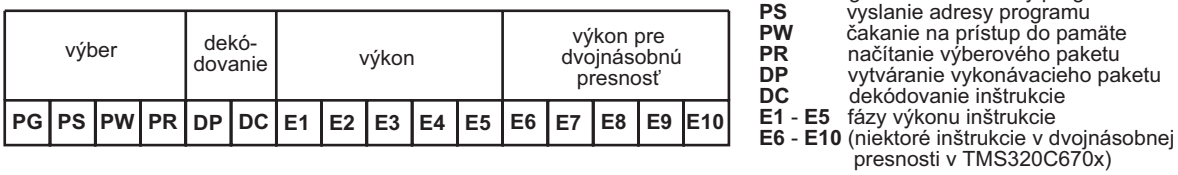
Každá z funkčných jednotiek môže vykonávať 32-bitové celočíselné operácie. Jednotky S a L môžu naviac vykonávať 40-bitové operácie a minimalizovať tak problémy s pretečením pri operáciách MAC. Väčšina funkčných jednotiek je ďalej rozdelená na subjednotky, ktoré vykonávajú rôzne činnosti uvedené v tab. 4.1 [79].

Tab. 4.1 Štruktúra funkčných jednotiek TMS320C62x

Jednotka L	Jednotka S	Jednotka D	Jednotka M
Sčítanie	Sčítanie	Sčítanie	Násobenie
Logické operácie	Logické operácie	Prístup do pamäte	
Bitové počítanie	Bitové manipulácie		
	Posuny		
	Konštanty		
	Vetvenie		

Všetky funkčné jednotky môžu začať inštrukciu v každom takte, pričom jednotlivé fázy zreťazenia zobrazené na obr. 4.10 sú rozdelené do troch kategórií:

- ***výber inštrukcie*** – 4 fázy zreťazenia
- ***dekódovanie inštrukcie*** – 2 fázy zreťazenia
- ***výkon inštrukcie*** – maximálne 10 fáz zreťazenia, pričom viac ako 90 % inštrukcií TMS320C62x využíva iba 5 fáz, posledných 5 fáz využívajú len inštrukcie pre podporu aritmetiky v dvojnásobnej presnosti.



Obr. 4.10 Fázy zreťazenia TMS320C62x

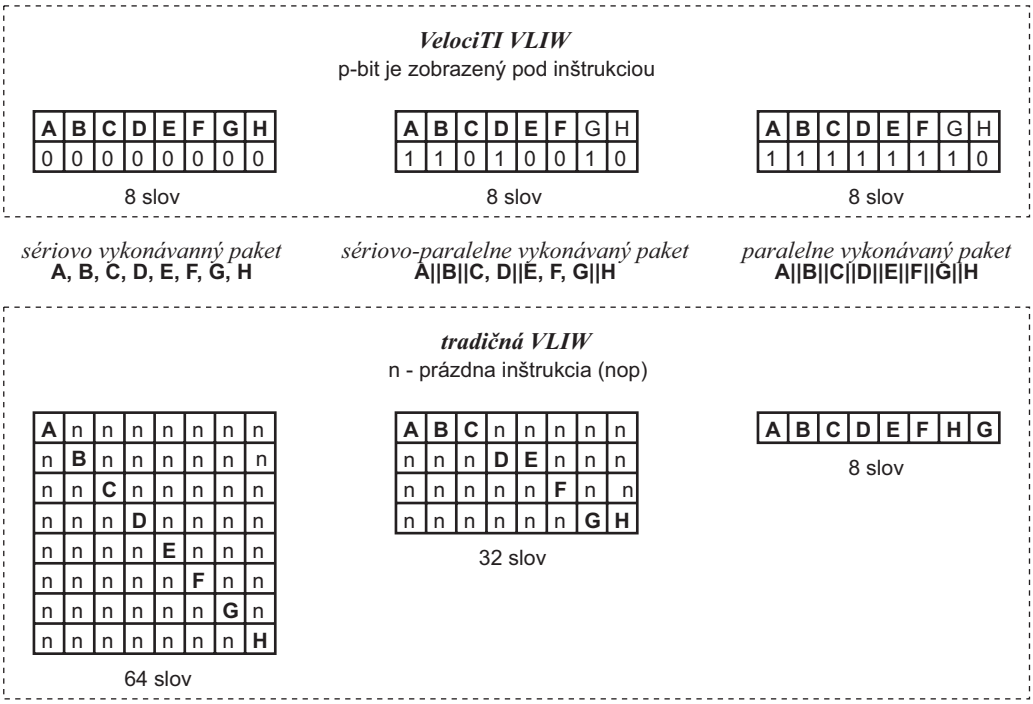
Úroveň zreťazenia je premenlivá (závislá na type inštrukcie a type funkčnej jednotky) a väčšia ako u klasických a rozšírených DSP.

Veľké nároky na programovú pamäť sú v architektúre VelociTI vyriešené metódou *kompresie inštrukcií* (instruction packing) do tzv. *inštrukčných paketov*. V terminológii VelociTI sa používajú dva druhy paketov – *výberový paket* (fetch packet) a *vykonávaný paket* (execution packet). Prvý reprezentuje skupinu inštrukcií súčasne vyberaných z inštrukčnej pamäte. Druhý reprezentuje skupinu inštrukcií vykonávaných paralelne v jadre procesora.

TMS320C62x má 256-bitové interné cesty pre výber ôsmich 32-bitových inštrukcií v každom takte. V typickej architektúre VLIW každá inštrukcia zodpovedá konkrétnej funkčnej jednotke. Ak je funkčná jednotka v určitom takte nevyužitá, v príslušnom inštrukčnom slote je umiestnená NOP inštrukcia. VelociTI architektúra využíva kompresiu výberových paketov pomocou jednoduchého kódovacieho mechanizmu, ktorý využíva tzv. p-bit (najmenej významový bit každej inštrukcie[20]). P-bit inštrukcie je nastavený ak inštrukcia začína vykonávanie paralelne s ďalšou inštrukciou, pričom typické príklady sú znázornené na obr. 4.11 [79]. Zmenšený počet výberov z programovej pamäte má okrem

[20] V jednej VLIW inštrukcii TMS320C62x je 8 inštrukcií a teda 8 p-bitov.

podstatne nižších nárokov na jej veľkosť aj vplyv na celkové zníženie príkonu. TMS320C6201 má na čipe 1 Mbit SRAM pamäte (64 Kbajtov programovej a 64 Kbajtov dátovej pamäte), pričom programová pamäť je konfigurovateľná ako programová alebo vyrovnávacia pamäť, ktorá môže obsahovať 16 K 32-bitových inštrukcií alebo 2 K 256-bitových výberových paketov.



Obr. 4.11 Princíp kompresie programovej pamäte v architektúre VelociTI

Výkonnosť procesora TMS320C62x pri realizácii niektorých typických algoritmov ČSS je dokumentovaná v tab. 4.2 [79], pričom rýchlosť procesora je približne 5 až 10-násobne vyššia ako výkonnosť DSP založených na klasickej harvardskej architektúre.

Tab. 4.2 Výkonnosť TMS320C62x/200 MHz pre vybrané algoritmy ČSS

Algoritmus	Parametre	Zložitosť (cykly)	Hodnoty parametrov	Cykly	Čas pre 200 MHz
FIR	M výstupov N koeficientov	0,5MN	M = 100 N = 32	1618	8 μs
Komplexný FIR	M výstupov N koeficientov	2MN	M = 100 N = 32	6410	32 μs
LMS FIR	M výstupov N koeficientov	1,125MN	M = 100 N = 32	5105	25,5 μs
Lattice analýza	N koeficientov	1,5N	N = 10	25	125 ns
Lattice syntéza	N koeficientov	2N	N = 10	38	190 ns
IIR	N bikvadov	4N	N = 10	56	28 ns
Komplexná FFT	N bodov	$1{,}25N\log_2 N$	N = 1024	13228	66 μs
Vektor Max	N rozmerný	0,5N	N = 100	64	320 ns
8×8 DCT	–	–	–	230	1,15 μs
8×8 IDCT	–	–	–	226	1,13 μs
Viterbiho IS54 dekodér	N bodov	66N	N = 89	5874	29,67 μs

Táto vysoká výkonnosť je dosiahnutá jednak možnosťou realizácie dvoch operácií MAC v jednom inštrukčnom cykle, zvýšenou taktovacou frekvenciou procesora a prítomnosťou ďalších funkčných jednotiek, ktoré sú *značne univerzálne*, čo je principiálne odlišné od *úzko špecializovaných jednotiek* (napr. adresové aritmetické jednotky) DSP založených na harvardskej architektúre. Vysoká ortogonalita architektúry a pokrok v technológii prekladačov sa naviac odráža vo vysokej účinnosti kódu generovaného prekladom z jazyka C čo je dokumentované v kap. 5.

Procesory radu TMS320C62x obsahujú samozrejme okrem jadra aj výkonné periférne obvody ako napr. radič DMA, rozšírené pamäťové rozhranie (extended memory interface), viackanálové sériové porty a pod. Z hľadiska výkonnosti je najrýchlejším ohláseným procesorom firmy TI v tejto kategórii TMS320C6203, ktorý je taktovaný frekvenciou 300 MHz, obsahuje 7 Mbitov SRAM pamäte, je vyrobený 0,15 μm technológiou CMOS s piatimi úrovňami metalizácie, používa napätie jadra 1,5 V a 3,3 V pre externé I/O obvody, typický príkon CPU je 0,25 W a kompletný čip (CPU + I/O) má príkon 1,5 W.

4.2.2.2.2 JADRO STAR CORE – LUCENT A MOTOROLA

TI má oproti ostatným výrobcom DSP na trhu značnú prevahu, čo je naznačené na obr. 2.6 a prejavuje sa jednak vo veľkom množstve rôznych typov DSP, ktoré TI vyrába ako aj určitým technologickým náskokom, ktorý má pred ostatnými výrobcami (uviedol napr. prvý univerzálny VLIW DSP podstatne skôr ako jeho konkurenti). Odpoveďou dvoch ďalších najväčších výrobcov DSP, firiem Lucent a Motorola bolo *vytvorenie spoločnej aliancie* v júni 1998 s cieľom vyvinúť konkurenčnú architektúru univerzálneho jadra VLIW DSP, ktoré dostalo názov architektúra Star Core 100 [80], [81]. Cieľom je použitie jadra na báze tejto architektúry v rôznych produktoch, ktoré bude Motorola a Lucent vyrábať samostatne[21]. Jadro Star Core 140 (ďalej len Star Core) je prvým ohláseným výsledkom spoločného vývoja.

Star Core je z hľadiska architektúry VLIW DSP jadro, ktoré do oblasti VLIW DSP prináša niekoľko zaujímavých myšlienok resp. prístupov:

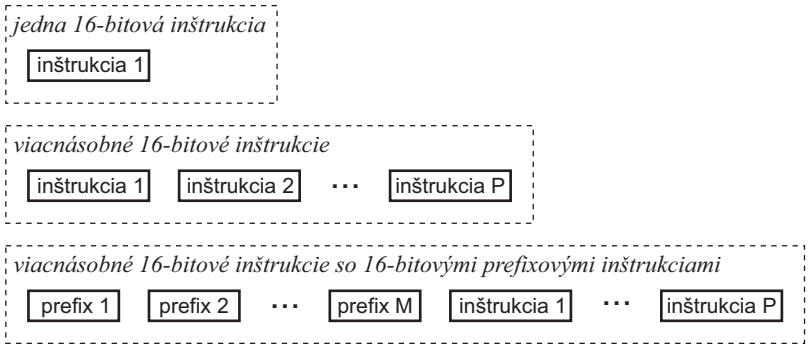
a) Minimalizácia potrebných programových pamätí

Základný problém VLIW architektúry rieši Star Core využitím inštrukcií s premenlivou dĺžkou (VLES – Variable-Length Execution Set) ktoré využívajú 16-bitové inštrukcie, čo je polovičná dĺžka oproti 32-bitovým inštrukciám vo VelociTI architektúre. Pri 16-bitovej šírke inštrukcií však nie je možné zakódovať všetky kombinácie inštrukcií a operandov, a Star Core používa premenlivý počet voliteľných „*prefixových*“ inštrukcií, ktoré sú súčasťou paralelne spracovávaných inštrukcií. Prefixové inštrukcie rozširujú funkčné možnosti základných 16-bitových inštrukcií (napr. umožnením prístupu k väčšej množine registrov). Vo všeobecnosti prefixové inštrukcie ovplyvňujú celú skupinu paralelných inštrukcií, ktoré nasledujú. Princíp VLES je znázornený na obr. 4.12 [80].

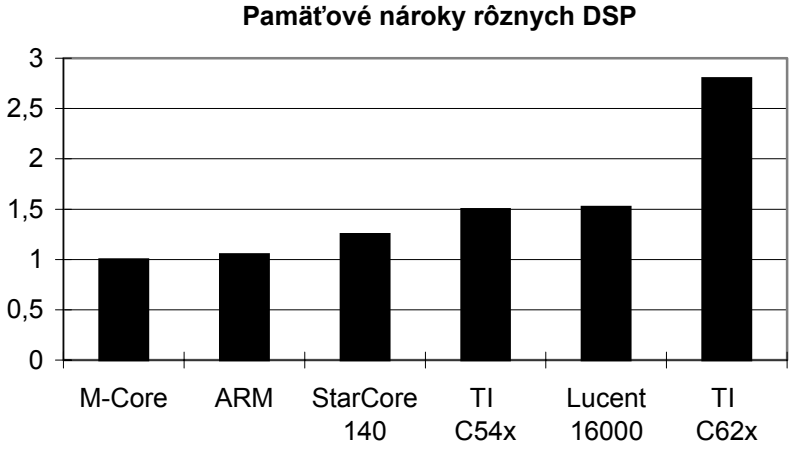
Princíp inštrukcií s premenlivou dĺžkou bol využitý už skôr v DSP16000 od firmy Lucent s cieľom kombinovať výkonnosť 32-bitových inštrukcií, ktorá je potrebná v kritických (z hľadiska rýchlosti) častiach programov s vysokou hustotou kódu a 16-bitové inštrukcie v ostatných častiach kódu. Podľa návrhárov bude Star Core dosahovať väčšiu hustotu kódu ako DSP na báze harvardskej architektúry a blížiť sa z hľadiska hustoty kódu k univerzálnym jednočipovým procesorom (M-Core, ARM), čo je naznačené na obr. 4.13

[21] Objavujú sa aj pesimistické názory, ktoré tvrdia, že práve tento fakt môže viesť ku skorému zániku aliancie, pretože vo finálnych zariadeniach majú Lucent a Motorola podobných koncových zákazníkov z oblasti telekomunikačného priemyslu.

[80] (tieto výsledky sú založené na súbore interných testov z oblasti algoritmov ČSS, kryptografických a riadiacich algoritmov používaných firmou Motorola).



Obr. 4.12 Princíp inštrukcií VLES využívaný v architektúre Start Core



Obr. 4.13 Porovnanie relatívnej hustoty kódu jednotlivých architektúr

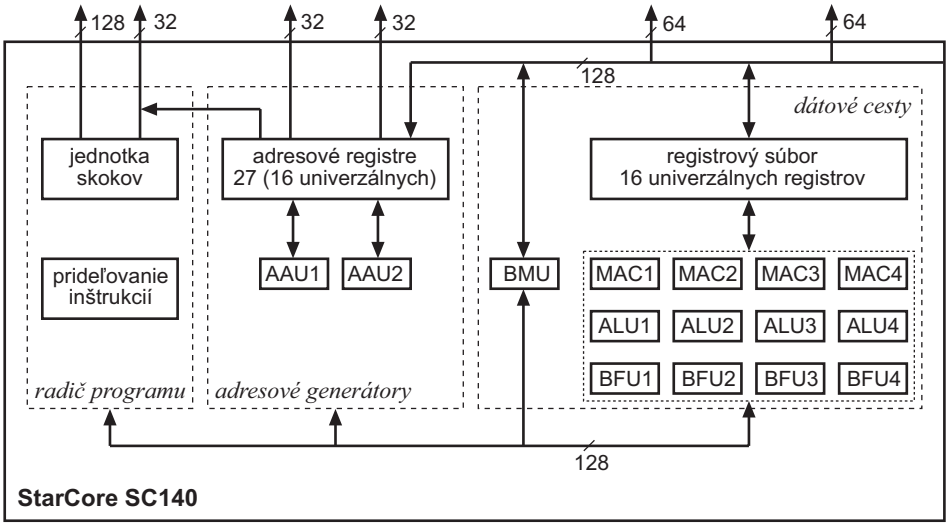
b) Škálovateľnosť (scalability) architektúry

Jedným zo základných cieľov novej architektúry Star Core je poskytnúť jadro DSP, ktoré by mohlo byť využívané v širokej triede produktov od nízkopríkonových až po vysokovýkonné jednočipové systémy. Motívom k tomuto cieľu je snaha znížiť vývojové náklady na podporu stále sa rozširujúcich aplikácii ČSS, ktoré z hľadiska výrobcov čipov neustále narastajú. Podľa dostupných informácií by *škálovateľnosť* mala zahrňovať možnosti zmeny:

- taktovacej frekvencie,
- šírky dát,
- počtu a typu funkčných jednotiek,
- počtu a šírky zberníc na čipe,
- priepustnosti pamäťového systému,
- počtu registrov,
- počtu a typu inštrukcií uskutočnených v jednom cykle.

c) Výkonná architektúra

Prvé ohlásené jadro DSP Star Core SC140 obsahuje 12 výkonných funkčných jednotiek, z toho 4 ALU, 4 BFU a 4 MAC jednotky, čo je 2-krát viac ako v konkurenčnej VelociTI architektúre. Všetky štvorice jednotiek sú *identické*, čo prispieva k vysokej ortogonalite architektúry. V architektúre sú naviac 2 špecializované adresové aritmetické jednotky AAU[22]), jedna jednotka pre manipuláciu s bitmi (BMU – Bit Manipulation Unit) a jedna jednotku vetvenia (BU – Branch Unit). Architektúra jadra Star Core je znázornená na obr. 4.14 [81]. V rámci jednej VLES inštrukcie môže jadro vykonať paralelne 6 inštrukcií, napr. 4 MAC inštrukcie a dva presuny. Táto kombinácia inštrukcií je ekvivalentná desiatim RISC inštrukciám vo VelociTI architektúre, ktorá na výpočet MAC operácie vyžaduje dve RISC inštrukcie. Predovšetkým prítomnosť štyroch MAC jednotiek umožňuje jadru dosiahnuť vyššiu relatívnu výkonnosť (v počte cyklov) oproti konkurenčným výrobkom firmy TI, čo je znázornené v tab. 4.3 [80].



Obr. 4.14 Blokový diagram jadra Start Core SC140

Tab. 4.3 Porovnanie výkonnosti jadra Star Core s produktmi TI

Algoritmus	TI C6x	TI C54x	SC 140	**SC140** oproti **C6x**	**SC140** oproti **C54x**
Reálny FIR	N/2	N	N/4	2x	4x
Komplexný FIR	2N	4N	N	2x	4x
LMS FIR	3N	3N	N	3x	3x
Bikvad (4 násobenia)	4N	5N	1,5N	2,67x	3,33x
Korelácia	N/2	2N	N/4	2x	8x
G1 vo VSELP	N	2N	N/2	2x	4x
Radix 2 FFT	4N	8N	2N	2x	4x

V každom takte je zo 16 funkčných jednotiek aktívnych *maximálne* 6, čo je z pohľadu tradičných VLIW značný odklon od pôvodnej koncepcie (*na čipe je viac paralelných jednotiek, ako môže VLES inštrukcia využiť*). Ďalšia novinka je v spôsobe prideľovania inštrukcií k jednotlivým funkčným jednotkám. Analýza a plánovanie paralelizmu je realizované počas prekladu ako u klasickej VLIW architektúry, mapovanie inštrukcií ku

[22] Napr. VelociTI architektúra nemá žiadnu špecializovanú jednotku na generovanie adries.

konkrétnym paralelným jednotkám je však realizované dynamicky počas vykonávania programu. Tento prístup je priamou podporou pre možnosť škálovateľnosti budúcich verzií procesorov, ktoré budú môcť využívať rôzny počet funkčných jednotiek.

Jadro používa 5-úrovňové zreťazenie tvorené fázami predvýberu inštrukcie, výberu inštrukcie, dekódovania/prideľovania (dispatch), generovania adresy a vykonania inštrukcie. Podľa údajov výrobcu by toto, relatívne nízkoúrovňové zreťazenie, malo prispieť k ľahšiemu programovaniu v asembleri a efektívnemu spracovaniu skokov a prerušení [81].

d) Nízkopríkonový návrh

Popri vysokej výpočtovú výkonnosti jadra bol *nízky príkon* ďalším cieľom, ktorý ovplyvňoval celkový návrh jadra Star Core. Jadro bude vyrábané u Motoroly technológiou HIP6 s rozlíšením 0,18 μm a počiatočná projektovaná hodinová frekvencia je 300 MHz, čo predstavuje výkon 1200 miliónov MACs (čo je vzhľadom na predchádzajúce úvahy približne ekvivalentné 3000 RISC MIPS). Pri napájacom napätí jadra 1,5 V (pri nešpecifikovanej veľkosti SRAM pamäte) je plánovaný príkon nižší ako 180 mW pri 300 MHz čo predstavuje hodnotu 0,1 mA/WLIW MIPS @ 1,5 V. Táto hodnota predstavuje z pohľadu jednočipových DSP procesorov špičkovú hodnotu, ktorú v súčasnosti nedosahuje žiadny DSP produkt[23]. Naviac jadro bude pracovať aj pri napätí 0,9 V, frekvencii 120 MHz a využívať príkon 0,066 mA/WLIW MIPS @ 0,9 V čo umožní napr. realizovať algoritmus GSM vokodéra s príkonom len 1,2 mW [81].

Na uvedených hodnotách sa podieľa niekoľko faktorov. Samozrejme veľký podiel má použitá 0,18μm technológia, čo je však všeobecná vlastnosť VLSI technológie. Z pohľadu architektúry jadra je to predovšetkým vďaka použitiu 16-bitových inštrukcií a princípu VLES. Naviac aj keď jadro obsahuje 16 funkčných jednotiek, je príkon aktuálnych jednotiek riadený individuálne v závislosti na ich aktuálnom využití *v rámci každého hodinového taktu*.

e) Perspektívy ďalšieho vývoja

Prvé vzorky čipov využívajúce jadro Star Core by mali byť dostupné koncom roku 1999 a plná produkcia by mala byť spustená v roku 2000. Jadro bude využívané predovšetkým v zložitejších čipoch (ASIC čipy na báze štandardných buniek) a tvoriť základ realizácie systémov na čipe. Prvým ohláseným čipom od firmy Motorola by mal byť integrovaný DSP (integrated DSP) s označením MSC8101, ktorý bude integrovať 300MHz jadro Star Core, 150MHz komunikačný procesorový modul, 300MHz EFCOP, 16-kanálový radič DMA a 4 Mbity SRAM pamäte. Jadro bude napájané napätím 1,5 V a periférne I/O rozhrania procesora napätím 3,3 V, pričom celková spotreba čipu by mala byť menšia ako 500 mW. Procesor bude optimalizovaný pre telekomunikačné aplikácie.

Vzhľadom na princíp škálovateľnosti (budúcich verzií jadier Star Core 1xx) bude možné pokryť širokú oblasť aplikácií od mobilných zariadení pre 3. generáciu mobilných systémov, až po spracovanie audio a video signálov v spotrebnej elektronike. Ceny čipov by vzhľadom na masové využitie v rôznorodých zariadeniach mali klesať. Kódová kompatibilita by mala prispieť k zníženiu vývojových nákladov u výrobcov zariadení a umožniť širšie využívanie optimalizovaných knižničných funkcií pre algoritmy ČSS.

---

[23] Napr. 16-bitový DSP TMS320UVC5402, ktorý je v súčasnosti z pohľadu príkonu jeden z najúspornejších komerčne dostupných DSP, má príkon 0,54 mW/MIPS pri napájacom napätí 1,2 V, čo zodpovedá hodnote 0,45 mA/MIPS @1,2 V. Naviac VLIW MIPS jadra Star Core je podstatne výkonnejšia ako MIPS procesora TMS320UVC5402 na báze harvardskej architektúry.

V súčasnosti sú dostupné informácie len o verzii s pevnou rádovou čiarkou a bude zaujímavé sledovať, či sa v budúcnosti objavia aj modifikácie s pohyblivou rádovou čiarkou, podobne ako to bolo v prípade VelociTI architektúry.

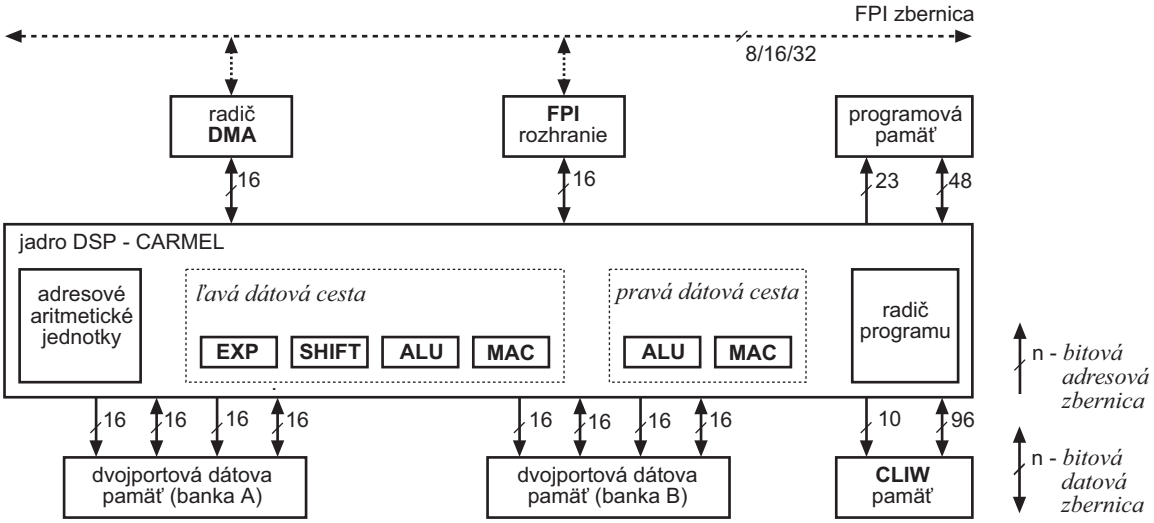
4.2.2.3 PRÍKLADY ARCHITEKTÚR VLIW DSP S ROZŠÍRENIAMI SIMD

Rozšírenia SIMD umožňujú znížiť predovšetkým nároky na programovú pamäť, čo je stále v tejto triede procesorov kľúčový faktor. V architektúre VLIW je možné využiť predovšetkým prítomnosť viacerých funkčných jednotiek (zvyčajne typu MAC alebo ALU), čo dokumentujú príklady DSP uvedené v rámci tejto podkapitoly. Tieto DSP naznačujú, že k architektúre VLIW je možné prejsť aj „plynulejším“ prechodom od architektúry DSP založených na modifikáciach harvardskej architektúry.

4.2.2.3.1 DSP JADRO CARMEL – SIEMENS

Siemens je výrobcom, ktorý v minulosti využíval na základe licencií jadrá DSP iných výrobcov – predovšetkým OakDSPCore od firmy DSP Group's, ktoré používal predovšetkým v produktoch pre mobilnú sieť GSM. Carmel je nové výkonné 16-bitové jadro DSP, ktoré Siemens vyvinul spolu s Izraelskou firmou I.C.Com, a malo by nahradiť staršie jadrá DSP predovšetkým v najnovších mobilných zariadeniach [82]. Siemens by mal na základe licenčných zmlúv poskytovať toto jadro aj ďalším výrobcom čipov a podobne ako Star Core jasne naznačuje orientáciu do oblasti produktov ASIC.

Bloková štruktúra jadra Carmel je znázornená na obr. 4.15 a je charakteristická odlišnou ľavou a pravou dátovou cestou.



Obr. 4.15 Štruktúra jadra Carmel

Jadro obsahuje 6 výkonných jednotiek, pričom v jednom takte môže jadro paralelne spracovať až 6 inštrukcií. Určitým odklonom od tradičnej VLIW architektúry je *používanie zložitejších inštrukcií*, ktoré na rozdiel od RISC inštrukcií napr. vo VelociTI architektúre, môžu vykonať aj niekoľko operácií (napr. násobenie a inkrementovanie smerníka). Dátové cesty *nie sú identické*, pričom ľavá obsahuje jednotku pre prácu s exponentom (exponent unit), posúvač, ALU a MAC jednotku. Pravá dátová cesta obsahuje len jednotky ALU a MAC. Podobne ako DSP16xxx od firmy Lucent obsahujú dátové cesty jadra Carmel aj technické prostriedky *pre podporu Viterbiho algoritmu*, čo naznačuje optimalizáciu procesora pre telekomunikačné aplikácie. Dátové cesty využívajú spoločnú banku

registrov, ktorá obsahuje šesť 40-bitových akumulátorov, 26 adresových registrov (z ktorých 16 môže byť využitých aj pre všeobecné použitie). Niektoré registre sú doplnené tieňovými registrami, čo umožňuje rýchle prepnutie kontextu procesora pri spracovaní prerušení. Oproti klasickej VLIW architektúre, ktorá pracuje s bankou registrov, umožňuje Carmel aj efektívnu *prácu priamo s pamäťou*, čo je umožnené prítomnosťou dvoch dvojportových pamäťových baniek (na obr. 4.15 označených ako banka A a banka B).

Jednotka ALU podporuje zbalenú aritmetiku v rámci jednotiek ALU nad 16-bitovými dátami, pričom jednotky MAC môžu realizovať aj samostatné operácie sčítania a odčítania, čo je pre MAC jednotky relatívne netradičná možnosť. Z pohľadu architektúry VLIW je zaujímavé predovšetkým využitie *konfigurovateľných inštrukcií s dlhým inštrukčným slovom* (CLIW – Configurable Long Instruction Words).

Princíp CLIW

Jadro Carmel podporuje inštrukcie s formátom 24 a 48 bitov, pričom môže v jednom takte spracovať jednu 24-bitovú, jednu 48-bitovú alebo dve 24-bitové inštrukcie. Väčšina inštrukcií existuje vo formáte 24 aj 48 bitov, pričom 24-bitové inštrukcie podporujú menej spôsobov adresovania a sú menej flexibilné. Pri paralelných inštrukciách je možné špecifikovať jednu 24-bitovú inštrukciu pre ľavú dátovú cestu a druhú pre pravú dátovú cestu. Tieto paralelné inštrukcie sú prvým typom VLIW inštrukcií. Druhým typom sú inštrukcie CLIW, ktoré podobne ako tradičné inštrukcie VLIW, umožňujú kombinovať maximálne 6 preddefinovaných inštrukcií do jednej CLIW *superinštrukcie*. Tradične DSP dosahovali vysokú výkonnosť akceleráciou kritických operácii (predovšetkým operácie MAC a zavedením špecializovaných MAC inštrukcií). CLIW inštrukcie posúvajú túto filozofiu o krok vpred a umožňujú programátorovi definovať určitý počet nových inštrukcií, ktoré môžu byť optimalizované pre konkrétnu aplikáciu. Hlavným motívom je zníženie veľkosti potrebných pamätí (a samozrejme zníženie príkonu čipu použitím užších zberníc).

CLIW inštrukcie majú dĺžku 144 bitov. Prvých 48 bitov je tvorených štandardnou inštrukciou (tzv. CLIW reference line), ktorá zabezpečí výber maximálne 4 operandov a výber zvyšných 96 bitov CLIW inštrukcie zo špeciálnej 96×1024-bitovej CLIW pamäte zobrazenej na obr. 4.15 .

CLIW inštrukcia definuje operácie pre maximálne 4 funkčné jednotky a dva paralelné presuny, čo je možné v jazyku asembler (ktorý má syntax veľmi podobnú jazyku C) zapísať nasledujúcim spôsobom:

```
CLIW meno_inštrukcie( ma1, ma2, ma3, ma4)          ; CLIW reference line
{
MAC1 || ALU1 || MAC2 || ALU2 || MOV1 || MOV2       ; definícia CLIW
}
```

pričom použitie CLIW inštrukcie v reálnom kóde vyzerá takto:

```
a4 = maxnum;
rep (M/2) block
{
        a01 = *(r4) - *(r0++)  ||  a21 = *(r4++) - *(r0+rn0);
        clr (a1, a3)  ||  rep ( N-1 ) single
        {
                CLIW VQ1 ( r4++, r0++, r0+rn0 )          ; použitie inštrukcie CLIW
                {
                        a01 = *ma1 - *ma2                ; s aktuálnymi parametrami
                      ||  a1 += sqr( a01 )
                      ||  a21 = *ma1 - *ma3
                      ||  a3   = sqr( a21 )
                }
        }
        a1 += sqr( a01 )  ||  a3 += sqr( a21 )
        minm( a4, a1, r0 );
        minm( a4, a3, r0+rn0);
}
```

ktorý okrem práce s CLIW inštrukciou dokumentuje aj prehľadnosť použitého asemblera.

Nízkopríkonový návrh

Jadro Carmel je optimalizované pre technológiou 0,25 μm a má príkon (bez periférii a pamäte) 200 mW@ 2,5 V pri taktovacej frekvencii 120 MHz. Čip využíva 8-úrovňové zreťazenie (bez vzájomného blokovania), čo je menej ako vo VelociTI architektúre. Z hľadiska návrhu (úroveň zreťazenia) a technologických možností (0,25 μm a 120 MHz) jadro nedosahuje špičkový výpočtový výkon a zdá sa, že hlavným cieľom je minimalizácia príkonu a umožnenie jednoduchého návrhu systémov na čipe s využitím technológie ASIC.

Perspektívy ďalšieho vývoja

Ďalšími naznačenými cieľmi bude vývoj 24-bitovej verzie, ktorá bude optimalizovaná pre spracovanie audio signálov (MPEG prípadne AC3 dekompresia) ako aj vývoj lacnejších verzií s nižšou výkonnosťou – tzv. Tiny Carmel. Podobne ako u jadra Star Core je možné očakávať väčší počet čipov využívajúcich jadro Carmel. Vzhľadom na jeho výkonnosť, ktorá je vyššia ako výkonnosť rozšírených DSP založených na harvardskej architektúre (ako napr. TMS320C54x) a možnosť zakúpenia vo forme licencie môže toto jadro získať zaujímavú časť trhu s DSP produktmi. Nezanedbateľným faktorom môže byť aj veľké potenciálne zázemie, ktorým je firma Siemens známa.

4.2.2.3.2 TIGERSHARC – ANALOG DEVICES

Firma Analog Devices je tradičným výrobcom vysokovýkonných DSP s pohyblivou rádovou čiarkou, pričom samozrejme vyrába aj úspešný rad procesorov s pevnou rádovou čiarkou (predovšetkým klasické ADSP218x a ohlásené ADSP219x).

Jej najvýkonnejšie procesory s pohyblivou rádovou čiarkou pre paralelné systémy vychádzajú z procesorov Sharc ADSP2106x, ktoré sú z hľadiska podpory pre multiprocesorové systémy priamym konkurentom napr. transputerov. Tieto DSP obsahujú na čipe typicky veľké množstvo SRAM pamätí. Po uvedení TMS320C67x firmou TI však Analog Devices stratil čelnú pozíciu v tejto oblasti[24].

[24] Aj keď vynikajúca podpora pre multiprocesorovú komunikáciu u procesorov ADSP2106x je stále významným faktorom pri výbere čipu pre stavbu paralelných systémovu ČSS [84].

Analog Devices na tento stav zareagoval vývojom druhej generácie, čipu ADSP2116x, ktorý nazval *Hammerhead* a uviedol začiatkom roku 1999. Zvýšenie výpočtového výkonu u tohto čipu je dosiahnuté jednak zvýšením taktovacej frekvencie na 100 MHz a tiež pridaním druhej paralelnej dátovej cesty. Táto druhá dátová cesta však môže byť využitá len pomocou SIMD inštrukcií paralelne s prvou, pričom podľa nezávislých testov tento čip nedosahuje výkonnosť TMS320C67x. Určitou výhodou je kompatibilta na úrovni asemblerovského kódu s ADSP2106x, pre maximálne využitie čipu však kód musí byť prepísaný s cieľom využiť SIMD inštrukcie.

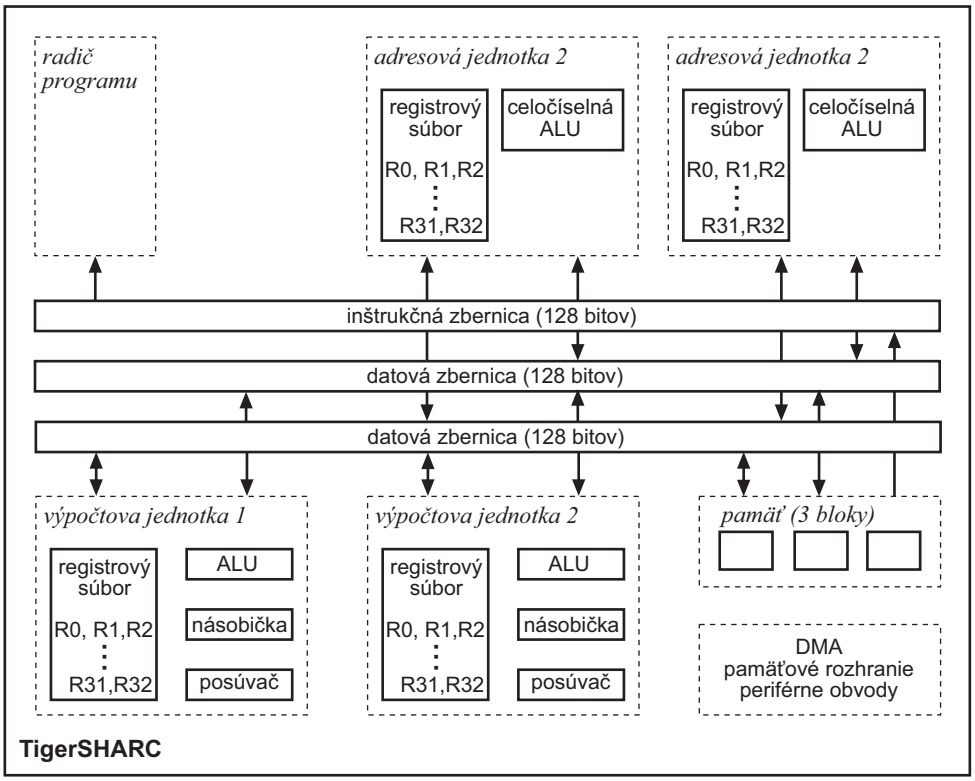
Analog Devices tiež zverejnil základné údaje o tretej generácii procesorov Sharc, ktoré nazval TigerSharc. Predpokladá sa, že vzorky tohto čipu by mali byť dostupné skôr ako produkty využívajúce jadro Star Core a úvodná taktovacia frekvencia by mala byť 250 MHz. Pri tejto frekvencii by mal čip vykonať $4\times$ ($2\times$) viac 16-bitových MAC operácií v pevnej rádovej čiarke ako TMS320C62x (jadro Star Core) s rovnakou taktovacou frekvenciou, čo je dokumentované v tab. 4.4 [83].

Tab. 4.4 Porovnanie architektúr Sharc, VelociTI a Star Core

Výrobca	Analog Devices			Lucent/ Motorola	Texas Instruments	
Typ	2106x	2116x	–	–	320C26x	320C27x
Názov	Sharc	Hammerhead	TigerSharc	StarCore	VelociTI	VelociTI
Dostupnosť	1994	1Q1999	Stred 1999	1Q2000	1Q1997	3Q1998
Kompatibilita	–	Asembler 2106x	–	–	–	Binárna s 320C62x
Architektúra	DSP	DSP+ SIMD	VLIW+ SIMD	VLIW	VLIW	VLIW
Takt. frekv.	60 MHz	100 MHz	250 MHz	300 MHz	250 MHz	167 MHz
16b MAC/s	60 miliónov	200 miliónov	2000 miliónov	1200 miliónov	500 miliónov	333 miliónov
FP MAC/s	60 miliónov	200 miliónov	500 miliónov	–	–	333 miliónov

Vysoká výpočtová výkonnosť procesora TigerSharc v oblasti aritmetiky s pevnou rádovou čiarkou je dosiahnutá SIMD rozšírením pôvodných dátových ciest pre pohyblivú rádovú čiarku. Architektúra procesora TigerSharc je znázornená na obr. 4.16 a obsahuje riadiacu jednotku, 2 adresové aritmetické jednotky, 2 výpočtové jednotky, pamäť, radič DMA a rôzne periférne jednotky.

Táto architektúra je prakticky totožná s architektúrou DSP na báze harvardskej architektúry (okrem dvoch výpočtových jednotiek, ktoré z nej vytvárajú VLIW architektúru), pričom vysoká priepustnosť v rámci čipu je podporovaná 128-bitovými zbernicami. Každá výpočtová jednotka obsahuje násobičku, ALU a posúvač, pričom tieto subjednotky môžu spracovať 32 alebo 64-bitové operandy a podporujú operácie v pevnej aj pohyblivej rádovej čiarke. V prípade 64-bitových operandov sa používajú dva spojené 32-bitové registre. Z pohľadu architektúry VLIW DSP sú najväčšou novinkou SIMD rozšírenia. Dátové presuny a niektoré výstupy z násobičky môžu byť široké až 128 bitov, čo je dosiahnuté spojením štyroch 32-bitových registrov.



Obr. 4.16 Architektúra procesora TigerSharc

SIMD rozšírenia výpočtových jednotiek

Voliteľne môže byť výpočet v obidvoch dátových cestách riadený v jednom takte *jednou (spoločnou) inštrukciou*. Z tohto pohľadu je TigerSharc SIMD procesorom. Naviac však procesor využíva ďalší *hierarchický SIMD* prístup na úrovni jednotlivých registrov, pričom môže pracovať s hodnotou v ľubovoľnom 32-bitovom registri ako s 32-bitovou hodnotou v pohyblivej rádovej čiarke vo formáte IEEE 754, jednou 32-bitovou, dvomi 16-bitovými alebo štyrmi 8-bitovými hodnotami v pevnej rádovej čiarke. Takéto hierarchické usporiadanie je bežné predovšetkým u výkonných procesorov pre všeobecné použitie, v oblasti DSP je však nezvyčajné[25] a bude široko využívané pri spracovaní multimediálnych dát akými sú zvuk, obrazy a videosekvencie.

Každá výpočtová jednotka môže realizovať v každom cykle jednu 32-bitovú operáciu MAC v pohyblivej rádovej čiarke, dve $32 \times 32 \rightarrow 64$ alebo štyri $16 \times 16 \rightarrow 32$-bitové MAC v pevnej rádovej čiarke, čo vzhľadom na dve paralelné jednotky umožňuje dosiahnuť výpočtový výkon uvedený v tab. 4.4 [83]. Internými dátovými zbernicami je možné preniesť 8 Gbajtov/s ($2\ jednotky \times 250\ MHz \times 128\ bitov$), čo predstavuje priepustnosť šestnásť 16-bitových slov/cyklus. Cez programovú zbernicu je možné preniesť paralelne štyri 32-bitové inštrukcie a tak je možné v každej jednotke realizovať paralelne napr. aritmetické operácie a posuny.

Zreťazenie procesora je 8-úrovňové, využíva vzájomné blokovanie a operácie násobenia, sčítania a čítania dát majú oneskorenie 2 cykly.

[25] Tento trend naznačuje určitú konvergenciu DSP s multimediálnymi rozšíreniami univerzálnych a multimediálnych procesorov.

Perspektívy ďalšieho vývoja

TigerSharc je prvým univerzálnym VLIW DSP, ktorý výrazne využíva SIMD rozšírenia, čo mu umožňuje dosiahnuť vysoký výpočtový výkon. Hierarchická SIMD architektúra však z pohľadu programovania znamená znásobenie problémov ktoré sú typické pre klasické DSP na báze harvardskej architektúry. Z pohľadu užívateľov bude predovšetkým dôležitá podpora v oblasti efektívnych knižničných funkcií, ktoré budú využívať rozšírenia SIMD.

Z hľadiska veľkosti pamätí a periférnych obvodov nie sú zatiaľ dostupné podrobnejšie informácie, je však možné očakávať, že čipy budú obsahovať veľké (minimálne niekoľko Megabitové) SRAM pamäte a podporu pre multiprocesorovú komunikáciu. DSP TigerSharc budú štandardne obsahovať 14-kanálový radič DMA.

4.3 MULTIMEDIÁLNE PROCESORY

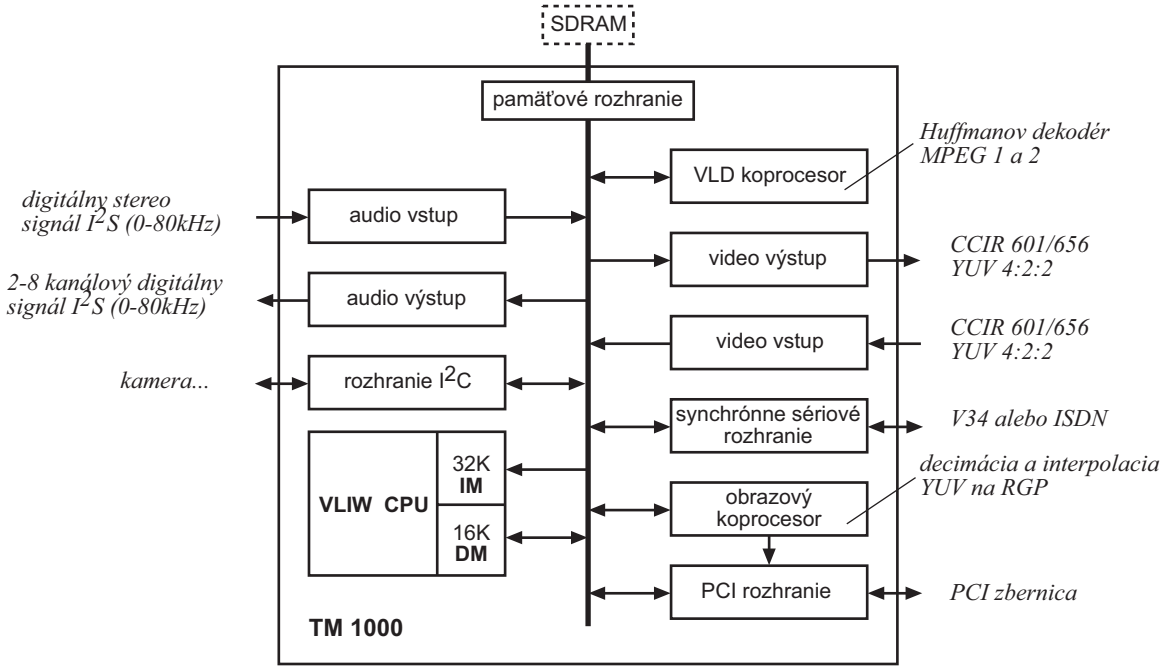
Aj keď výkonnosť doteraz opísaných univerzálnych VLIW DSP je značná, v oblasti spracovania *multimediálnych dát* je často potrebné využiť architektúry s vyšším výpočtovým výkonom dosahujúcim hodnoty niekoľkých GOPS [170]. Z hľadiska cieľových aplikácií je výhodné ak tieto architektúry sú *programovateľné*, čo umožňuje veľmi jednoduchú adaptáciu na nové štandardy, ktorých rýchly vývoj je pre oblasť multimediálnych dát charakteristická. Počas niekoľkých uplynulých rokov sa z oblasti univerzálnych DSP vyčlenila skupina programovateľných procesorov, ktorá sa zvykne nazývať termínom *multimediálne procesory*[26] [85]. Do tejto skupiny je možné zahrnúť aj klasické výkonné mikroprocesory s multimediálnymi rozšíreniami (ich výpočtová výkonnosť je však podstatne nižšia).

Adaptácia programovateľných procesorov pre oblasť spracovania multimediálnych dát je v súčasnosti realizovaná predovšetkým s využitím nasledujúcich princípov [21]:

- ***Vytvorením špecializovaných inštrukcií*** pre často sa opakujúce typy operácií. Typickým príkladom je operácia MAC s prípadnou saturáciou, prípadne ďalšie operácie používané napr. pri dekompresii obrazov.
- ***Využitím špecializovaných modulov*** (koprocesorov alebo akcelerátorov) prispôsobených pre konkrétne aplikácie, ktoré široko využívajú paralelizmus a zreťazenie.

Tieto prístupy sa kombinujú v špeciálnych multimediálnych procesoroch, ktorých prehľad je uvedený napr. v [73], [85]. Typickým príkladom štruktúry madiálneho procesora je bloková schéma multimediálneho procesora Trimedia TM-1000 firmy Philips zobrazená na obr. 4.17 [72]. Tento procesor obsahuje jadro CPU na báze architektúry VLIW, ktoré obsahuje 27 funkčných jednotiek, ktorých význam je uvedený v tab. 4.5 [72]. Počet a štruktúra týchto jednotiek bola optimalizovaná predovšetkým z hľadiska realizácie MPEG dekodéra a vybraných 3D grafických operácií.

[26] Multimediálne procesory (media processors) – sú programovateľné procesory určené pre zrýchlenie súbežného spracovania rôznych typov multimediálnych dát ako sú napr. digitálne video, digitálne audio, počítačové animácie, texty a grafika. Môžu pracovať ako jednočipové procesory, prípadne obsahovať stykové obvody (napr. PCI zbernicu) umožňujúce ich integráciu do nadradených (počítačových) systémov [74].



Obr. 4.17 Architektúra multimediálneho procesora Trimedia TM-1000

Tab. 4.5 Funkčné jednotky VLIW CPU

Funkčná jednotka	Počet
Konštanty	5
ALU (celočíselná)	5
Čítanie/zápis	2
DSP ALU	2
DSP MUL	2
Posúvač	2
Vetvenie	3
Násobička (celočíselná/pohyblivá)	2
ALU (pohyblivá)	2
Porovnanie (pohyblivá)	1
Odmocnina/delenie (pohyblivá)	1

Jadro CPU má 128 32-bitových registrov pre všeobecné použitie a využíva VLIW inštrukcie obsahujúce maximálne 6 inštrukcií, pričom VLIW inštrukcie používajú premenlivú dĺžku a špeciálnu kompresiu. Jednotlivé inštrukcie môžu byť jednoduché RISC inštrukcie ako aj špecializované *multimediálne inštrukcie* optimalizované predovšetkým pre štandardné video kompresné a dekompresné algoritmy, ktorých zložitosť je ekvivalentná až 11-tim RISC inštrukciám. CPU obsahuje aj SIMD rozšírenia pre prácu s 8 a 16-bitovými dátami. Jadro je schopné programovo realizovať MPEG2 audio/video dekodér a MPEG1 kodér.

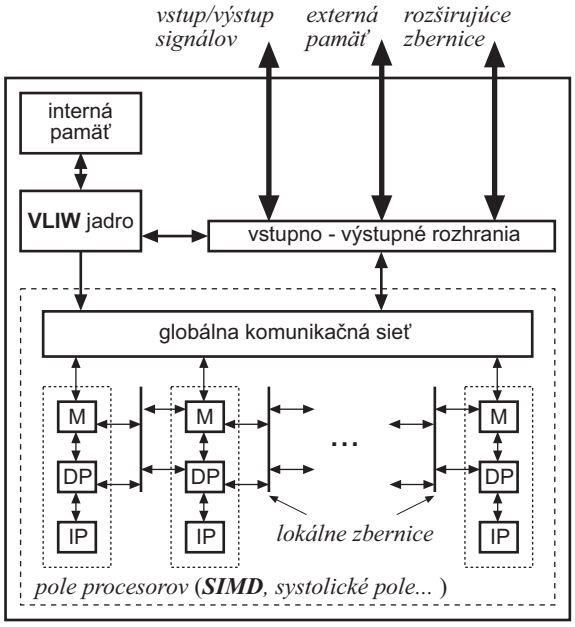
Na čipe sú umiestnené koprocesory pre Huffmanove dekódovanie a prevzorkovanie obrazu (novší čip TM-1100 obsahuje naviac dešifrovací koprocesor pre DVD aplikácie). Z koncepčného pohľadu ide teda o veľkú podobnosť s najnovšími VLIW DSP.

4.3.1 MULTIMEDIÁLNE ROZŠÍRENIA SIMD

Tieto inštrukcie zvyčajne využívajú existujúce dátové cesty procesora (napr. známe MMX rozšírenie firmy Intel je umiestnené v registroch koprocesora pre pohyblivú rádovú čiarku) a pracujú ako vektorové SIMD koprocesory. Sú typické pre univerzálne výkonné mikroprocesory a známe ako *multimediálne rozšírenia*. Do tejto kategórie patrí MMX rozšírenie procesorov Pentium a Pentium II od firmy Intel, vizuálna inštrukčná sada (VIS – Visual Instruction Set) procesorov UltraSPARC od firmy Sun alebo AltiVec procesorov PowerPC od firmy Motorola [85]. Tieto rozšírenia využívajú princíp zbalenej aritmetiky a sú optimalizované pre spracovanie audio a video signálov.

4.3.2 KOPROCESORY S VYSOKÝM STUPŇOM PARALELIZMU

Koncepčne je možné znázorniť všeobecnú štruktúru vysokovýkonného multimediálneho procesora ako spojenie *univerzálneho reprogramovateľného jadra*, ktoré realizuje časti algoritmov s malou regularitou a *vysokovýkonného* (v optimálnom prípade do určitej miery programovateľného) koprocesora, čo je principiálne znázornené na obr. 4.18 [21].



Obr. 4.18 Všeobecná architektúra výkonného multimediálneho procesora

V súčasnosti sa ako univerzálne jadrá používajú architektúry VLIW z rovnakých dôvodov ako v prípade VLIW DSP. Z pohľadu koprocesorov s masívnym paralelizmom sa ako najperspektívnejšie javia [21], [29]:

a) Koprocesory na báze systolických polí

Teória synchrónnych systolických polí (SA – Systolic Arrays) a asynchrónnych systolických polí (WA – Wavefront Arrays) patrí v oblasti teórie ČSS k veľmi dobre prepracovaným oblastiam [86], [87] a pokroky v technológii VLSI umožňujú realizovať štruktúry SA a WA už priamo vo VLSI čipoch. Typické algoritmy ako je číslicová FIR filtrácia a konvolúcia patria do oblasti algoritmov, ktoré je prakticky s minimálnymi nárokmi možné transformovať do SA prípadne WA. Dokonca aj rekurzívne IIR filtre, prípadne aj zložitejšie algoritmy napr. pre adaptívnu lineárnu a nelineárnu číslicovú

filtráciu je možné pomocou vhodných algoritmov transformovať do SA a WA [88], [89]. Pri vhodnej štruktúre jednotlivých stavebných blokov SA alebo WA je možné dosiahnuť čiastočnú reprogramovateľnosť celej štruktúry. Teória systolických polí je veľmi dobre rozpracovaná v dostupnej literatúre a jej nasadenie v typických aplikáciách je v súčasnosti predovšetkým otázkou technologickou.

b) Koprocesory na báze programovateľných hradlových polí

Jedným zo zaujímavých trendov v oblasti programovateľých architektúr pre ČSS je využitie užívateľsky reprogramovateľných hradlových polí známych pod označením FPGA[27] (Field Programmable Gate Arrays) [29]. Tieto súčiastky umožňujú zmeniť štruktúru prepojení logických obvodov priamo na čipe zmenou konfigurácie zapísanej v pamäti SRAM a tým „*vyrobiť*“ číslicový obvod s užívateľom špecifikovanými vlastnosťami. Obvody tohto typu patria do kategórie obvodov ASIC a z hľadiska algoritmov ČSS majú nasledujúce výhody [90]:

- ***Ľahká rekonfigurovateľnosť***. Zmena zapojenia je užívateľsky definovaná a je ju možné zmeniť zmenou konfiguračného súboru. Táto vlastnosť je určitom zmysle ekvivalentná programovateľnosti procesorov.
- ***Vysoká rýchlosť***. Je daná predovšetkým možnosťou využiť paralelizmus architektúry, ktorý rastie pri náraste hustoty integrácie. V súčasnosti je napr. možné realizovať FIR filtre s frekvenciu vzorkovania presahujúcou 100 MHz, čo je ďaleko nad možnosti súčasných DSP.
- ***Univerzálnosť***. FPGA umožňujú realizovať principiálne rôznorodé algoritmy ČSS. Existujú napr. knižnice špecializovaných funkcií pre konkrétne obvody FPGA. Efektívne implementácie často vyžadujú použitie špeciálnych algoritmov a prístupov (napr. distribuovanej aritmetiky).

S využitím technológie ASIC bude možné na čip vložiť kombinácie procesorov a obvodov FPGA pričom FPGA koprocesory budú realizovať kritické časti algoritmov ČSS [91], [92], [93], [94], [95]. V súčasnosti je možné v súčiastkach FPGA využiť už viac ako milión ekvivalentných hradiel, čo umožňuje realizovať na čipoch FPGA kompletné systémy ČSS, čím sa v určitom zmysle vývoj technických prostriedkov dostáva (aj keď na kvalitatívne vyššej úrovni) do štádia umožňujúceho vytvárať špeciálne technické prostriedky optimalizované pre konkrétny algoritmus.

[27] Niektorí výrobcovia používajú aj termín CPLD – Complex Programmable Logic Arrays.

4.4 ZHRNUTIE

Zaujímavý názor na teóriu architektúr počítačov je uvedený v [41]. Profesor Hlavička v úvode skrípt konštatuje:

„Vysokoškolská skripta by měla nabízet především teorii, zatímco příklady by měly sloužit pouze k tomu, aby využití této teorie ilustrovaly. Toto pravidlo lze dodržet ve většině technických disciplin, ne však v architektuře počítačů. Všechny uznávané učebnice architektury počítačů jsou v podstatě konstatováním faktů, tedy popisem toho, co se v oblasti počítačových struktur zatím podařilo vymyslet a hlavně uplatnit v praxi. Rozdíly ve spracování pak spočívají především v tom, jakým způsobem jsou popisované typy počítačů uspořádány nebo roztříděny a s jakou dávku fantazie jsou zpětně zdůvodňovány vypozorované zákonitosti, případně rozdíly. Takový popis si ovšem v žádnem případě nezaslouží označení teorie, takže nezbývá než si otevřeně přiznat, že teorie architektury počítačů neexistuje.“

Tento názor by zrejme našiel rad oponentov, ktorí by s nim nesúhlasili. Oblasť moderných DSP, ktoré boli opísané v tejto kapitole, patrí do oblasti počítačových architektúr, ktoré sa v súčasnosti dynamicky vyvíjajú a podľa autora tejto práce tieto znaky spĺňuje. Cieľom tejto kapitoly bola klasifikácia, konštatovanie najnovších faktov a naznačenie vývoja v kategórii moderných DSP, ktoré v autorovi dostupnej literatúre nebolo komplexnejšie realizované.

Paralelizmus je z pohľadu technických prostriedkov pri využívaní súčasnej CMOS VLSI technológie najčastejšie využívaným prostriedkom na zvýšenie rýchlosti DSP. V tejto kapitole boli opísané rôzne formy využitia paralelizmu, ako v oblasti DSP založených na klasickej harvardskej architektúre, tak aj v najmodernejších VLIW DSP a multimediálnych procesoroch. Vývoj v tejto oblasti zaznamenal v minulých troch – štyroch rokoch veľmi rýchly vývoj a predovšetkým z pohľadu klasických DSP priniesol výraznú zmenu v architektúre najvýkonnejších DSP. Zdá sa, že konvergencia univerzálnych procesorov a DSP bude naďalej pokračovať, k čomu by mohla prispieť aj ohlásená (v apríli 1999) aliancia medzi firmami Analog Devices a Intel, ktorej cieľom je vyvinúť nové výkonné jadro DSP na báze VLIW architektúry.

Prechod DSP k architektúre VLIW je nevyhnutným dôsledkom snahy o zvyšovanie výkonnosti DSP a využitie pokroku v technológii VLSI. Aj keď samotný princíp VLIW je v počítačovej oblasti známy a široko využívaný (z pohľadu počítačových architektúr je to však len jedena z mnohých architektúr a napr. v [41] je jej venovaná len jedna strana), integrácia VLIW DSP do monolitickej formy prináša mnoho nových prístupov a modifikácii. Prvý VLIW DSP – TMS320C62x je ešte do značnej miery podobný univerzálnej VLIW architektúre. Novšie jadrá VLIW DSP (napr. Star Core) využívajú pre VLIW architektúru mnohé atypické rozšírenia, ktoré vyplývajú z optimalizácie pre monolitickú integráciu a predstavujú určitý návrat k základným stavebným blokom klasických DSP – predovšetkým MAC a AAU jednotkám (samozrejme na kvalitatívne vyššej úrovni), čo je dané predovšetkým ich optimalizáciou pre algoritmy ČSS. Tieto trendy naznačujú, že moderné DSP budú využívať novú jedinečnú *hybridnú architektúru*, založenú na výhodných vlastnostiach architektúr DSP a VLIW procesorov.

Porovnanie základných vlastností DSP a jadier DSP v súčasnosti dostupných (vrátane niektorých DSP z prvej a druhej generácie, ktoré sa v niektorých zariadeniach stále používajú) a oznámených VLIW DSP je uvedené v prehľadnej tabuľke v prílohe na str. 110. Tabuľka je uvedená v originálnej forme, so všetkými pôvodnými informáciami a

obsahuje aj hodnoty BDTImark naznačujúcich ich výkon pri implementácií všeobecných algoritmov ČSS.

Z pohľadu užívateľov je popri výraznom náraste výkonnosti dôležité predovšetkým zjednodušenie programovania tejto triedy procesorov a teda snaha o pokrok v oblasti programových prostriedkov. Tejto problematike je venovaná časť nasledujúcej kapitoly.

5 VÝVOJOVÉ PROSTRIEDKY

Vysoký výpočtový výkon DSP je len jedným z nutných predpokladov pre úspešnú realizáciu systémov ČSS. Na dosiahnutie vysokého výkonu v praktických aplikáciách má podstatný (a často dokonca dominantný) vplyv kvalita *vývojových prostriedkov*. Aj keď súčasné DSP sú z hľadiska typov vývojových prostriedkov porovnateľné s univerzálnymi mikroprocesormi, vzhľadom na požiadavku práce v reálnom čase je nízkoúrovňové programovanie v asembleri často jedinou reálnou alternatívou. V rámci tejto kapitoly budú opísané jednotlivé prostriedky na tvorbu programov, ktoré sú v súčasnosti typicky využívané pri vývoji aplikácií na báze DSP, pričom budú naznačené aj východiska a metódy riešenia, ktoré autor práce použil v rámci projektu Copernicus[28] pri vývoji operačného systému DSP OS a optimalizovaných *vektorových knižničných funkcií*, ktoré predstavujú jeden z možných spôsobov efektívneho využitia dostupných vývojových prostriedkov a sú založené na niektorých všeobecných princípoch. Tieto, v podstate *inžinierske riešenia*, majú v hierarchii realizácie systémov ČSS na báze DSP významné postavenie a často rozhodujú (predovšetkým v prípade zložitejších aplikácií) o úspechu celého riešenia.

5.1 ZÁKLADNÉ VÝVOJOVÉ PROSTRIEDKY PRE DSP

Do tejto kategórie patria predovšetkým asemblery, linkovacie programy, knižničné funkcie, simulátory, vývojové dosky a emulátory.

5.1.1 ASEMBLERY

Asembler je program, ktorý prekladá špecifické inštrukcie procesora zapísané v zdrojovom ASCII súbore do binárneho *objektového kódu* (object code) pre cieľový DSP. Zvyčajne tento objektový kód (pokiaľ je v tzv. *relatívnom tvare*) vyžaduje dodatočnú transformáciu (*relokáciu* a *linkovanie*), ktoré sa realizujú linkovacím programom, ktorý produkuje vykonateľný binárny kód pre cieľový DSP. Vo fáze linkovania je možné použiť dostupné knižničné funkcie.

Väčšina súčasných asemblerov sú tzv. *macro* asemblery, ktoré umožňujú definovať (alebo používať preddefinované) *parametrizovateľné bloky kódu* (makrá), ktoré sú do zdrojového kódu vkladané počas prekladu. Makrá umožňujú programátorovi zmenšiť množstvo zdrojového kódu, ktorý je treba udržiavať (čím je možné zvýšiť spoľahlivosť programov) ako aj eliminovať nadbytočné inštrukcie, ktoré je potrebné použiť pri volaní podprogramov. Nasledujúci kód dokumentuje použitie asemblerovského makra pri realizácii FIR filtra pomocou procesora Motorola DSP5600x.

[28] Copernicus grant CIPA-CT94-0220 – Innovative Methods of Noise and Vibration Analysis on the reciprocating Machinery for the Purpose of Quality Control and Diagnostics [109].

```
; definícia makra „FIR", ktoré implementuje FIR filter s N koeficientmi
FIR   macro  N
      clr    a
      rep    #N-1
      mac    x0,y0,a     x:(r0)+,x0    y:(r4)+,y0
      macr   x0,y0,a     (r0)-
      endm

; inicializácia smerníkov na koeficienty a dáta
      move   #data,r0
      move   #koeficienty,r4

; volanie makra pre FIR filter s 512 koeficientmi
      FIR    512
```

Zaujímavou alternatívou ku klasickým asemblerom sú tzv. *algebraické asemblery*, u ktorých je snaha používať štýl písania asemblerovských programov, ktoré pripomínajú algebraický zápis algoritmu. Tento prístup využíva napr. firma Analog Devices, čo dokumentuje nasledujúci kód FIR filtra pre procesor ADSP21xx:

```
.ENTRY     fir;
fir:   MR = 0, MX0 =  DM( I0, M1),    MY0 = PM(I4, M5)
       DO sop UNTIL CE;
sop:          MR = MR + MX0*MY0( SS ), MX0=DM(I0, M1), MY0=PM(I4, M5)
       MR = MR+MX0*MY0(RND);
       IF  MV  SAT  MR;
       RTS
```

Existuje dokonca snaha používať syntax, ktorá pripomína jazyk C, čo je naznačené v prípade kódu pre asembler VLIW jadra Carmel na str. 50.

Pokrok v oblasti paralelných VLIW DSP je sprevádzaný tzv. *optimalizačnými asemblermi*, ktorých úlohou je vhodne zoskupiť sekvenčne zapísané inštrukcie do paralelných VLIW inštrukcií s cieľom optimalizovať výkonnosť VLIW DSP paralelným vykonaním čo najväčšieho počtu inštrukcií v jednotlivých taktoch. Efektívnosť týchto optimalizačných asemblerov je daná okrem pokroku v oblasti moderných prekladačov [96], [97] predovšetkým zvýšenou ortogonalitou architektúr VLIW DSP a možnosťou zoskupovať niekoľko paralelných a relatívne nezávislých inštrukcií. Medzi najčastejšie používané metódy optimalizácie patrí *rozvinutie slučiek* (loop unrolling) a *softvérové zreťazenie* (software pipelining) [79]. Tieto techniky je možné dokumentovať pomocou implementácie skalárneho súčinu vyjadreného C kódom

```
short       *a, *b;
inc         c;
for ( i = 0; i < 100; i++ ) c += a[ i ]*b[ i ];
```

ktorý je možné zlúčením dvoch cyklov optimalizovať pre VelociTI architektúru, ktorá obsahuje dve násobičky a dve sčítačky. Využitie týchto prostriedkov umožňuje nasledujúci kód v asembleri TMS320C62x využívajúci spojenie dvoch za sebou idúcich iterácií

```
LOOP:
      . LDW       .D1   *AA++,  AI                ; načítanie a[ i ] & a[ i+1 ]
      . LDW       .D2   *BA++,  BI                ; načítanie  b[ i ] & b[ i+1 ]
      . MPY       .M1X  AI,  BI,  AP              ; a[ i ] *b[ i ]
      . MPYH      .M2X  AI,  BI,  BP              ; a[ i+1 ]*b[ i+1]
      . ADD       .L1    AP,  ACA,  ACA           ; ca += a[ i ]*b[ i ]
      . ADD       .L2    BP,  BCB,  BCB           ; cb += a[ i+1 ]*b[ i+1 ]
      . [ B0 ] SUB        .S2  B0  1  B0          ; zníženie počítadla cyklov
      . [ B0 ] B          .S1  LOOP               ; skok na slučku
```

Tento kód je optimalizačným asemblerom ďalej optimalizovaný s využitím softvérového zreťazenia do tvaru [79]

```
            B     .S1   LOOP                      ; skok na slučku
            B     .S1   LOOP                      ; skok na slučku
            B     .S1   LOOP                      ; skok na slučku
            B     .S1   LOOP                      ; skok na slučku
      ||    ZERO  .L1   A2                        ; vynulovanie pomocnej premennej
      ||    ZERO  .L2   B2                        ; vynulovanie pomocnej premennej
            B     .S1   LOOP                      ; skok na slučku
      ||    ZERO  .L1   A3                        ; vynulovanie pomocnej premennej
      ||    ZERO  .L2   B3                        ; vynulovanie pomocnej premennej
      ||    ZERO  .D1   A1                        ; vynulovanie pomocnej premennej
      ||    ZERO  .D2   B1                        ; vynulovanie pomocnej premennej
LOOP: LDW   .D1   *A4++,  A1                      ; načítanie a[ i ] & a[ i+1 ]
      ||    LDW   .D2   *B4++,  B1                ; načítanie b[ i ] & b[ i+1 ]
      ||    MPY   .M1X  A1,  B1,   A2             ; a[ i ]*b[ i ]
      ||    MPYH  .M2X  A1,  B1,  B2              ; a[ i+1 ]*b[ i+1 ]
      ||    ADD   .L1    A2,   A3,  A3            ; ca += a[ i ]*b[ i ]
      ||    ADD   .L2    B2,   B3,  B3            ; cb += a[ i+1 ]*b[ i+1]
      || [B0] SUB   .S2    B0,   1,   B0           ; zníženie počítadla cyklov
      || [B0] B     .S1    LOOP                    ; skok na slučku

            ADD   .L1X    A3,  B3,  A3            ; c = ca+cb (po všetkých slučkách)
```

Vložením piatich skokov na slučku pred samotnú slučku je zabezpečené naplnenie fronty inštrukcií a kontinuálne vykonávanie dvoch MAC operácií v jednom cykle aj napriek tomu, že inštrukcia načítania má oneskorenie 5 taktov.

Väčšina asemblerov generuje objektový kód v tvare tzv. COFF (Common Object File Format) súboru, ktorý je v tejto oblasti štandardom.

5.1.2 KNIŽNIČNÉ FUNKCIE

Pre aplikácie na báze DSP majú optimalizované knižnice kľúčový význam. Aj keď typické jadrá algoritmov ČSS implementované na DSP sú relatívne krátke, ich optimalizácia je z pohľadu programátora značne náročná úloha, ktorá vyžaduje dobrú znalosť architektúry cieľového DSP. Naviac je často potrebné zvážiť možnosti modifikovať samotný algoritmus ČSS do tvaru, ktorý je pre danú architektúru vhodnejší, čo zvyčajne vyžaduje špecifické znalosti z oblasti teórie ČSS. Je logické, že výrobcovia DSP poskytujú pre svoje procesory optimalizované kódy, ktoré umožňujú relatívne efektívnu implementáciu základných algoritmov ČSS (FIR, IIR, FFT, DCT…). Tieto kódy sú dostupné v rámci WWW stránok jednotlivých výrobcov, aplikačných príručiek a špecializovaných kníh [98], [99], [100], [101], [102].

Aj keď je často možné tieto kódy ďalej vylepšiť (čo často s odstupom času robia aj samotní výrobcovia[29]), sú tieto kódy dobrým štartovacím bodom pre vývoj aplikačných programov.

Knižničné funkcie je možné pomocou asemblera preložiť do relatívneho objektového tvaru a spojiť pomocou programu – *knihovníka* do jedného súboru – knižnice.

5.1.3 SIMULÁTORY

Simulátory sú programy, ktoré simulujú prácu procesora na úrovni jednotlivých inštrukcií. Z pohľadu optimalizácie kódu pre DSP majú kľúčovú úlohu a umožňujú odhaliť problémy vznikajúce napr. vplyvom zreťazenia, prípadne zvýšiť efektivitu kódu identifikáciou inštrukcií, medzi ktorými sa vytvárajú tzv. „pipeline bubbles", t.j. okamihy, kedy zreťazené technické prostriedky nie sú plne využité. Vhodným preusporiadaním inštrukcií je často možné zvýšiť využitie prostriedkov procesora. Simulátory by mali splňovať predovšetkým nasledujúce vlastnosti (uvedené v poradí ich dôležitosti):

- ***funkčnú presnosť*** – simulátor by mal presne modelovať činnosť procesora,
- ***časovú presnosť*** – simulátor by mal odrážať napr. rôznu dĺžku trvania inštrukcií v závislosti na všetkých faktoroch (napr. pomalších externých pamätiach),
- ***kompletnosť*** – simulátor by mal simulovať aj všetky periférne obvody ako sú časovače, kanály DMA, sériové kanály a pod.,
- ***rýchlosť*** – simulátory sú výrazne pomalšie ako reálne DSP a v praxi realizujú rádovo tisícky inštrukcií za sekundu, pričom pre niektoré zložitejšie algoritmy môže byť doba simulácie veľmi dlhá.
- ***užívateľsky priateľské*** – mali by umožňovať pohodlnú prácu aj pri zložitých simuláciách.

Aj keď tieto požiadavky sú z pohľadu užívateľa samozrejmé, praktické skúsenosti ukazujú, že v simulátoroch, ktoré sa snažia splniť všetky požiadavky (a to nie sú zďaleka všetky simulátory), existujú chyby, ktoré sa snažia výrobcovia priebežne odstraňovať. Dostupnosť najnovších verzií je preto v praxi veľmi dôležitá.

5.1.4 VÝVOJOVÉ DOSKY, EMULÁTORY

Pri vývoji zariadení na báze DSP je cieľom realizovať systém ČSS, ktorý pracuje s reálnymi vstupnými resp. výstupnými dátami. V určitej etape vývoja je dôležité mať k dispozícii reálne technické zariadenie (často sa technické a programové prostriedky vyvíjajú paralelne a cieľové zariadenie nie je k dispozícii). Na overenie činnosti algoritmov ČSS v reálnom čase sú vhodné vývojové dosky. Všetci hlavní výrobcovia poskytujú lacné vývojové moduly (v cenách 70 – 250 $), ktoré je možné dokonca v prípade menej náročných zariadení využiť aj ako konečné technické riešenie. Výhodné je pokiaľ cieľový DSP podporuje emuláciu na čipe, čo zlepšuje[30] možnosti ladenia priamo v reálnych podmienkach pri minimálnych nárokoch na dodatočné technické vybavenie.

---

[29] Čo do určitej miery potvrdzuje názor, že tvorba optimálnych programov je náročná úloha vyžadujúca komplexný pohľad na úlohu.

[30] Nie je však vhodné tieto možnosti preceňovať. Pri spracovaní rýchlych signálov a obmedzenej záznamovej pamäti v rámci emulačnej logiky je často potrebné využiť rôzne nepriame metódy ladenia.

Využitie klasických emulátorov je v praxi veľmi problematické predovšetkým vzhľadom na stále sa zvyšujúcu taktovaciu frekvenciu DSP a používanie prevažne SMD technológie, čo vylučuje nasadenie klasickej emulačnej hlavice.

5.2 VYŠŠIE PROGRAMOVACIE JAZYKY

Programovanie vo *vyšších programovacích jazykoch* (v oblasti DSP zvyčajne C, menej často C++ a ADA) má niekoľko významných výhod [103], [104], [175]:

- ***Spoľahlivosť kódu***. Je všeobecne známe, že je ľahšie vyvinúť bezchybný kód vo vyššom programovacom jazyku ako pomocou asemblera.
- ***Udržiavateľnosť (maintainability) kódu***. Zrozumiteľnosť kódu vo vyššom programovacom jazyku je oveľa vyššia (samozrejme pokiaľ je dobre napísaný), a teda kód je ľahšie udržiavateľný aj pre iného programátora. Naopak o kóde v asembleri je všeobecne známe, že aj autor programu s odstupom času má problém kód upravovať.
- ***Prenositeľnosť kódu***. Vyšší programovací jazyk je relatívne nezávislý (závisí na konkrétnej implementácii) na cieľovom procesore, na rozdiel od asemblera, ktorý je závislý na type procesora a teda ťažšie prenositeľný.
- ***Produktivita práce***. Vyššie programovacie jazyky umožňujú pracovať na vyššej úrovni abstrakcie, čo umožňuje priame zvýšenie produktivity práce. Hľadanie chýb v asembleri zaberá počas ladenia viac času, čo v konečnom dôsledku tiež znižuje efektivitu práce.

Základnou nevýhodou vyšších programovacích jazykov z pohľadu DSP je *nižšia rýchlosť* výsledného kódu a *zväčšenie dĺžky kódu*. Aj keď v niektorých menej kritických aplikáciách je možné vyšší programovací jazyk použiť, pre časovo kritické aplikácie je nutné použiť klasické postupy kombinovania vyššieho programovacieho jazyka a nízkoúrovňového programovania v asembleri.

5.2.1 JAZYK C PRE KLASICKÉ A ROZŠÍRENÉ DSP

Jazyk C má v oblasti DSP dominantné postavenie. Je to spôsobené jednak všeobecnou znalosťou jazyka medzi programátormi, jeho „tesnou väzbou“ s technickými prostriedkami (napr. bitové operácie) a jeho relatívnou jednoduchosťou. Ďalším faktorom jeho popularity je dostupnosť kvalitného voľne dostupného C prekladača (GNU C) od združenia Free Software Foundation, ktorý je možné adaptovať pre rôzne typy procesorov (vrátane DSP), čo niektorí výrobcovia aj využívajú (napr. C prekladače od firmy Motorola pre rady DSP56xxx a DSP9600x a Analog Devices pre rad ADSP21xx sú založené na tejto technológii). Z pohľadu predovšetkým DSP s pevnou rádovou čiarkou však jazyk C má niekoľko podstatných nevýhod predovšetkým v týchto oblastiach:

a) Podpora pre zlomkový formát

Jazyk C nemá podporu pre tento prirodzený dátový typ DSP s pevnou rádovou čiarkou, čo núti prekladač generovať kód používajúci celočíselné operácie. Tieto operácie využívajú dátové cesty DSP neefektívnym spôsobom a v podstate eliminujú efektívne využívanie najvýkonnejšej MAC inštrukcie.

b) Podpora pre duálne pamäte

Jazyk C nerozlišuje rôzne pamäťové priestory, čo je naopak pre DSP vychádzajúce z modifikácií harvardskej architektúry kľúčová vlastnosť. Pomocou štandardného jazyka C tak nie je možné priamo pristúpiť k jednej z (dátových) pamätí.

c) Podpora pre špeciálne spôsoby adresovania

Špeciálne spôsoby adresovania podporujú najčastejšie využívané dátové štruktúry (napr. oneskorovacie linky číslicových filtrov, preusporiadaný výstup FFT algoritmu a pod.) priamo pomocou technických prostriedkov DSP, typicky umiestnených v AAU. Keďže jazyk C nemá priamu podporu pre tieto dátové typy, je kód generovaný univerzálnym prekladačom jazyka C značne neefektívny.

Ďalšou nevýhodou pre jazyk C je veľká neortogonalita inštrukčnej sady procesorov s pevnou rádovou čiarkou (DSP s pohyblivou rádovou čiarkou zvyčajne obsahujú registrový súbor univerzálnych registrov, ktorý je z pohľadu prekladača podstatne vhodnejší) čo je z pohľadu optimalizačných techník používaných v prekladačoch zásadný problém. Prakticky všetky implementácie prekladačov C generujú výstupný súbor v asemblerovskom formáte, aby bolo možné realizovať ručnú optimalizáciu (v optimálnom prípade len) časovo kritických segmentov kódu.

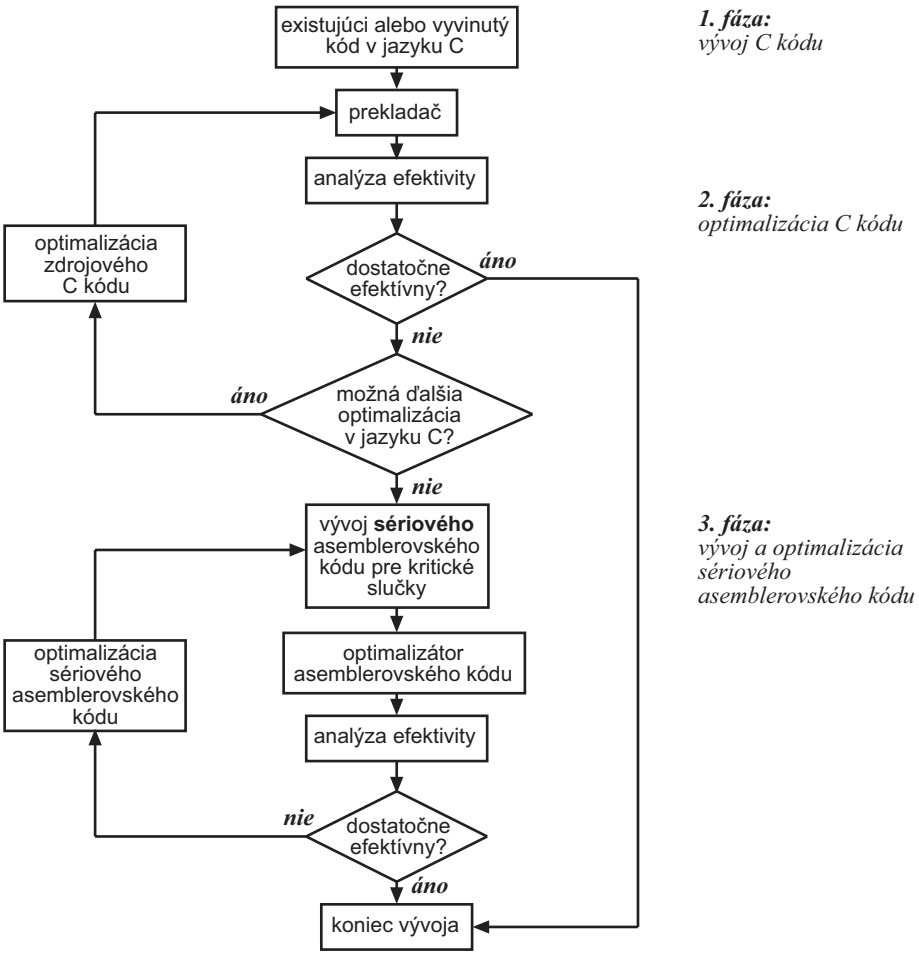
V praxi však existuje obrovský tlak na zvýšenie produktivity práce pri programovaní DSP, čo mnohí výrobcovia prekladačov pre DSP podporujú zavádzaním vlastných rozšírení jazyka C a to predovšetkým podporou nových dátových typov (fixpnt, long fixpnt, accum...), podporou modulo adresovania (circ smerníky), podporou duálnych pamätí (xdata, ydata) a možností vkladania segmentov asemblerovského kódu priamo do zdrojového kódu jazyka C (napr. direktívy __asm). Medzi najprepracovanejšie rozšírenia patrí tzv. numerické rozšírenie jazyka DSP/C, ktoré okrem uvedených rozšírení využíva aj rozšírenia pre vektorové spracovanie a je stručne opísané v podkapitole 5.2.3.

5.2.2 JAZYK C PRE VLIW DSP

Jedným z cieľov uvedenia architektúr VLIW DSP (samozrejme popri výraznom zvýšení výpočtovej výkonnosti) bolo poskytnúť architektúru s vysokým stupňom ortogonality. Zdá sa, že pokrok v oblasti prekladačov C pre VLIW architektúry umožňuje do značnej miery minimalizovať potrebu ručnej optimalizácie. Významnú úlohu v generovaní výstupného kódu majú aj optimalizačné asemblery, ktoré paralelizujú pôvodne sériový vstupný asemblerovský kód. Typický proces tvorby programov pre procesory VLIW je znázornený na obr. 5.1 [79].

Problematika efektivity programovania v jazyku C pre VLIW DSP je zatiaľ veľmi nová. Prakticky dostupné sú výsledky predovšetkým pre architektúru VelociTI, ktorá je na trhu univerzálnych VLIW DSP najdlhšie a je dokumentovaná pre niektoré typické algoritmy ČSS v tab. 5.1 [79]. Je však potrebné zdôrazniť, že zdrojový kód v jazyku C musí byť optimalizovaný pre použitý prekladač, čo je napr. pre implementáciu IIR filtrov pomocou architektúry VelociTI dokumentované v [79].

Bude zaujímavé sledovať, aká bude efektivita prekladačov pre ďalšie VLIW DSP, pričom predovšetkým pre VLIW DSP s rozsiahlym využívaním SIMD rozšírení (zvlášť v architektúre TigerSharc) bude zrejme potrebné rozsiahle využívanie optimalizovaných vektorových funkcií, ktoré môžu byť efektívne implementované napr. v rámci prekladača pre jazyk DSP/C.



Obr. 5.1 Programovanie VLIW DSP v jazyku C

Tab. 5.1 Efektivita programovania VelociTI architektúry v jazyku C

Testovací algoritmus	Optimalizovaný asembler (cyk)	Prekladač C (cyk)	Relatívna účinnosť (%)
CRC	45	90	50
DCT, inverzný MPEG	247	282	88
FIR filter	715	1118	64
FIR LMS filter	69	77	90
Harmonický oscilátor	106	140	76
IIR filter	58	69	84
Krížový analyzačný filter	27	42	64
Krížový syntetizačný filter	40	52	77
Maximálna hodnota	30	58	52
Viterbiho GSM ekvalizér	5182	8535	61
Viterbiho V32 dekodér	79	113	70

5.2.3 JAZYK DSP/C (NUMERICAL C)

V rámci snáh o rozšírenie jazyka C bola vytvorená skupina NCEG (Numeric C Extensions Group), ktorej cieľom je doporučenie rozšírení jazyka o výraznejšiu podporu numerických operácií v rámci rozšíreného jazyka DSP/C [4]. V tejto skupine je z výrobcov DSP aktívny

predovšetkým Analog Devices, čo sa prejavuje aj v jeho podpore v rámci dodávaného prekladača pre procesory radu Sharc [105]. Vzhľadom na podporu vektorového spracovania na báze hierarchickej SIMD architektúry v najnovšom produkte TigerSharc je využitie DSP/C veľmi aktuálne.

DSP/C je nadmnožinou jazyka C a ponúka rozšírenia predovšetkým v oblasti práce s vektormi, zavedením nových vektorových operácií, nových komplexných a zlomkových typov, podpory pre modulo smerníky a oneskorovacie linky a polí s premenlivou dĺžkou. Ilustratívnym príkladom je realizácia FIR filtra v tomto jazyku:

```
fixpnt *circ    sm;
fixpnt          oneskorovacia_linka[ N ], koef[ N ];
volatile fixpnt vstup, vystup;                          /* mapované AD,DA prevodníky */
…
sm = linka;
for( ;;)
{
        *--sm =  vstup;                                 /* načítanie vstupnej vzorky */
        vystup = sum( oneskorovacia_linka[ ; ] * koef[ ; ] ); /* výpočet FIR filtra */
}
```

Samozrejme rozšírenia jazyka sú omnoho bohatšie a podrobnosti je možné nájsť napr. v [4], [105]. Nevýhodou však je, že implementácie jazyka pre iné procesory nie sú (zatiaľ) dostupné.

5.2.4 VEKTOROVÉ KNIŽNIČNÉ FUNKCIE

Jednou z možností využiť programovanie DSP vo vyššom programovacom jazyku je vytváranie optimalizovaných vektorových knižničných funkcií, ktoré je možné využívať priamo z vyššieho programovacieho jazyka. Tento prístup je principiálne možné využiť pre klasické DSP ako aj pre VLIW DSP. Optimalizované funkcie by mali mať nasledujúce vlastnosti [137]:

- ***Vektorové spracovanie***. Vektorové funkcie spracovávajú naraz blok *N* údajov (vektor), čo umožňuje znížiť réžiu prekladača pri volaní funkcie na akceptovateľnú hodnotu. Réžia prekladača *R* (tvorená inštrukciami potrebnými na uloženie hodnôt pracovných registrov pred funkčným volaním, prenos parametrov funkcie pomocou registrov resp. softvérového zásobníka a obnovením hodnôt registrov po ukončení funkcie) je tak rozpočítaná na celý blok a hodnota na jednu vzorku *R/N* je pre bloky veľkosti rádovo niekoľko stoviek údajov prakticky zanedbateľná.
- ***Optimalizácia***. Funkcie je potrebné optimalizovať predovšetkým z hľadiska rýchlosti. Funkcie, prípadne ich časovo kritické segmenty kódu musia plne využívať zdroje DSP ako sú interné pamäte, MAC a AAU jednotky. V prípade optimalizácie funkcií v asembleri je možné tieto požiadavky ľahko splniť a tak C funkcie optimalizované v asembleri sú v praxi typickým riešením. Ich efektivita pri dostatočne veľkých blokoch (a samozrejme dobrej optimalizácii) je prakticky zhodná s rýchlosťou plne asemblerovských programov.
- ***Udržiavateľnosť a rozšíriteľnosť***. Knižnice by mali byť písané tak, aby umožňovali ich modifikáciu s vynaložením minimálneho úsilia. Je vhodné, aby samotné implementácie jadier funkcií boli nezávislé na mechanizme spolupráce s prekladačom (spôsobe odovzdávania parametrov, alokácii pamäte a pod.). Rozširovanie knižníc by tak malo byť umožnené aj programátorom, ktorý majú minimálne znalosti o interných

mechanizmoch prekladača (dobrá znalosť architektúry cieľového DSP je však nevyhnutná, týka sa však len programátorov, ktorí potrebujú knižnice modifikovať).

Využívanie optimalizovaných vektorových funkcií je v oblasti programovania DSP typické. Medzi jeho hlavné výhody patrí možnosť písania výsledného programu v jazyku C dokonca aj v prípade, že prekladač nepodporuje žiadne rozšírenia pre DSP. Príklad optimalizovanej C funkcie na kopírovanie medzi dátovými pamäťami X a Y harvardskej architektúry DSP5600x [112]

```
void memcpy_x_2_y( int pocet, int* x_zdroj, int* y_ciel);
```

je v prípade prekladača na báze GNU C možné zapísať v tvare:

```
PROLOGUE memcpy_x_2_y(R0_+R1_+R4_),LEAF
SETUP_PARAM_PTR r0                      ; ukazovateľ na prvý parameter (počet)
PARAM_24 r4,r0                          ; pocet do registra r0
PARAM_24 r4,r1                          ; zdrojová adresa do registra r1
PARAM_24 r4,r4                          ; cieľová adresa do registra r4
move x:(r1)+,a                          ; predvýber
START_OF_OVERLAY_CODE                   ; presun jadra do internej pamäte
do    r0,_kopirovanie                   ; spracovanie vektora
      move    x:(r1)+,a    a,y:(r4)+    ; optimalizovaný presun
_kopirovanie
END_OF_OVERLAY_CODE                     ; ukončenie kódu v internej pamäti
EPILOGUE (R0_+R1_+R4_),LEAF
```

pričom asemblerovské makrá (zapísané veľkými písmenami) zabezpečujú všetky činnosti potrebné pre správnu spoluprácu s príslušným prekladačom a jasne izolujú samotné jadro funkcie (z dôvodu názornosti veľmi jednoduchej).

Makrá (START_OF_OVERLAY_CODE a END_OF_OVERLAY_CODE) zabezpečujú, pre programátora úplne transparentne, využívanie malej internej programovej pamäte DSP5600x ako pevnej vyrovnávacej pamäte s využitím techniky dynamických prekryvných programových modulov dokonca aj v prípade veľkých programov.

Tvorba efektívnych a spoľahlivých vektorových knižníc je časovo náročná úloha. Veľmi často špecializované knižnice pre ČSS nie sú súčasťou prekladačov (GNU prekladače pre DSP56xxx [106], [107]) alebo sú extrémne nespoľahlivé (GNU prekladač pre ADSP21xx [108]). V rámci medzinárodného projektu Copernicus [109] autor práce vytvoril a koordinoval vývoj základných vektorových knižníc pre špecializované technické prostriedky MINISYS a ANOVIS založené na signálových procesoroch Motorola DSP56001 a DSP56002, ktoré tvorili základné technické prostriedky v rámci uvedeného projektu. V dobe vzniku projektu (rok 1995) bol jediným dostupným prekladačom pre platformu DSP5600x GNU C prekladač od firmy Motorola a uvedené knižnice boli vytvorené práve pre tento prekladač.

V súčasnosti sú dostupné knižnice:

- ***libvmath*** – obsahuje základné vektorové operácie na presun dát, prácu s kruhovými vyrovnávacími pamäťami, vektorové operácie v základnej a dovojnásobnej presnosti, dekadické logaritmy pre prepočet na dB [124].
- ***libfft*** – obsahuje rôzne verzie algoritmov pre výpočet FFT optimalizovaných z hľadiska rýchlosti, veľkosti kódu a presnosti výpočtu [138], [146], [176]. Súčasťou knižnice sú aj podporné funkcie na generovanie oknových funkcií, funkcie pre výpočet DFT s dĺžkou, ktorá nie je mocninou dvoch [149].

- ***libfilt*** – obsahuje základné implementácie FIR filtrov, rôznych foriem IIR filtrov a polyfázových FIR filtrov [177].
- ***libfsk*** – obsahuje číslicové oscilátory, CORDIC algoritmus, základné bloky FSK a ASK diskriminátora [110], [111].

Tieto knižnice boli vytvárane v rámci projektu Copernicus, v rámci diplomových prác ako aj v rámci kontraktov hospodárskej spolupráce a v súčasnosti sú základným stavebným blokom algoritmov ČSS implementovaných v rámci zariadenia ANOVIS (dodávaných firmou MEDAV GmbH, koordinátorom projektu Copernicus) a nasadených v štandardnej prevádzke u niektorých významných výrobcov prevodoviek v krajinách EU. Tieto knižnice okrem množstva rutinnej programátorskej práce (knižnice majú niekoľko desiatok tisíc riadkov zdrojového kódu) obsahujú aj niektoré nové aplikačne orientované algoritmy, časť ktorých je uvedená v rámci kapitoly 6.

Tvorba optimalizovaných a stabilných knižníc vyžaduje značné ľudské zdroje a je preto výhodné, pokiaľ je možné s minimálnym programátorským úsilím knižnice využiť aj pre novšiu generáciu procesorov. Vytvorené knižnice sú príkladom *vhodne zvolenej koncepcie* tvorby optimalizovanej knižnice.

Samozrejme nutnou podmienkou je zabezpečenie určitej kompatibility už na úrovni technických prostriedkov, čo je napr. u novšieho radu DSP563xx od firmy Motorola zabezpečené kompatibilitou inštrukčnej sady (inštrukcie DSP563xx tvoria nadmnožinu inštrukcií DSP5600x) a kompatibilným mechanizmom riadenia zreťazenia (mechanizmus blokovania v DSP563xx zabezpečuje kompatibilitu s časovo stacionárnym riadením u DSP5600x). Aj keď GNU C prekladač pre DSP563xx využíva z hľadiska rýchlosti a veľkosti kódu efektívnejší mechanizmus spolupráce hlavného programu s podprogramami, z pohľadu opísaných knižníc je možné uvedené zmeny realizovať zmenou príslušných asemblerovských makier. Naviac vylepšené možnosti práce vyrovnávacej programovej pamäte u radu DSP563xx je možné taktiež využiť prepísaním makier START_OF_OVERLAY_CODE a END_OF_OVERLAY_CODE.

Koncepcia uvedených knižníc [112] bola vytvorená skôr, ako bol uvedený prvý procesor radu DSP5630x (koniec roku 1995) a príslušný C prekladač [107] a aj keď uvedený prístup je z pohľadu programovacích techník štandardným postupom, je v oblasti DSP knižníc často nedoceňovaný, pričom jeho uplatnenie môže priniesť značné úspory.

5.3 GENERÁTORY KÓDU

Generátory kódu patria medzi programovacie nástroje, ktoré umožňujú znížiť náročnosť programovania použitím programov schopných generovať segmenty kódu pre cieľový DSP. Generátory kódu sú v praxi typické predovšetkým pre návrh a implementáciu číslicových filtrov [113], [114], [115]. Tieto generátory kódu typicky umožňujú realizovať návrh rôznych číslicových filtrov (FIR, IIR, polyfázových, zrkadlových (mirror) filtrov a pod.). Ich užitočnosť však z pohľadu praktického využitia spočíva predovšetkým v možnosti generovať koeficienty číslicových filtrov po zadaní základnej špecifikácie číslicových filtrov. Generovaný výstupný asemblerovský kód je principiálne možné rýchlo vygenerovať aj ručne pomocou samostatných optimalizovaných knižníc, takže generátor kódu môže predovšetkým ušetriť určitý čas v počiatočnej fáze vývoja, v prípade profesionálnej práce je ich prínos veľmi malý.

Zložitejšie návrhové prostredia často vychádzajú z možnosti generovať výsledný kód z grafickej reprezentácie algoritmu (napr. SIMULINK spolu s príslušnými toolboxami MATLAB-u [116], návrhové prostredie GABRIEL [117] a jeho nasledovník PTOLEMY

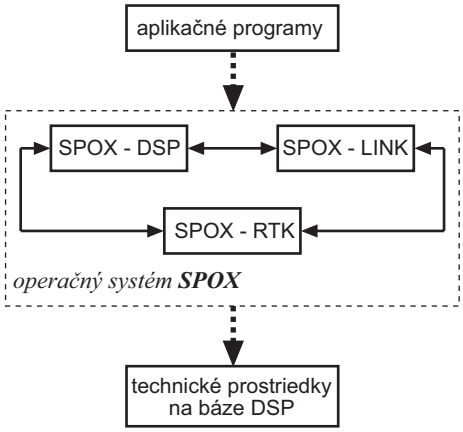
[119]) a ako štandard používajú výstupný generovaný kód v jazyku C, menej často aj v asembleri určitého cieľového DSP (napr. DSP5600x v systéme PTOLEMY). Aj keď je využitie metód grafického programovania z programátorského hľadiska veľmi lákavé, je v súčasnosti efektivita výstupného kódu pre potreby aplikácii na báze DSP stále nízka a môže byť nasadzovaná len v prípade veľmi jednoduchých aplikácií. Z pohľadu praktických aplikácií je v praxi podstatne výkonnejšie využívanie operačných systémov reálneho času optimalizovaných pre aplikácie DSP a kombinované spolu s výkonnými vektorovými knižnicami.

5.4 OPERAČNÉ SYSTÉMY REÁLNEHO ČASU

Cieľom týchto operačných systémov je poskytnúť vysokoúrovňové programátorské prostredie (typicky na úrovni jazyka C) pre tvorbu aplikačných programov pri zachovaní odozvy systému na udalosti v reálnom čase. Medzi najznámejšie systémy reálneho času patrí systém SPOX [4] distribuovaný firmou Loughborough Sound Images [118].

5.4.1 OPERAČNÝ SYSTÉM SPOX

SPOX je integrovaný systém troch hlavných programových komponentov pôvodne určených[31] pre (v čase uvedenia najvýkonnejšie) DSP s pohyblivou rádovou čiarkou TMS320C3x a DSP96002 [118]. SPOX reprezentuje vysokoúrovňové rozhranie k technickým prostriedkom na báze DSP, umožňujúce zvýšiť produktivitu programátorskej práce a zvýšiť prenositeľnosť medzi platformami na báze rôznych DSP. Štruktúra komponentov SPOX je znázornená na obr. 5.2 [118].



Obr. 5.2 Zložky operačného systému SPOX

Každý z komponentov je z pohľadu programátora súbor C funkcií, ktoré pokrývajú rôzne úlohy vznikajúce v systémoch ČSS na báze DSP:

- ***SPOX-RTK*** – umožňuje súbežne vykonávaným programom synchronizovať ich činnosť a poskytovať v reálnom čase odozvu na externé udalosti.
- ***SPOX-LINK*** – umožňuje programom vykonávaným na cieľovom DSP transparentne pristupovať k zdrojom (napr. pevným diskom, pamätiam a pod.) nadradeného systému.

[31] SPOX sa stále rozvíja a v súčasnosti existuje dokonca aj pre PC platformu.

- ***SPOX-DSP*** – umožňuje spracovanie vektorov dát, čítaných a zapisovaných cez tzv. logické I/O kanály, pomocou vektorových, maticových a filtračných funkcií.

Z pohľadu aplikačného programátora je systém SPOX využívaný pomocou aplikačného programového rozhrania (API – Application Programming Interface), ktoré je tvorené funkciami v SPOX-DSP a SPOX-LINK. Toto rozhranie umožňuje izolovať programátora od technických prostriedkov cieľového DSP. SPOX API tak tvorí „virtuálny DSP".

Operačný systém SPOX je prepracovaný a komplexný systém a predstavuje v oblasti DSP systémov (predovšetkým na báze DSP s pohyblivou rádovou čiarkou) špičkové riešenie, ktoré je podporované širokou ponukou technických prostriedkov na báze DSP od všetkých popredných svetových výrobcov. Nevýhodou je značná cena (tisícky libier) samotného systému ako aj kompatibilných DSP produktov a nedostupnosť pre typické DSP s pevnou rádovou čiarkou.

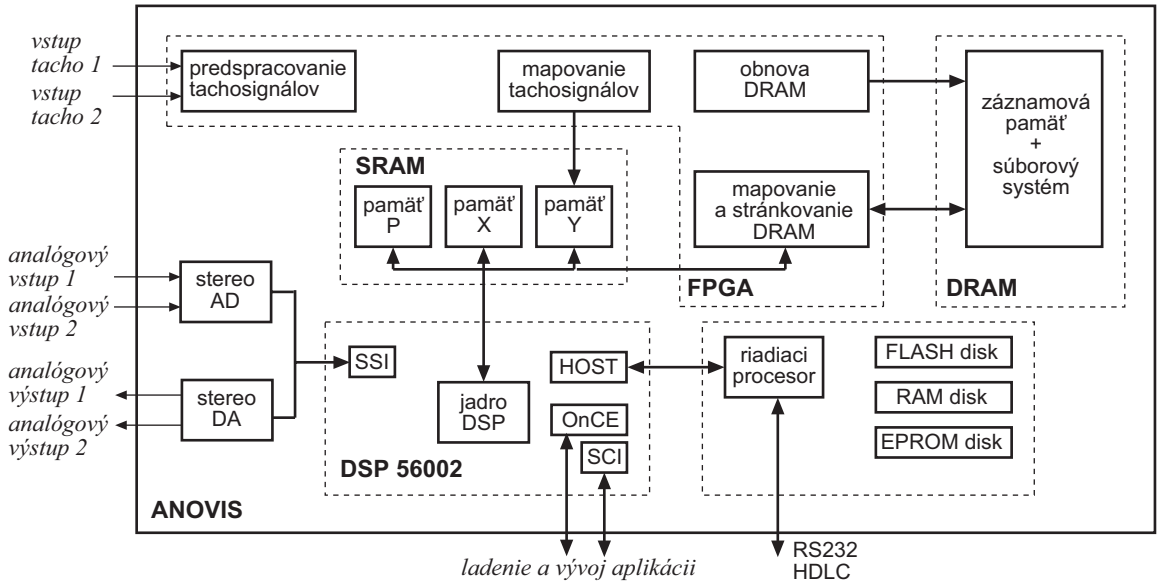
5.4.2 OPERAČNÝ SYSTÉM DSP OS

Tento systém bol autorom práce navrhnutý a implementovaný v rámci už spomenutého projektu Copernicus pre systém MINISYS a neskôr rozšírený v rámci hospodárskej spolupráce aj pre systém ANOVIS [120], [121], ktoré boli postavené na báze procesorov Motorola DSP5600x. Pri tvorbe DSP OS boli sledované nasledujúce ciele [122]:

- umožniť programovanie algoritmov ČSS pre uvedené systémy aj programátorom s minimálnou znalosťou programovania cieľových DSP (Motorola DSP5600x) s využitím programovacieho jazyka C,
- poskytnúť systém dosahujúci rýchlosť spracovania typických algoritmov ČSS, ktorá je porovnateľná s rýchlosťou ručne optimalizovaného kódu,
- umožniť prenos DSP OS aj na iné technické prostriedky na báze procesorov Motorola DSP5600x.

Z pohľadu DSP OS je štruktúra systému ANOVIS zobrazená na obr. 5.3 . Systém obsahuje DSP56002/66 MHz a 128 Kslov rýchlych SRAM pamätí, pričom 64 Kslov je mapovaných do dátovej Y pamäte a 64 Kslov je spoločných pre dátovú pamäť X a programovú pamäť P duálnej harvardskej architektúry DSP56002. Stereo A/D a D/A prevodníky na báze sigma-delta modulácie tvoria štandardný analógový vstup a výstup zariadenia a sú pripojené cez sériové synchrónne rozhranie. Obvody špeciálnych tachosignálov[32] spolu so *záznamovou pamäťou* (transient memory) DRAM veľkosti 16 Mslov sú mapované pomocou hradlového poľa FPGA XILINX do pamäťového priestoru 512 slov v Y pamäti DSP56002. FPGA zabezpečuje obnovovanie DRAM, stránkovanie pri prístupe z DSP56002 a predspracovanie tachosignálov. Komunikácia s nadradeným riadiacim systémom je realizované pomocou 8-bitového HOST rozhrania DSP56002. Nadradený systém obsahuje RAM, EPROM a FLASH disky a prepojenie na systémové rozhrania (RS232 a HDLC) pre spoluprácu s nadradenými systémami. Rozhrania SCI a OnCE DSP56002 [180] sú v systéme využité na ladenie aplikácií.

[32] Tieto signály sú generované pomocou tzv. tachosenzorov a patria medzi základné typy signálov, ktoré sa používajú pri analýze a diagnostike systémov pracujúcich na rotačnom princípe [178].



Obr. 5.3 Štruktúra systému ANOVIS

DSPOS sa skladá z nasledujúcich častí [123] [124]:

a) Alokácia pamäte

Rozloženie pamäte DSP vychádza z dostupných zdrojov DSP56002 a spôsobu prideľovania pamäte v rámci GNU C prekladača [106], pričom umožňuje využívanie všetkých interných dátových pamätí DSP56002 (2×256 slov) pre optimalizované vektorové funkcie. Alokácia DRAM pamäte je realizovaná vo forme jednoduchého súborového systému, ktorý umožňuje alokovať ako lineárne, tak aj kruhové súbory.

b) Zavádzanie systému

DSP OS využíva univerzálny zavádzač, ktorý na báze zavádzacieho režimu (bootstrap mode) procesora DSP56002 identifikuje aktuálne parametre technických prostriedkov (typ procesora, veľkosť inštalovaných pamätí a pod.), vykoná kontrolu inštalovaných pamätí, zabezpečí dekompresiu (kódy BIOS-u a aplikačných programov sú v nadradenom systéme uložené v komprimovanom tvare) a inicializáciu programovej a dátových pamätí. Po týchto činnostiach zrealizuje CRC kontrolu obsahu pamätí a vykoná spustenie systému BIOS.

c) Systém BIOS

Je jadrom systému DSP OS a zabezpečuje základné nízkoúrovňové činnosti systému:

- prístup k A/D a D/A prevodníkom,
- prístup k súborovému systému v DRAM pamätiach,
- prístup k tachosignálom,
- prístup k diskovým zdrojom nadradeného systému,
- rezidentnú činnosť jadra aj pri zmene programu DSP,
- dekompersiu a zavedenie programov z nadradeného systému,
- prístup k sériovej linke nadradeného systému,
- prístup nadradeného systému k súborovému systému v DRAM pamätiach,
- alokáciu a uvoľnenie pamätí SRAM pripojených k DSP.

d) Knižničné funkcie

Tieto knižničné funkcie poskytujú aplikačným programom dodatočné funkcie, ktoré nie sú dostupné pomocou systému BIOS. Do tejto kategórie je možné zahrnúť aj

optimalizované vektorové funkcie popísané v kapitole 5.2.4, pričom ich jediný rozdiel je v úplnej nezávislosti vektorových funkcií na operačnom systéme.

Aplikačný programátor využíva DSP OS ako súbor linkovateľných C funkcií, ktoré umožňujú vytvárať kompletné algoritmy ČSS vo vyššom programovacom jazyku C bez detailných znalostí programovania DSP. Optimalizované vektorové knižničné funkcie je naopak možné vytvárať s minimálnou znalosťou mechanizmov použitého prekladača, čo zvyšuje ich udržiavateľnosť, spoľahlivosť a prenositeľnosť.

DSP OS je systém podstatne jednoduchší ako systém SPOX, umožňuje však plné využitie vyššieho programovacieho jazyka na programovanie aplikácii v jazyku C v rámci systému ANOVIS a všetky súčasné priemyselné aplikácie systému ANOVIS využívajú DSP OS.

Systém je s minimálnymi úpravami prenositeľný aj na iné technické prostriedky na báze procesorov DSP56002. Napr. v rámci práce [125] bol systém prenesený na univerzálnu vývojovú dosku Motorola EVM56002 a okrem zmeny niektorých systémových konštánt DSP OS vyžadoval iba zmenu spolupráce s použitým A/D a D/A kodekom [126].

V rámci ďalšieho rozvoja systému sa v súčasnosti pracuje na prenose DSP OS na platformu založenú na procesoroch radu DSP563xx, ktoré by mali poskytnúť výrazné zvýšenie výkonnosti. Medzi aktuálne úlohy patrí predovšetkým začlenenie koprocesorov a 6-kanálového radiča DMA do filozofie celého systému

Spoločnou vlastnosťou systémov SPOX a DSP OS je využívanie *vektorového prístupu* k spracovaniu údajov a využitie jazyka C, čo sa prejavuje vo zvýšených nárokoch na pamäť (programovú aj dátovú). Najnovšie typy DSP už poskytujú na čipoch SRAM pamäte, ktorých veľkosť presahuje 100 Kslov a tak napr. DSP OS môže pracovať aj v jednočipových zariadeniach.

5.5 ZHRNUTIE

Typy programových prostriedkov pre DSP sú v podstate totožné s prostriedkami pre štandardné mikroprocesory. Väčší dôraz na programovanie v asembleri je daný predovšetkým cieľovými aplikáciami DSP, ktoré vyžadujú maximálny výpočtový výkon. Trendy v oblasti programovania sa však aj v tejto oblasti posúvajú stále viac do oblasti vyšších programovacích jazykov. V prípade zariadení, ktoré sú vyrábané v *nízkych sériach*, je programovanie s využitím postupov využívajúcich operačné systémy optimalizované pre DSP aplikácie v súčasnosti optimálnym riešením. V prípade veľkosériových produktov však ešte stále platí, že programovanie v asembleri je jedným z kľúčových prostriedkov na splnenie požadovaných parametrov (cena, rýchlosť). Ukazuje sa, že programovanie v asembleri (či už kompletných aplikácií, alebo len knižničných funkcií) bude v tejto oblasti ešte dlho používaným prostriedkom, bude sa však zrejme týkať stále menšieho okruhu programátorov. Zvlášť v prípade architektúr VLIW DSP sa očakáva zvýšené využívanie vyšších programovacích jazykov.

Poslednou oblasťou, ktorá ma výrazný vplyv na úspešnosť realizácie systémov ČSS na báze DSP je oblasť samotných algoritmov ČSS. Tejto problematike je venovaná nasledujúca kapitola.

6 VYBRANÉ ALGORITMY ČSS A ICH IMPLEMENTÁCIA NA DSP

Medzi teóriou algoritmov ČSS a architektúrami VLSI obvodov existuje veľmi úzka súvislosť. V špecifickom prípade programovateľných DSP sa teória algoritmov ČSS prejavuje predovšetkým snahou optimalizovať algoritmy ČSS pre konkrétne architektúry komerčných DSP s cieľom dosiahnuť maximálnu výpočtovú výkonnosť pri obmedzeniach konkrétnej architektúry DSP. Typickými obmedzujúcimi faktormi sú napr. limitovaná veľkosť interných pamätí, malá presnosť vplyvom obmedzenej dĺžky slova DSP, alebo špeciálna štruktúra dátových ciest (napr. niekoľko paralelných MAC jednotiek, prítomnosť koprocesorov a pod.). Typickými príkladmi je optimalizácia algoritmu FFT pre rozmery presahujúce rozmery interných dátových pamätí DSP, prípadne realizácia bikvadu – základného bloku IIR filtra.

V praktických aplikáciách je často dokonca výhodné implementovať z pohľadu architektúry DSP veľmi nevýhodný čiastkový algoritmus (napr. výpočet DFT pre blok, ktorého dĺžka nie je mocninou dvoch), pokiaľ je možné v rámci cieľovej aplikácie získať významné výhody. Tento fakt naznačuje, že *teória algoritmov ČSS* je v praxi často nutným predpokladom úspešnej aplikácie DSP v systémoch ČSS.

V rámci tejto kapitoly sú uvedené vybrané algoritmy ČSS, ktoré autor práce implementoval predovšetkým na procesoroch Motorola DSP5600x a DSP5630x v rámci rôznych výskumno-vývojových projektov s cieľom uvedené fakty dokumentovať. Prvá podkapitola popisuje základné metódy spektrálnej analýzy a výpočtu FFT na DSP5600x. Druhá podkapitola uvádza základné rovnice potrebné pre využitie koprocesorov VCOP, FCOP a CCOP v procesore DSP56305 na ekvalizáciu a šifrovanie v systéme GSM. Hlavným cieľom je poukázať *na význam teórie ČSS pri efektívnom mapovaní alogoritmov ČSS do štruktúry komerčných DSP*.

6.1 SPEKTRÁLNA ANALÝZA A ALGORITMY FFT

Spektrálna analýza je v technickej praxi široko využívaná metóda analýzy signálov a je založená predovšetkým na diskrétnej forme Fourierovej transformácie. DFT je pre signál $x[n]$ konečnej dĺžky N (t.j. $x[n]=0$ pre $n<0$ a $n\geq N$) definovaná vzťahom

$$X[k]=\sum_{n=0}^{N-1}x[n]e^{-j\frac{2\pi kn}{N}} \qquad k=0,1,\ldots,N-1 \tag{6.1}$$

Pre inverznú DFT (IDFT) platí vzťah

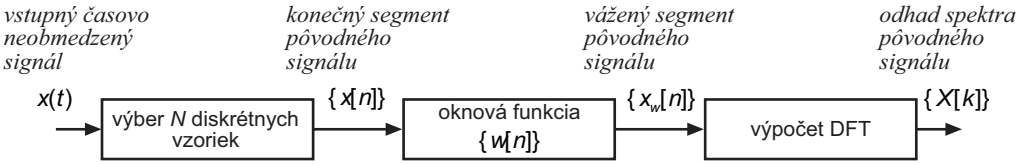
$$x[n]=\frac{1}{N}\sum_{k=0}^{N-1}X[k]e^{j\frac{2\pi kn}{N}} \qquad n=0,1,\ldots,N-1 \tag{6.2}$$

Algoritmy výpočtu DFT a IDFT a zvlášť ich optimalizované formy na báze FFT a inverznej FFT (IFFT) patria v oblasti algoritmov ČSS medzi základné algoritmy, pričom architektúry DSP sú prakticky od ich vzniku optimalizované pre ich podporu. Algoritmy FFT a IFFT sú využívané nielen na spektrálnu analýzu, ale aj napr. na efektívny výpočet rýchlej *kruhovej konvolúcie* (circular convolution) v spektrálnej oblasti a slúžia ako základný stavebný blok moderných modulačných metód OFDM (Orthogonal Frequency Division Multiplex) [181], [182] ktoré sa využívajú napr. v perspektívnych modemoch xDSL [67], [68]. Aj keď optimalizácia algoritmov FFT z hľadiska limitovanej veľkosti interných pamätí bola aktuálna predovšetkým pri klasických DSP, je otázka potlačenia vplyvu zaokrúhľovacích chýb aktuálna aj v moderných DSP vzhľadom na malú šírku slova a aritmetiku v pevnej rádovej čiarke, ktorú tieto DSP stále využívajú.

Popri týchto základných algoritmoch sú v spektrálnej analýze využívané aj pomocné funkcie, z ktorých najčastejšie využívané sú tzv. oknové funkcie. Využitie týchto algoritmov a funkcií v praktických implementáciach vyžaduje optimalizáciu predovšetkým z hľadiska malých interných pamätí a zvýšenia presnosti výpočtu.

6.1.1 VÝPOČET OKNOVÝCH FUNKCIÍ

Oknové funkcie nachádzajú široké využite v oblasti spektrálnej analýzy a umožňujú znížiť vplyv konečnej dĺžky analyzovaného segmentu signálu pri odhade spektra pôvodného, časovo neobmedzeného signálu. Základný princíp využitia oknovej funkcie v spektrálnej analýze je znázornený na obr. 6.1 .



Obr. 6.1 Princíp využitia oknovej funkcie v spektrálnej analýze

V praxi sa využíva veľké množstvo rôznych oknových funkcií [130], [131]. Medzi najčastejšie používané patria oknové funkcie založené na K-prvkovom kosínusovom rozvoji určené vzťahom[33]

$$w[n] = \sum_{m=0}^{K-1} a_m \cos\left(\frac{2\pi mn}{N}\right) \qquad n = 0,1,\ldots,N-1 \tag{6.3}$$

6.1.1.1 PRÍKLADY KOSÍNUSOVÝCH OKNOVÝCH FUNKCIÍ

Medzi najznámejšie oknové funkcie na báze kosínusového rozvoja patrí Hammingove okno s parametrami

$$a_0 = 0{,}54 \quad a_1 = -0{,}46 \tag{6.4}$$

Hannovo okno s parametrami

[33] Tento vzťah vyjadruje nesymetrickú oknovú funkciu (prvá a posledná vzorka nie sú zhodné), čo vyplýva zo základných vlastností DFT. V literatúre je často možné nájsť aj vyjadrenia oknových funkcií v symetrickom tvare, pričom tieto sa používajú napr. pri návrhu FIR filtrov pomocou oknových funkcií.

$$a_0 = 0{,}5 \quad a_1 = -0{,}5 \tag{6.5}$$

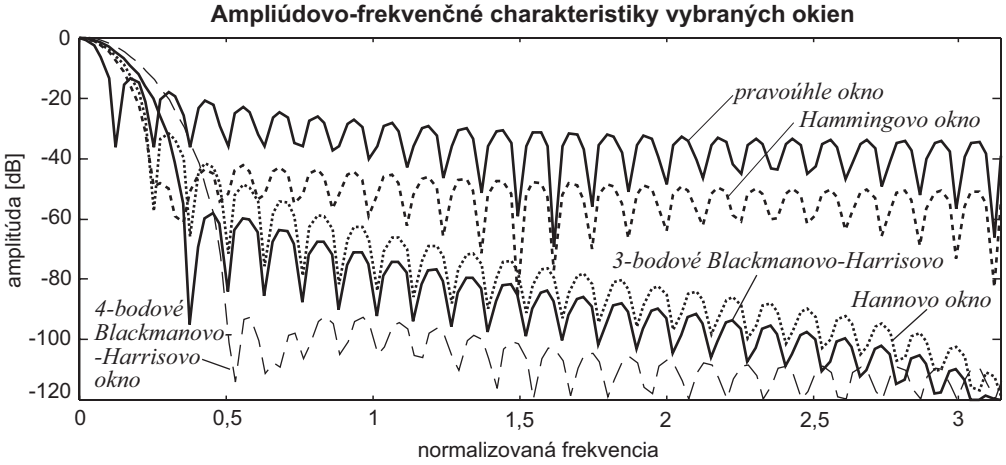
tzv. trojbodové Blackmanovo-Harrisovo okno (3BH) s parametrami

$$a_0 = 0{,}42 \quad a_1 = -0{,}50 \quad a_2 = 0{,}08 \tag{6.6}$$

a tzv. štvorbodové Blackmanovo-Harrisovo okno (4BH) s parametrami

$$a_0 = 0{,}35875 \quad a_1 = -0{,}48829 \quad a_2 = 0{,}14128 \quad a_3 = -0{,}01168 \tag{6.7}$$

Amplitúdovo-frekvenčné charakteristiky jednotlivých oknových funkcií sú znázornené na obr. 6.2 , ktorý naznačuje, že zmenou jednotlivých koeficientov je možné spektrálne vlastnosti okna výrazne meniť. Výber vhodného okna je v praxi ovplyvnený konkrétnou aplikáciou. Veľké potlačenie postranných lalokov 4BH okna bolo napr. využité na potlačenie vplyvu vyšších harmonických v prístroji na analyzovanie harmonického skreslenia sieťového napätia [132] na báze adaptívneho Goertzelovho algoritmu [133], [134]. Vplyv relatívne veľkej šírky hlavného laloku (z pohľadu aplikácie oknovej funkcie nevýhodná vlastnosť) bolo možné v tejto aplikácii eliminovať výberom vhodnej dĺžky spracovávaného stacionárneho signálu.



Obr. 6.2 Amplitúdovo-frekvenčné charakteristiky niektorých okien na báze kosínusového rozvoja

6.1.1.2 OPTIMALIZÁCIA PRE DSP

Z pohľadu implementácie na DSP je potrebné optimalizovať predovšetkým pamäťové nároky algoritmu a rýchlosť generovania vzoriek. V praktických aplikáciách je často potrebné generovať oknové funkcie, ktorých dĺžka presahuje veľkosť inštalovaných pamätí. Oknovú funkciu je v týchto prípadoch potrebné generovať priamo v reálnom čase.

Vzťah (6.3) nie je z pohľadu implementácie na DSP vhodný, pretože vyžaduje v prípade *K*-prvkového kosínusového okna výpočet *N(K-1)* hodnôt cos(.) , pričom výpočet goniometrických funkcií je z pohľadu DSP relatívne zložitá operácia. S využitím goniometrických vzťahov [135]

$$\cos(m\alpha) = \cos^m(\alpha) - \binom{m}{2}\sin^2(\alpha)\cos^{m-2}(\alpha) + \binom{m}{4}\sin^4(\alpha)\cos^{m-4}(\alpha) - \ldots \tag{6.8}$$

$$\sin^2(\alpha)+\cos^2(\alpha)=1 \tag{6.9}$$

je možné transformovať vzťah (6.3) do tvaru

$$w[n]=\sum_{m=0}^{K-1}h_m\cos^m\left(\frac{2\pi n}{N}\right)=\sum_{m=0}^{K-1}h_m c^m[n] \qquad n=0,1,\ldots,N-1 \tag{6.10}$$

pričom

$$c[n]=\cos\left(\frac{2\pi n}{N}\right)=\cos(\delta[n]) \tag{6.11}$$

a tým znížiť počet potrebných hodnôt $\cos(.)$ na hodnotu *N*. Použité mocninové operácie sú z pohľadu implementácie pomocou DSP podstatne jednoduchšie a je ich možné efektívne implementovať napr. štandardným Hornerovým algoritmom výpočtu v tvare

$$w[n]=h_0+c[n]\left(h_1+c[n]\left(h_2+\ldots\right)\right) \tag{6.12}$$

Hodnoty koeficientov h_m pre niektoré typické oknové funkcie sú uvedené v tab. 6.1 .

Tab. 6.1 Transformované koeficienty vybraných oknových funkcií

Koeficient	Hannovo	Hammingovo	3BH	Flattop	Taylorovo
h_0	0,5	0,54	0,34	0,03052	0,430502
h_1	-0,5	-0,46	-0,5	-0,5	-0,486803
h_2	–	–	0,16	0,46948	0,07893
h_3	–	–	–	–	0,002136
h_4	–	–	–	–	0,008064
h_5	–	–	–	–	0,00528
h_6	–	–	–	–	-0,010112
h_7	–	–	–	–	-0,00832

Výpočet hodnôt $\cos(.)$ je možné realizovať rôznymi spôsobmi:

a) Tabuľková metóda

Táto metóda je výhoda pokiaľ je možné rozložiť potrebné (ekvidištantné) hodnoty $\delta[n]$ do tvaru

$$\delta[n]=(l+mL)\frac{2\pi}{N} \quad n=l+mL,\ l=0,1,\ldots,L-1 \quad m=0,1,\ldots,M-1,\ N=LM \tag{6.13}$$

pričom množina $2(L+M)$ hodnôt (prípadne menej, pokiaľ sa využije symetria)

$$A_c[l]=\cos(l\frac{2\pi}{N}),\quad A_s[l]=\sin(l\frac{2\pi}{N}) \quad l=0,1,\ldots,L-1 \tag{6.14}$$

$$B_c[m]=\cos(m\frac{2\pi}{M}),\quad B_s(m)=\sin(m\frac{2\pi}{M}) \quad m=0,1,\ldots,M-1 \tag{6.15}$$

je uložená v pamäti vo forme tabuľky a hodnoty $c[n]$ sa určujú podľa vzťahu

$$c[n] = A_c[l]B_c[l] - A_s[l]B_s[l] \tag{6.16}$$

Metóda umožňuje rýchle generovanie hodnôt $c[n]$ s dostatočnou presnosťou. Nutným predpokladom je, aby rozmer okna nebol prvočíslom, čo je v prípade napr. okien, ktorých veľkosť je mocninou dvoch vždy splnené. Metóda je vhodná pre menšie rozmery okien, pri ktorých sú jej pamäťové nároky nízke. V prípade zmeny rozmeru okna je potrebné tabuľky prepočítať, čo môže byť v určitých aplikáciách nevýhodné.

b) Mocninový rozvoj

Metóda vychádza zo štandardného rozvoja funkcie cos(.) do Taylorovho rozvoja a umožňuje realizovať výpočet $c[n]$ v tvare

$$c[n] = 1 - \frac{\delta^2[n]}{2!} + \frac{\delta^4[n]}{4!} - \frac{\delta^6[n]}{6!} + \ldots \tag{6.17}$$

Základnou vlastnosťou tejto metódy je možnosť generovať výsledné hodnoty aj v neekvidištantných intervaloch čo sa často využíva v číslicovo riadených oscilátoroch (napr. pri realizácii fázových závesov). Výhodnou vlastnosťou sú aj malé pamäťové nároky tejto metódy. Nevýhodou metódy je vyššia výpočtová náročnosť.

c) Číslicový oscilátor

Táto metóda umožňuje využiť niektoré výhodné vlastnosti dvoch predchádzajúcich metód a pri malých pamäťových nárokoch a malej výpočtovej náročnosti generovať presné hodnoty $c[n]$. Základná forma číslicového oscilátora vychádza zo vzťahu [136]

$$c[n] = 2\cos\left(\frac{2\pi}{N}\right)c[n-1] - c[n-2] \tag{6.18}$$

ktorý umožňuje pri vhodne zvolených počiatočných podmienkach generovať sekvenčné hodnoty $c[n]$ s použitím len jedného násobenia a sčítania, čo je z pohľadu implementácie veľmi výhodné. Podstatnou nevýhodou, ktorá sa v prípade *sekvenčného generovania* hodnôt prejaví veľmi výrazne, je *vysoká citlivosť* štruktúry na kvantovanie koeficientov. Táto vlastnosť sa prejavuje postupným narastaním chýb v generovaných hodnotách a je pre väčšie dĺžky okna (rádovo desiatky tisíc vzoriek) prakticky nepoužiteľná.

Z pohľadu citlivosti na zaokrúhľovacie chyby sú štruktúry založené na opise pomocou *stavových premenných* podstatne vhodnejšie, pričom generovanie hodnôt $c[n]$ je možné implementovať pomocou viazaného oscilátora opísaného vzťahom

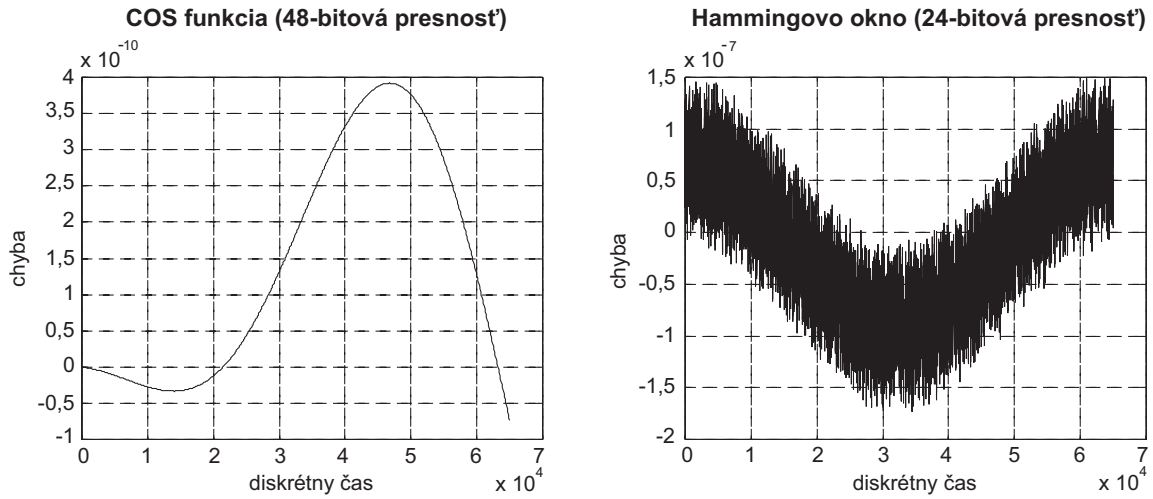
$$\begin{bmatrix} c[n] \\ s[n] \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{2\pi}{N}\right) & -\sin\left(\frac{2\pi}{N}\right) \\ \sin\left(\frac{2\pi}{N}\right) & \cos\left(\frac{2\pi}{N}\right) \end{bmatrix} \begin{bmatrix} c[n-1] \\ s[n-1] \end{bmatrix} \tag{6.19}$$

pričom

$$s[n] = \sin\left(\frac{2\pi n}{N}\right) \qquad pre \quad c[0] = 1, \quad s[0] = 0 \tag{6.20}$$

a hodnoty $c[n]$ sú určené vzťahom (6.11). Štruktúra výpočtu viazaného oscilátora obsahuje štyri násobenia a dve sčítania.

V prípade požiadavky generovať oknovú funkciu s presnosťou blížiacou sa dĺžke slova použitého DSP je potrebné realizovať výpočty podľa vzťahu (6.19) s vyššou presnosťou ako je dĺžka slova. S výhodou je možné využiť výpočty v dvojnásobnej presnosti, čo niektoré DSP (napr. DSP56002) podporujú v rámci špeciálneho režimu dátových ciest. Hodnoty chyby výstupu kosínusovej zložky viazaného oscilátora pre $N = 65000$ a implementáciu v 48-bitovej dvojnásobnej presnosti procesora DSP56002 a 24-bitových hodnôt výstupnej Hammingovej oknovej funkcie sú znázornené na obr. 6.3 [137].



a) b)

Obr. 6.3 Implementácia oknovej funkcie pomocou DSP56002: a) chyba výpočtu kosínusovej zložky viazaného oscilátora pre $N = 65000$ a 48-bitovú presnosť b) chyba výstupnej 24-bitovej hodnoty Hammingovej oknovej funkcie pre $N = 65000$

Priebeh chyby výstupu kosínusovej zložky viazaného oscilátora potvrdzuje, že dominantný je vplyv posunu pólov číslicového oscilátora a vplyv zokrúhľovacích chýb výpočtu je prakticky zanedbateľný. Celková chyba výstupnej oknovej funkcie je menšia ako 2 LSB bity v procesore DSP56002 ($1\ \mathrm{LSB} \approx 1{,}2 \times 10^{-7}$).

Uvedené metódy (okrem výpočtov využívajúcich mocninový rozvoj, ktoré budú súčasťou inej špecializovanej knižnice) sú súčasťou knižnice *libfft* ako súčasť podporných funkcií pre spektrálnu analýzu. Rýchlosť optimalizovanej vektorovej knižničnej C funkcie pre dĺžku bloku $M = 1000$ výstupných vzoriek oknovej funkcie, ktorá využíva viazaný oscilátor je uvedená v tab. 6.2 [137], [138] a dokumentuje jej vhodnosť pre analýzu signálov akustického pásma v reálnom čase.

Tab. 6.2 Rýchlosť optimalizovanej vektorovej knižničnej C funkcie pre DSP56002 a rôzne typy oknových funkcií (66MHz verzia DSP56002 vykoná 66 miliónov cyk/s)

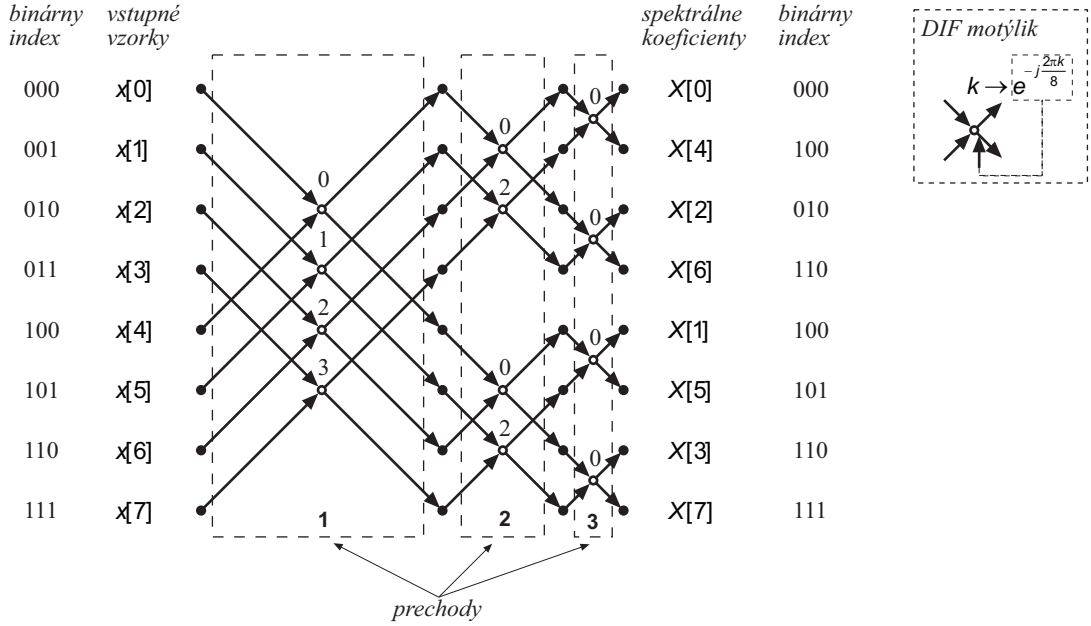
Veľkosť bloku	$K = 2$	$K = 3$	$K = 4$	$K = 8$
M=1000	68800 cyk	72812 cyk	76818 cyk	88836 cyk

6.1.2 ALGORITMY FFT

Algoritmy FFT a IFFT sú rýchlostne optimalizované algoritmy výpočtu DFT a IDFT. Existuje množstvo algoritmov FFT [139], z ktorých sa v oblasti DSP využívajú (z dôvodu podpory špeciálnych adresových režimov AAU) predovšetkým FFT a IFFT pre

$$N = 2^z \qquad z \in \{1, 2, 3 \ldots\} \tag{6.21}$$

ktoré využívajú známy rozklad FFT rozmeru N na dve FFT rozmeru $N/2$ a jeho rekurzívne opakovanie až do $N = 4$ alebo $N = 2$ a realizáciu FFT s rozmerom 4 resp. 2 pomocou tzv. RADIX 4 resp. RADIX 2 motýlikov. Výber vhodnej formy motýlika (RADIX 2 alebo RADIX 4) závisí predovšetkým na štruktúre dátových ciest konkrétneho DSP a tiež na polohe motýlika v procese výpočtu. Na obr. 6.4 je znázornený rozklad FFT pre $N = 8$, ktorý sa skladá z troch *prechodov*.

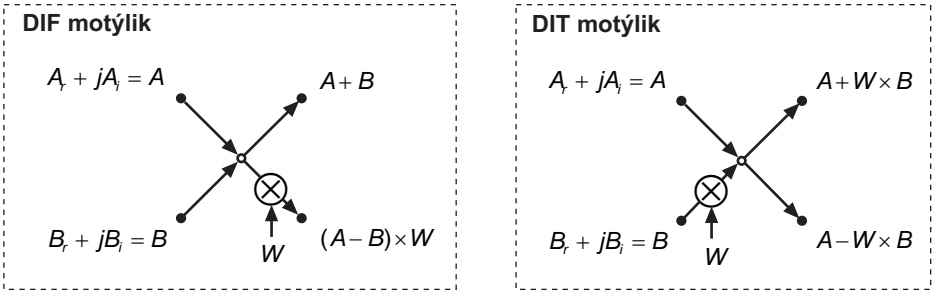


Obr. 6.4 Rozklad FFT s $N = 8$ na jednotlivé prechody

Veľmi často napr. rýchlostne optimalizované verzie výpočtu FFT využívajú v prvom prechode RADIX 4 motýliky a v ďalších prechodoch RADIX 2 motýliky.

Výpočty FFT a IFFT sa z hľadiska štruktúry algoritmu líšia len nepatrne a pri všeobecnej analýze algoritmov sa zvyčajne uvažuje len jeden z nich (FFT). Tento prístup je využitý aj v ďalšom opise, niektoré špecifické vlastnosti IFFT sú uvedené v samostatnej podkapitole.

V procese výpočtu FFT pomocou uvedených algoritmov dochádza k preusporiadaniu výstupných hodnôt do bitovo-reverzného usporiadania (tzv. DIF – decimácia vo frekvenčnej oblasti), čo je možné kompenzovať preusporiadaním v časovej oblasti (tzv. DIT – decimácia v časovej oblasti). Tieto všeobecne známe formy výpočtu FFT využívajú odlišné motýliky zobrazené na obr. 6.5 [136]. Vzhľadom na inštrukčnú sadu DSP nie sú tieto motýliky rovnocenné a napr. DSP5600x umožňuje realizovať jadro DIT motýlika pomocou 6 inštrukcií a jadro DIF motýlika pomocou 7 inštrukcií [140]. Táto vlastnosť naznačuje, že vhodnosť špecifickej formy implementácie FFT pre konkrétny typ DSP je potrebné analyzovať individuálne.



Obr. 6.5 DIF a DIT motýliky

6.1.2.1 PAMÄŤOVÁ OPTIMALIZÁCIA

Základný problém predovšetkým starších typov DSP je obmedzená veľkosť interných pamätí, čo vedie v prípade súčasného prístupu k trom externým pamätiam (v prípade duálnej harvardskej architektúry) až k trojnásobnému zníženiu pripustnosti DSP, pretože prístup k externej adresovej a dátovej zbernici musí byť multiplexovaný. Naviac, v praxi sú externe pripojené pamäte často pomalšie ako interné pamäte v DSP a prístup k nim musí byť spomalený vložením ďalších čakacích cyklov.

Pamäťovú optimalizáciu je možné riešiť predovšetkým nasledujúcimi spôsobmi:

a) Interné dátové pamäte

Pokiaľ je veľkosť interných dátových pamätí harvardskej architektúry obmedzená na hodnotu $2\times D$ (napr. 2×256 slov pre DSP5600x), je z hľadiska architektúry optimálne realizovať algoritmus FFT s rozmerom $N \le D$, čo je pre typické aplikácie veľmi nízka hodnota. V prípade rýchlostne optimalizovaného kódu FFT pre $N > D$ je výhodné využiť rozklad FFT s rozmerom *N* na FFT s menším rozmerom *D*, čo je možné realizovať napr. *výpočtom záverečných prechodov* výpočtu FFT v interných pamätiach, pričom prvý a druhý prechod je možné zlúčiť a použiť jeden prechod s RADIX 4 motýlikmi. Vhodnosť konkrétnej metódy závisí na vlastnostiach architektúry cieľového DSP, efektivite práce s externou pamäťou a veľkostiach dostupných externých pamätí.

b) Rozklad na menšie transformácie

Rozklad na menšie transformácie je možné realizovať aj pomocou metód, ktoré rozkladajú (jednorozmernú) transformáciu DFT s rozmerom $N = N_1 N_2$ na postupné počítanie DFT rozmerov N_1 a N_2 [32], [136]. Na základe substitúcií

$$k = m + nN_1 \qquad m,t = 0,1,\ldots,N_1 - 1 \tag{6.22}$$

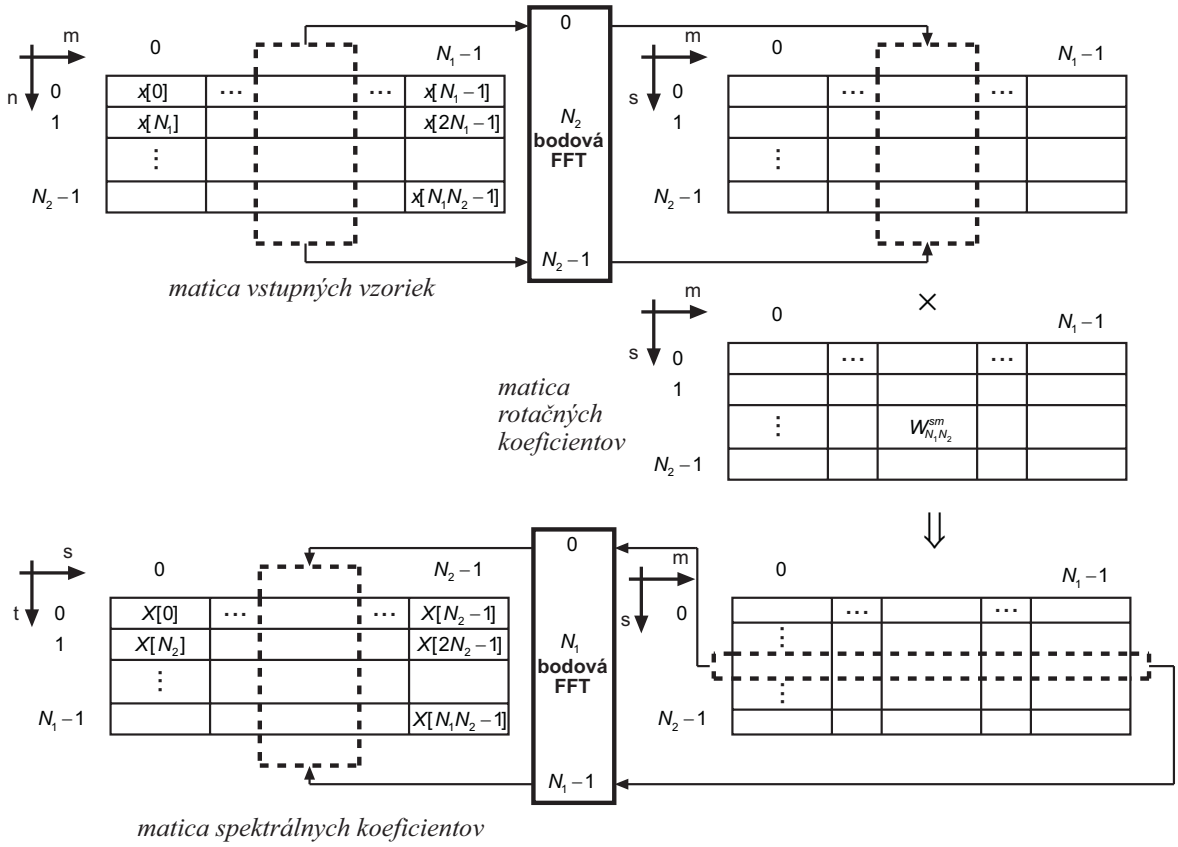
$$l = s + tN_2 \qquad n,s = 0,1,\ldots,N_2 - 1 \tag{6.23}$$

je možné prepísať výpočet DFT do tvaru

$$X[s+tN_2] = \sum_{m=0}^{N_1-1}\sum_{n=0}^{N_2-1} x[m+nN_1]W_{N_1N_2}^{(m+nN_1)(s+tN_2)} = \sum_{m=0}^{N_1-1} W_{N_1}^{tm} W_{N_1N_2}^{sm} \sum_{n=0}^{N_2-1} x[m+nN_1]W_{N_2}^{sn} \tag{6.24}$$

Výpočet DFT s rozmerom $N = N_1 N_2$ je možné podľa rovnice (6.24) interpretovať ako postupné počítanie DFT s dĺžkami N_1 a N_2, čo je znázornené na obr. 6.6 . Vstupná postupnosť $x[n]$ je uložená v matici $N_1 \times N_2$. V prvom kroku sa počíta N_1 DFT zo stĺpcov dĺžky N_2. Výsledkom je matica rovnakého rozmeru, ktorá sa v druhom kroku

násobí *maticou rotačných koeficientov* $W_{N_1N_2}^{sm}$. V treťom kroku sa vypočíta N_2 DFT dĺžky N_1 z riadkov matice z predchádzajúceho kroku. Výsledná matica obsahuje spektrálne koeficienty DFT s rozmerom $N = N_1N_2$. Násobenie maticou rotačných koeficientov je možné presunúť aj pred výpočet DFT [136]. V praxi je výpočet DFT rozmerov N_1 a N_2 realizovaný algoritmami FFT, pričom pri dostatočne malých rozmeroch je možné pri ich výpočte využiť interné dátové pamäte DSP. Nevýhodou je nutnosť uchovávať veľkú maticu rotačných koeficientov a zvýšené nároky na presun dát.



Obr. 6.6 Princíp využitia N_1 a N_2 rozmerných DFT na výpočet $N = N_1N_2$ rozmernej DFT

Vhodnosť konkrétnej metódy opäť závisí na vlastnostiach architektúry cieľového DSP, efektivite práce s externou pamäťou a veľkostiach dostupných externých pamätí.

c) Veľkosť kódu

Optimalizácia algoritmu FFT, ktorý využíva interné dátové pamäte vyžaduje samostatne optimalizovaný kód pre každý rozmer *N* algoritmu FFT. Tento kód je naviac relatívne dlhý. Rýchlostne optimalizovaný kód je tak výhodné využiť predovšetkým v aplikáciách, ktoré sú z hľadiska rýchlosti kritické, pričom je potrebné mať k dispozícii dostatočne veľkú programovú pamäť.

Pokiaľ je cieľom optimalizovať rýchlosť algoritmov FFT a zároveň aj veľkosť programového kódu (t.j. umožniť pomocou jedného programového kódu realizovať výpočty FFT pre rôzne N), je výsledný kód pomalší. Praktické výsledky pre DSP5600x sú

uvedené v kap. 6.1.2.5. Vo všeobecnosti je možné očakávať spomalenie oproti rýchlostne optimalizovaným algoritmom o niekoľko desiatok percent.[34]

d) Veľkosť dátových pamätí

Pokiaľ je celková veľkosť dátových pamätí obmedzená je výhodné vykonať výpočet FFT s *nahradzovaním* (inplace execution), t.j. spôsobom, ktorý vstupné dáta prepíše výstupnými dátami (výsledkami). Prístup k preusporiadaným vektorom je u DSP umožnený využitím špeciálneho režimu AAU. Takýto prístup je možný, komplikuje však štruktúru algoritmu napr. z pohľadu programovania vo vyššom programovacom jazyku.

Algoritmy výpočtu FFT typicky využívajú tabuľky *komplexných rotačných faktorov* (twiddle factors), zvyčajne v tvare

$$e^{-j\frac{2\pi k}{N}} = \cos\left(\frac{2\pi k}{N}\right) - j\sin\left(\frac{2\pi k}{N}\right) \qquad k = 0,1,\ldots,V-1 \tag{6.25}$$

ktoré vyžadujú pamäť $2V$ slov. Pre praktické implementácie je veľkosť tabuliek $V = N/2$, alebo v prípadne kvalitných implementácií len $V = N/4$ komplexných hodnôt, čo už možno považovať v praktických implementáciách za veľmi nízku hodnotu.

6.1.2.2 ZVÝŠENIE PRESNOSTI

Výpočet FFT vyžaduje veľké množstvo výpočtov a výsledok výpočtu je preto ovplyvnený zaokrúhľovacími chybami. Z pohľadu implementácie na DSP s pevnou rádovou čiarkou je situácia naviac komplikovaná možnosťou pretečenia výsledkov aritmetických operácií, ktoré môžu zvýšiť chybu celkového algoritmu FFT. Pri programovaní DSP sa používajú tieto základné metódy potlačenia možnosti pretečenia:

a) Ochranné bity

Táto metóda vychádza z faktu, že pre amplitúdovo obmedzený (vo všeobecnosti komplexný) vstup FFT algoritmu

$$|x[n]| \le 1 \qquad n = 0,1,\ldots,N-1 \tag{6.26}$$

platí

$$|X[k]| \le \sum_{n=0}^{N-1} |x[n]| \cdot \left| e^{-j\frac{2\pi kn}{N}} \right| \le N \qquad k = 0,1,\ldots,N-1 \tag{6.27}$$

a teda postačujúcou podmienkou, aby mohli byť výsledky FFT $X[k]$ reprezentované v zlomkovom formáte DSP, je *zmena mierky* vstupného signálu tak, aby splňoval podmienku

$$|x[n]| \le \frac{1}{N} \tag{6.28}$$

čo je možné realizovať jednoduchým prenásobením vstupného signálu faktorom $1/N$. Z pohľadu samotného algoritmu FFT nie je potrebná žiadna modifikácia a tak táto metóda umožňuje dosiahnuť najvyššiu rýchlosť. Jednoduchá zmena mierky vstupného signálu však znižuje počet platných bitov vstupného signálu o hodnotu

[34] Z hľadiska optimalizácie kódu pre DSP je rozdiel niekoľkých desiatok percent považovaný za značný.

$$b_N = \lceil \log_2(N) \rceil \tag{6.29}$$

čo v prípade väčších hodnôt N, napr. 2048 predstavuje zníženie o hodnotu 11 bitov. V prípade napr. 16-bitového DSP ostáva na reprezentáciu $x[n]$ len 5 bitov. Ani v prípade 24-bitového DSP nie je situácia bezproblémová. V praxi sa často používajú aj väčšie rozmery FFT a tak uvedená metóda je pre väčšie hodnoty N prakticky nepoužiteľná.

b) Automatická zmena mierky

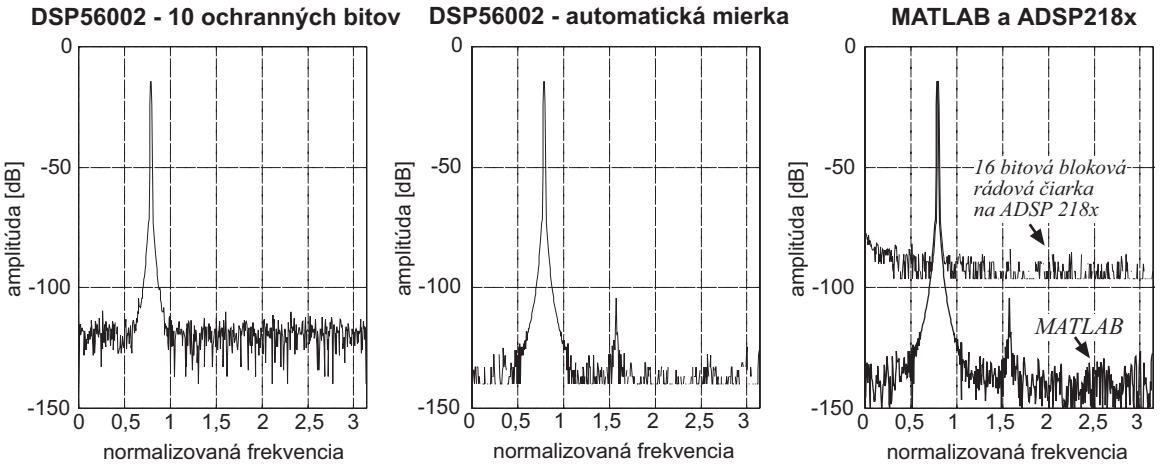
Namiesto zmeny mierky vstupných vzoriek je možné využiť postupné znižovanie jednotlivých medzivýsledkov. Túto operáciu je možné jednoducho realizovať počas výpočtu FFT motýlika. Z obr. 6.5 je zrejmé, že v prípade DIF aj DIT motýlika je zabezpečené, že pre vstupy $|A| \le 1$ a $|B| \le 1$ budú výstupy splňovať podmienku $|A'| \le 2$ a $|B'| \le 2$. Jednoduchým predelením výstupov každého motýlika faktorom 2 je možné zabezpečiť, že všetky medzivýsledky výpočtov FFT sú v absolútnej hodnote menšie ako 1.

Niektoré typy DSP umožňujú nastaviť dátové cesty do špeciálneho režimu, ktorý zabezpečí automatické delenie faktorom 2 bez zníženia priepustnosti DSP, čím je možné dosiahnuť rýchlosť výpočtu algoritmov FFT s automatickou zmenou mierky blížiacu sa (existujú určité obmedzenia, napr. nutnosť použiť len RADIX 2 motýliky) najrýchlejším algoritmom FFT bez zmeny mierky.

Výhodou automatickej zmeny mierky je presnejší výpočet FFT ako v prípade spracovania vstupného signálu s ochrannými bitmi čo je dokumentované na obr. 6.7 pre komplexný vstupný signál

$$x[n] = \sin(2\pi 0{,}1255n) + j2^{-15}\sin(2\pi 0{,}2505n) \tag{6.30}$$

Signál (6.30) bol kvantovaný na 16 bitov (typická rozlišovacia schopnosť A/D prevodníkov na báze sigma delta modulácie), vážený 3BH oknom a spracovaný 1024-bodovou komplexnou FFT. Ako referenčné sú použité výsledky z programu MATLAB vypočítané so 64-bitovou aritmetikou v pohyblivej rádovej čiarke.



Obr. 6.7 Porovnanie presnosti výpočtu FFT s ochrannými bitmi a automatickou zmenou mierky na 24-bitovom DSP5600x a blokovou pohyblivou rádovou čiarkou na 16-bitovom ADSP218x

Nevýhodou uvedenej metódy je automatická zmena mierky aj v prípade, že výstupy motýlikov sú menšie ako 1 a zmena mierky nie je potrebná, čo vo všeobecnosti vedie k väčším chybám ako je teoreticky dosiahnuteľné minimum.

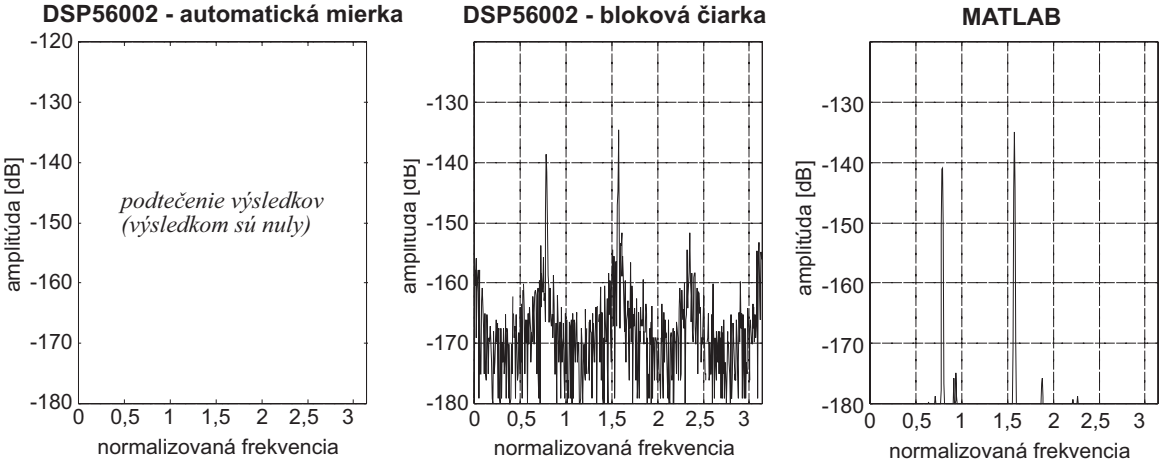
c) Reprezentácia v blokovej pohyblivej čiarke

Táto reprezentácia umožňuje bloku dát využívať spoločný exponent. V algoritme FFT je blokom súbor hodnôt v jednotlivých prechodoch algoritmu FFT. V prípade, že aspoň jeden výstup motýlika v rámci jedného prechodu má absolútnu hodnotu väčšiu ako 1, je potrebné predeliť všetky výstupy motýlikov v rámci celého prechodu (čím je zabezpečený spoločný exponent pre celý blok). Podmienená realizácia zmeny mierky vyžaduje testovanie celého bloku a je z pohľadu rýchlosti z uvedených metód najpomalšia.

Z pohľadu presnosti je táto metóda naopak najpresnejšia a jej presnosť sa blíži presnosti, ktorú je možné dosiahnuť v aritmetike s pohyblivou rádovou čiarkou pokiaľ má DSP s pevnou rádovou čiarkou dĺžku slova zhodnú s dĺžkou mantisy referenčného systému s pohyblivou rádovou čiarkou. Pokiaľ používa DSP s pevnou rádovou čiarkou menšiu dĺžku slova, je úroveň zaokrúhľovacích chýb podstatne vyššia, čo je dokumentované na obr. 6.7 na výstupe komplexnej FFT implementovanej pomocou 16-bitového ADSP218x [108]. Vhodnosť reprezentácie v blokovej pohyblivej rádovej čiarke sa prejaví predovšetkým pri spracovaní malých signálov, kedy automatická zmena mierky spôsobuje podtečenie výsledkov, pričom aritmetika v blokovej pohyblivej rádovej čiarke dosahuje výsledky porovnateľné s referenčnými výsledkami v MATLABe, čo je dokumentované na obr. 6.8 . pre komplexný vstupný signál

$$x[n] = 2^{-21}\sin(2\pi 0{,}1255n) + j2^{-20}\sin(2\pi 0{,}2505n) \tag{6.31}$$

ktorý bol kvantovaný na 24 bitov (tento signál môže v praktických aplikáciách reprezentovať interný signál zložitejších algoritmov), opäť vážený 3BH oknom a spracovaný 1024-bodovou komplexnou FFT. Výsledky potvrdzujú výhodnejšie numerické vlastnosti aritmetiky v blokovej pohyblivej rádovej čiarke, ktorá umožňuje zvýšenie dynamického rozsahu bez nutnosti zmeny mierky vstupného signálu.



Obr. 6.8 Porovnanie presnosti metódy výpočtu FFT s automatickou zmenou mierky a blokovou pohyblivou rádovou čiarkou pomocou 24-bitového DSP56002

DSP typicky podporujú výpočty v blokovej pohyblivej rádovej čiarke možnosťou testovania pretečenia v rámci aritmetických operácií (napr. 24-bitové DSP56xxx) prípadne zahrnutím špeciálnej jednotky pre normovanie priamo do dátových ciest (napr. ADSP21xx). Zvlášť v prípade 16-bitových DSP je podpora tejto reprezentácie veľmi často používaná vzhľadom na potrebu efektívne využiť relatívne krátku dĺžku slova.

6.1.2.3 VÝPOČET REÁLNEJ FFT

Reálna FFT (RFFT) je výpočet FFT pre postupnosť $x[n]$, ktorá je čisto reálna. V literatúre je možné nájsť celý rad optimalizovaných algoritmov pre výpočet RFFT [141], [142], [143], [144]. Ich cieľom je znížiť počet niektorých operácií (napr. násobení), počet presunov, prípadne zvýšiť rýchlosť na zreťazených procesoroch. Tieto formy optimalizácie sú však z pohľadu architektúr DSP na báze harvardskej architektúry nevýhodné, v prípade VLIW architektúr by však tieto formy mohli byť zaujímavé.

Pre modifikované harvardské architektúry je výhodné využiť linearitu FFT a Hermitovskú symetriu spektra reálnej postupnosti $x[n]$ v tvare

$$X[k] = X^*[-k] = X^*[N-k] \tag{6.32}$$

pričom pre výpočet RFFT sa využívajú predovšetkým optimalizované algoritmy FFT:

a) Výpočet dvoch *N* rozmerných RFFT pomocou jednej *N* rozmernej FFT

Na základe linearity DFT platí pre reálne postupnosti $x[n]$ a $y[n]$ vzťah [142]

$$DFT\big(z[n]\big) = Z[k] = DFT\big(x[n] + jy[n]\big) = Z_r[k] + jZ_i[k] =$$

$$= \big(X_r[k] - Y_i[k]\big) + j\big(X_i[k] + Y_r[k]\big) \tag{6.33}$$

pričom indexy *r* a *i* označujú reálnu a imaginárnu zložku. Aplikovaním vzťahu (6.32) na $Z[k]$ je možné určiť DFT pôvodných reálnych postupností

$$X[k] = X_r[k] + jX_i[k] = \frac{1}{2}\big(Z_r[k] + Z_r[N-k]\big) + \frac{1}{2}\big(Z_i[k] - Z_i[N-k]\big) \tag{6.34}$$

$$Y[k] = Y_r[k] + jY_i[k] = \frac{1}{2}\big(Z_i[N-k] + Z_i[k]\big) + j\frac{1}{2}\big(Z_r[N-k] - Z_r[k]\big) \tag{6.35}$$

pre $k = 0,1,\ldots,N/2$ [35]. Takto je možné priamo využiť optimalizované algoritmy FFT s následnou jednoduchou modifikáciou výsledkov podľa vzťahov (6.34) a (6.35).

b) Výpočet *N* rozmernej RFFT pomocou *N*/2 rozmernej FFT

Tento spôsob vychádza zo štandardného DIT rozkladu FFT algoritmu dĺžky *N* v tvare

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi nk}{N}} = \sum_{n=0}^{N/2-1} x[2n] e^{-j\frac{2\pi nk}{N/2}} + e^{-j\frac{2\pi k}{N}} \sum_{n=0}^{N/2-1} x[2n+1] e^{-j\frac{2\pi nk}{N/2}} \tag{6.36}$$

pre $k = 0,1,\ldots,N-1$, čo je možné realizovať pomocou dvoch RFFT dĺžky *N*/2 pre reálne postupnosti $\{x[2n]\}$ a $\{x[2n+1]\}$ a následnou kombináciou výsledkov podľa vzťahu (6.36). Výpočet RFFT dvoch reálnych postupností dĺžky *N*/2 je možné realizovať pomocou FFT komplexnej postupnosti

$$z[n] = x[2n] + jx[2n+1] \tag{6.37}$$

---

[35] Pre k=0 a k=N/2 sú hodnoty $X[k]$ (a samozrejme aj $Y[k]$) čisto reálne, čo sa často v praktických implementáciach využíva na uloženie len N/2 (pre k=0,1,...,N/2-1) komplexných spektrálnych hodnôt pre každú postupnosť, pričom je uložená modifikovaná komplexná hodnota $X'[0]=X[0]+jX[N/2]$ pre $X[k]$ (podobne aj pre $Y[k]$).

s dĺžkou N/2 pomocou vzťahov (6.34) a (6.35).

c) Využitie symetrie

Existujú aj alternatívne metódy využitia Hermitovskej symetrie DFT reálnej postupnosti. Typickým algoritmom je Glenn-Berglandov algoritmus výpočtu RFFT [140]. Optimalizované DSP implementácie (minimálne pre DSP5600x [145]) však nedosahujú rýchlosť predchádzajúcich metód a v DSP praxi sa prakticky nevyužívajú.

6.1.2.4 VÝPOČET IFFT

Rovnice pre IFFT (6.2) a FFT (6.1) sú veľmi podobné. Faktor 1/*N* je v praktických implementáciach nepodstatný a je ho možné zahrnúť do zmeny mierky spracovávaných signálov. Rozdiel v rotačných faktoroch

$$e^{-j\frac{2\pi kn}{N}} \quad pre\ FFT \qquad a \qquad e^{j\frac{2\pi kn}{N}} \quad pre\ IFFT \tag{6.38}$$

je možné riešiť v praktických implementáciách predovšetkým nasledujúcimi spôsobmi:

a) FFT kód s odlišnými tabuľkami

Pri tejto metóde je programový kód pre FFT a IFFT *rovnaký*, každý z nich však využíva *odlišnú tabuľku rotačných faktorov*. V prípade, že je potrebné súčasne využívať FFT aj IFFT je potrebné mať v pamäti uložené súčasne dve tabuľky. Táto metóda je výhodná, pokiaľ je programový kód FFT väčší ako veľkosť tabuľky pre rotačné faktory. Naviac tabuľky je možné generovať aj počas vykonávania programu, čo v prípade programového kódu nie je prakticky realizovateľné.

b) Modifikovaný kód s jednou tabuľkou

Táto metóda je duálnou k predchádzajúcej metóde a umožňuje využívať *spoločnú tabuľku rotačných faktorov* a rozdielne programové kódy pre FFT a IFFT. Metóda je výhodná predovšetkým v prípade, že sa využíva univerzálny programový kód pre všetky rozmery FFT a IFFT a je potrebné využívať súčasne rôzne rozmery FFT a IFFT.

c) Výmena reálnej a imaginárnej časti

Táto metóda realizuje výpočet IFFT výmenou reálnej a imaginárnej časti vstupného vektora, výpočtom FFT a opätovnou výmenou reálnej a imaginárnej časti výstupného vektora, čo je zrejmé z nasledujúcich rovníc, ktoré opisujú výpočet vo FFT motýliku (W_r a W_i sú reálna a imaginárna časť rotačných faktorov, A_r a A_i sú reálny a imaginárny vstup FFT motýlika)

$$(A_r + jA_i)(W_r + jW_i) = (A_rW_r - A_iW_i) + j(A_iW_r + A_rW_i) \tag{6.39}$$

$$(A_i + jA_r)(W_r - jW_i) = j(A_rW_r - A_iW_i) + (A_iW_r + A_rW_i) \tag{6.40}$$

Nevýhodou tejto metódy je nutnosť realizácie dvojnásobnej výmeny reálnych a imaginárnych častí *N*-prvkových komplexných vektorov.

d) Súvislosťou medzi FFT a IFFT

Táto metóda využíva menej známu vlastnosť FFT transformácie, ktorú je možné zapísať v tvare [130], [146]

$$\begin{bmatrix} x[0] \\ x[N-1] \\ x[N-2] \\ \vdots \\ x[2] \\ x[1] \end{bmatrix} = \frac{1}{N} FFT_N \left(FFT_N \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-2] \\ x[N-1] \end{bmatrix} \right) \tag{6.41}$$

čo umožňuje využiť jeden programový kód a jednu tabuľku rotačných faktorov v aplikáciách, ktoré sú z hľadiska veľkostí dostupných pamätí kritické. Nevýhodou uvedenej metódy je nutnosť preusporiadania výstupného vektora, čo môže výrazne skomplikovať vysokoúrovňové programovanie DSP. Oproti predchádzajúcej metóde stačí preusporiadanie len výstupného vektora.

6.1.2.5 OPTIMALIZOVANÉ KNIŽNIČNÉ FUNKCIE PRE FFT

Väčšinu z naznačených *všeobecných prístupov* k implementácii FFT je možné využiť v rámci tvorby optimalizovaných vektorových knižníc *pre všetky typy* DSP. Pre konkrétny cieľový DSP, prípadne prekladač je naviac potrebné vykonať špecifické optimalizácie. Funkcie optimalizovanej knižnice *libfft* pre DSP5600x napr. využívajú len *obmedzenú množinu registrov* DSP5600x (aby bolo možné využiť zvyšné registre pre súbežnú prácu jadra operačného systému), *využívajú viacnásobné dynamické prekryvné segmenty* programového kódu do limitovanej internej programovej pamäte, obmedzene využívajú *limitovaný zásobník* DSP5600x a pod [137].

Z hľadiska predchádzajúcej analýzy je zaujímavé predovšetkým porovnanie rýchlosti algoritmov FFT a RFFT optimalizovaných z hľadiska rýchlosti (ktoré využívajú pre rôzne veľkosti N rôzny kód) s univerzálnym kódom (jeden kód pre všetky N). Počty cyklov dosiahnuteľné pomocou funkcií z knižnice libfft sú uvedené v tab. 6.3 pre žiadne čakacie stavy (0w) a jeden čakací stav (1w) pre prístup k externým SRAM pamätiam [138].

Tab. 6.3 Porovnanie rýchlosti rôznych verzií optimalizovaných FFT a RFFT

Rozmer FFT	Odlišný kód pre rôzne rozmery FFT				Jeden kód pre všetky rozmery FFT			
	Komplexná FFT		Reálna FFT		Komplexná FFT		Reálna FFT	
	cyk (0w)	cyk (1w)	cyk (0w)	cyk (1w)	cyk (0w)	cyk (1w)	cyk (0w)	cyk (1w)
256	15100	21543	9960	14062	23716	34564	13988	20107
512	32296	37929	19092	27304	50530	74138	27938	40641
1024	69652	82452	40192	49356	109216	160880	58336	85403
2048	156742	191928	84750	104431	236766	349638	124190	182701
4096	403212	515256	186144	234854	512284	757788	266076	392451
8192	768480	979064	403212	515256	1104218	1636447	570274	842597
16384	–	–	883902	1148107	–	–	1219552	1804219

Pre porovnanie rýchlosti je často využívaná FFT s rozmerom 1024. Najrýchlejšie publikované implementácie [140] vyžadujú 59989 cyklov (0w) pre komplexnú a 34886 cyklov (0w) pre reálnu FFT. Tieto implementácie sú napísané kompletne v asembleri, využívajú optimálne rozloženie pamätí a využívajú ochranné bity (t.j. umožňujú maximálnu optimalizáciu). Implementácie v tab. 6.3 sú z hľadiska užívateľa C funkcie (a teda vyžadujú určitú réžiu), pracujú so štandardným rozložením pamätí a využívajú dynamické prekryvné segmenty pre limitovanú internú programovú pamäť. Aj napriek týmto faktom dosahujú knižničné funkcie veľmi dobré výsledky. Údaje v tab. 6.3

umožňujú tiež porovnanie rozdielov medzi rýchlosťami implementácie v prípade rôznych čakacích stavov a jasne naznačujú, že len nepatrne pomalšie SRAM pamäte s jedným čakacím stavom (ktoré je veľmi často potrebné použiť pri taktovacích frekvenciách 80 MHz a viac) znižujú výkon DSP prakticky až o 20 %.

6.1.3 GOERTZELOV ALGORITMUS

Goertzelov algoritmus umožňuje vypočítať hodnotu spektrálneho DFT koeficientu $X[k]$ N-prvkovej postupnosti $x[n],\quad n = 0,1,\ldots,N-1$ ako výstup IIR filtra [136]

$$v^k[n] = 2\cos\left(\frac{2\pi k}{N}\right)v^k[n-1] - v^k[n-2] + x[n], \qquad v^k[-1] = v^k[-2] = 0 \tag{6.42}$$

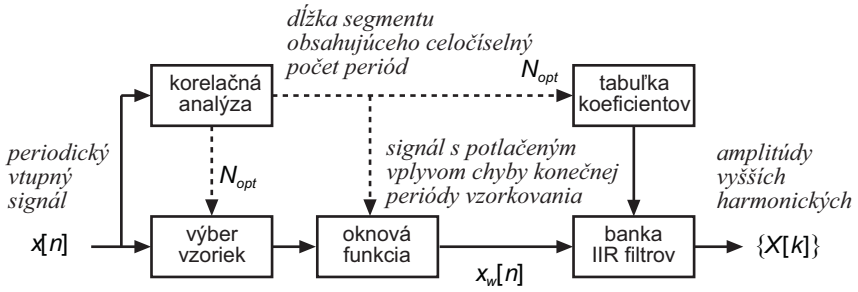
pre $n = 0,1,\ldots,N-1$ a výpočtom $X[k]$ na základe vzťahu

$$X[k] = v^k[N-1]e^{-j\frac{2\pi k}{N}} - v^k[N-2], \qquad k \in \{k_1, k_2, \ldots, k_K\} \tag{6.43}$$

Algoritmus má výpočtovú náročnosť $O(N^2)$ a je zložitejší ako algoritmy FFT, pričom z hľadiska výpočtovej náročnosti je vhodný, pokiaľ počet počítaných koeficientov splňuje podmienku

$$K \le \log_2(N) \tag{6.44}$$

Výhodnou vlastnosťou tohto algoritmu je možnosť vypočítať DFT koeficienty pre ľubovoľný rozmer N a ľubovoľnú hodnotu k s jednoduchou možnosťou preladenia banky IIR filtrov v prípade zmeny rozmeru N (bez nutnosti meniť štruktúru algoritmu, čo je napr. nutné v prípade špecializovaných algoritmov FFT pre všeobecné N). Táto vlastnosť bola výhodne využitá napr. v [133], [134] pri harmonickej analýze sieťového napätia pomocou adaptívneho Goertzelovho algoritmu, ktorého štruktúra je znázornená na obr. 6.9 [134].



Obr. 6.9 Harmonická analýza signálu s využitím adaptívneho Goertzelovho algoritmu

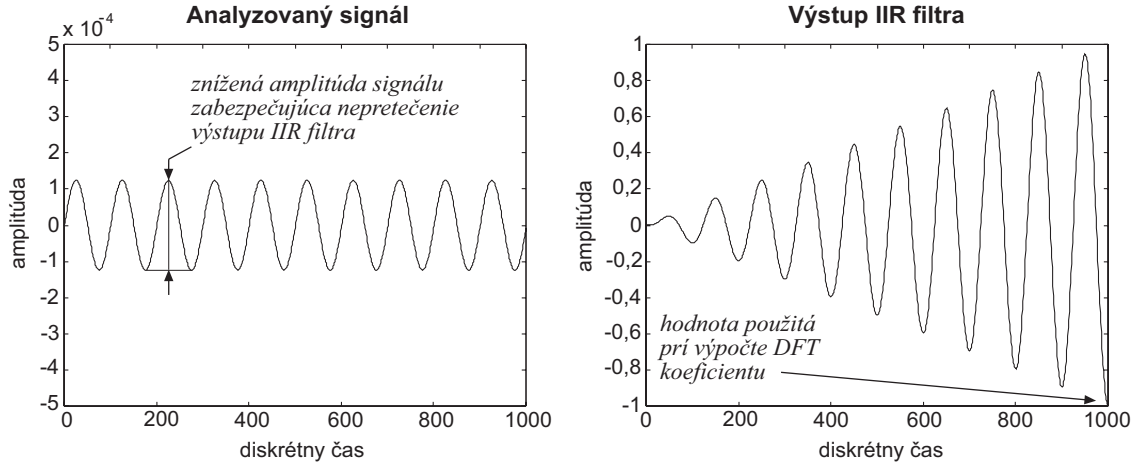
Principiálnou nevýhodou filtra v prípade aritmetiky s pevnou rádovou čiarkou je nutnosť použitia zmeny mierky na vstupe filtra tak, aby nedošlo k pretečeniu na výstupe jednotlivých IIR filtrov, čo je znázornené na obr. 6.10 pre vstupný analyzovaný signál

$$x[n] = \frac{1}{8000}\sin\left(\frac{2\pi n}{100}\right) \qquad n = 0,1,\cdots,N-1 \tag{6.45}$$

$N = 1000$ a výstup filtra $v^{10}[n]$ použitého pri výpočte koeficientu DFT

$$X[10] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi(10)n}{N}} \tag{6.46}$$

Zníženie amplitúdy vstupného signálu má z hľadiska presnosti rovnaké negatívne dôsledky ako využitie ochranných bitov pri výpočte FFT, pričom je dokonca potrebné použiť väčšie zmenšenie mierky ako v prípade algoritmu FFT.



Obr. 6.10 Priebeh výstupu IIR filtra $v^{10}[n]$ *pri výpočte DFT koeficientu* $X[10]$ *pomocou Goertzelovho algoritmu*

Vzhľadom na rezonančný charakter IIR filtrov, ktoré majú póly na jednotkovej kružnici, dochádza vplyvom kvantovania koeficientov aj k posunu pólov, čo vnáša do výpočtu ďalšie chyby. Implementácia v pohyblivej rádovej čiarke je v tomto prípade podstatne výhodnejšia a bola využitá aj v [133], [134].

6.1.4 CHIRP DFT

Táto metóda výpočtu patrí do kategórie konvolučných metód výpočtu DFT [184] (podobne ako Goertzelov algoritmus) a môže byť využitá ako výpočtová metóda na realizáciu spektrálnej lupy alebo výpočet DFT pre všeobecný rozmer *M* pomocou optimalizovaných FFT algoritmov s rozmerom, ktorý je mocninou dvoch. Existujú aplikácie metód ČSS, pre ktoré je výhodné využiť práve DFT všeobecného rozmeru *M* a tým znížiť zložitosť výsledného komplexnejšieho algoritmu. V prípade implementácie týchto algoritmov na DSP (ktoré sú optimalizované pre FFT s rozmerom rovným mocnine dvoch) je možné druhú vlastnosť Chirp DFT s výhodou využiť.

6.1.4.1 ZÁKLADNÝ PRINCÍP

Prvých *K* spektrálnych DFT koeficientov *M* prvkovej postupnosti $x[n] = 0,1,\ldots,M-1$ je možné určiť pomocou vzťahu

$$X[k] = \sum_{n=0}^{M-1} x[n] e^{-j\frac{2\pi kn}{M}} \quad k = 0,1,\ldots K-1 \tag{6.47}$$

pričom jeho priama aplikácia vyžaduje *MK* komplexných násobení a sčítaní. Zníženie počtu operácií je možne realizovať nasledujúcimi transformáciami [184]:

Zavedením

$$W = e^{j\frac{2\pi}{M}} \tag{6.48}$$

a použitím rovnosti

$$nk = 0{,}5\left[n^2 + k^2 - (k-n)^2\right] \tag{6.49}$$

je možné vzťah (6.47) zapísať v tvare

$$X[k] = W^{-0{,}5k^2} \sum_{n=0}^{M-1} x[n] W^{-0{,}5n^2} W^{0{,}5(n-k)^2} \tag{6.50}$$

Zavedením postupností

$$g[n] = x[n] W^{-0{,}5n^2} \tag{6.51}$$

$$f[n] = W^{0{,}5n^2} \tag{6.52}$$

je možné prepísať vzťah (6.50) do tvaru

$$X[k] = W^{-0{,}5k^2} \{g * f\}[k] \tag{6.53}$$

pričom znak $*$ reprezentuje konvolúciu.

6.1.4.2 IMPLEMENTÁCIA POMOCOU FFT ALGORITMU

Vzťah (6.53) je možné efektívne vypočítať pomocou kruhovej konvolúcie vo frekvenčnej oblasti pomocou algoritmov FFT pre rozmer, ktorý je mocninou dvoch. Postupnosť $\{g[n]\}$ je konečná a má rovnakú dĺžku ako postupnosť $\{x[n]\}$. Postupnosť $\{f[n]\}$ je nekonečná, vzhľadom na potrebu určiť len konečný počet K spektrálnych koeficientov, je potrebné využiť len segment $\{f[n], n = -M+1,\ldots,-2,-1,0,1,2,\ldots,K-1\}$ s konečnou dĺžkou.

Výsledný algoritmus sa skladá z nasledujúcich krokov:

1. nájdenie čísla L, ktoré je mocninou dvoch a pre ktoré platí

$$L \geq M + K - 1 \tag{6.54}$$

2. vygenerovanie M prvkovej postupnosti postupnosti $\{g[n]\}$ podľa vzťahu (6.51) a jej doplnenie nulami na dĺžku L,

3. vygenerovanie postupnosti $\{f[n], \quad n = 0,1,2,\ldots,K-1\}$ podľa vzťahu (6.52) a jej spojenie s postupnosťou $\{f[n], \quad n = -(L-K),\ldots,-2,-1\}$, ktorá je kruhovo posunutou verziou postupnosti $\{f[n]\}$ pre nezáporné indexy,

4. vykonanie operácie

$$IFFT\{FFT\{g[n]\} FFT\{f[n]\}\} \tag{6.55}$$

pomocou optimalizovaných algoritmov FFT a IFFT,

5. vynásobenie prvých K členov vzniknutej postupnosti korekčným členom podľa (6.53).

6.1.5 HARMONICKÁ ANALÝZA VIBRAČNÝCH SIGNÁLOV

Všeobecným pravidlom pri aplikácií algoritmov ČSS je snaha o využitie maxima vopred známych informácií o charaktere spracovávaného signálu. Typickým jednoduchým príkladom je spektrálna analýza časovo stacionárneho segmentu sieťového napätia rozvodnej siete 230 V/50 Hz. Tento signál vzhľadom na periodickú štruktúru obsahuje základnú a vyššie harmonické, ktorých poloha vo frekvenčnej oblasti sa mení v závislosti na (pomalých zmenách) frekvencie sieťového napätia. Cieľom je určiť hodnoty jednotlivých harmonických, ktoré sú zaujímavým diagnostickým parametrom [134]. Optimálne z hľadiska presnosti je spracovanie celočíselného počtu periód spracovávaného signálu, čo vyžaduje DFT analýzu vo všeobecnosti premenlivého počtu vzoriek [134].

Zložitejším príkladom je analýza *vibračného signálu* zariadení využívajúcich *rotačný princíp*, akým je napr. prevodovka s 2 hriadeľmi, ktorých prevod je určený vzťahom

$$p = \frac{z_2}{z_1} \tag{6.56}$$

pričom z_1 a z_2 sú počty zubov ozubených kolies umiestnených na jednotlivých hriadeľoch a súvislosť medzi počtom otáčok n_1 a n_2 jednotlivých hriadeľov je určená vzťahom

$$\frac{n_1}{n_2} = \frac{z_2}{z_1} \tag{6.57}$$

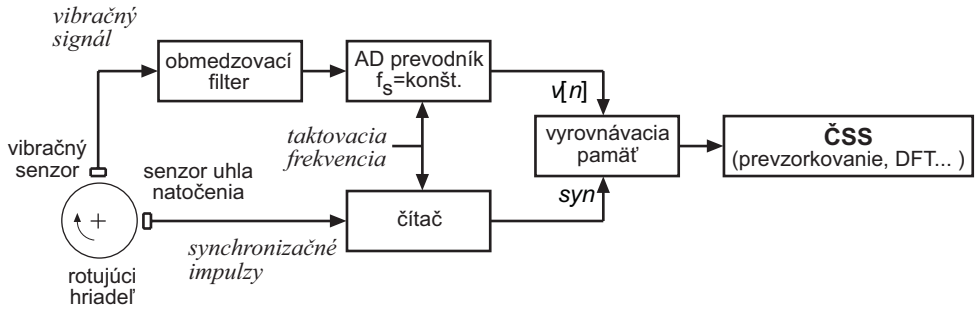
V prípade, že hriadele rotujú konštantnou uhlovou rýchlosťou, je možné v spektre identifikovať typické spektrálne zložky s rôznym mechanizmom vzniku [147]. Tieto zložky sú uvedené v tab. 6.4 pričom hodnoty n_1 a n_2 sú vyjadrené v otáčkach za minútu.

Tab. 6.4 Spektrálne zložky vibračného signálu prevodovky s 2 hriadeľmi

Význam	Vstupný hriadeľ	Výstupný hriadeľ
Frekvencia otáčania hriadeľa a jej vyššie harmonické	$f_{s1}[k] = k\frac{n_1}{60}, \quad k = 1,2,...$	$f_{s2}[k] = k\frac{n_2}{60}, \quad k = 1,2,...$
Zubová frekvencia a jej vyššie harmonické	$f_{gm}[k] = f_{s1}[k]z_1, \quad k = 1,2,...$	$f_{gm}[k] = f_{s2}[k]z_2, \quad k = 1,2,...$
Postranné pásma zubových frekvencií	$f_{sb1}[k,l] = f_{gm}[k] \pm lf_{s1}, k,l = 1,2,...$	$f_{sb2}[k,l] = f_{gm}[k] \pm lf_{s2}, k,l = 1,2,...$

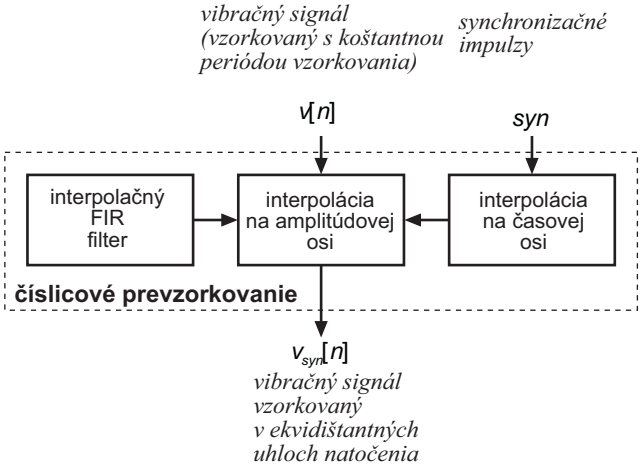
Situácia je v praxi naviac komplikovaná skutočnosťou, že spracovávaný vibračný signál (ktorý je typicky vzorkovaný v časovo ekvidištantných intervaloch napr. pomocou sigma-delta prevodníkov[36]) nie je stacionárny, čo je spôsobené meniacou sa rýchlosťou otáčania hriadeľov. V praxi je dokonca žiadúce analyzovať mechanický systém pri *zmene otáčok*, pretože práve v tomto režime je možné odhaliť mechanické chyby a mechanický systém je analyzovaný v zapojení znázornenom na obr. 6.11 [148], [178].

[36] Existuje aj možnosť využitia klasických AD prevodníkov, ktorých okamihy vzorkovania sú synchronizované s frekvenciou otáčania, takáto realizácia však vyžaduje relatívne zložité analógové riešenie s využitím preladiteľného analógového obmedzovacieho filtra (tzv. sledovacieho filtra).



Obr. 6.11 Princíp snímania vibračného signálu s konštantnou frekvenciou vzorkovania

S pomocou synchronizačných impulzov, ktoré snímajú uhol natočenia hriadeľa, je možné realizovať prevzorkovanie časovo nestacionárneho vibračného signálu na signál uhlovo stacionárny [178], pre ktorý vyššie uvedené úvahy ostávajú v platnosti. Prevzorkovanie signálu je možné realizovať pomocou adaptívne preladiteľného FIR filtra, ktorého štruktúra je znázornená na obr. 6.12 [148]. Úlohou FIR filtra je zároveň potlačenie spektrálnych zložiek nad maximálnou analyzovanou frekvenciou čím nahradzuje preladiteľný analógový obmedzovací (sledovací) filter. Realizácia tohto filtra predstavuje z pohľadu implementácie výpočtovo najnáročnejšiu operáciu, pričom pre ďalšie úvahy je najdôležitejšia skutočnosť, *že zložitosť jeho implementácie je úmerná počtu prevzorkovaných vzoriek*.



Obr. 6.12 Princíp využitia FIR filtra na prevzorkovanie vibračného signálu

6.1.5.1 ORTOGONÁLNA HARMONICKÁ SPEKTRÁLNA ANALÝZA

Ortogonálna harmonická spektrálna analýza (orthogonal order spectral analysis) umožňuje určiť DFT koeficienty signálu bez vzájomného ovplyvňovania (spectral leakage) medzi jednotlivými spektrálnymi zložkami. Táto metóda vyžaduje spracovanie segmentu (vibračného) signálu, ktorý obsahuje celočíselný počet otáčok jednotlivých hriadeľov.

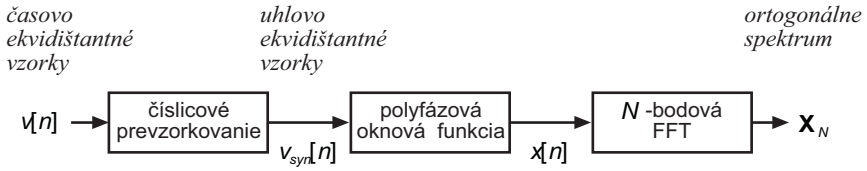
V prípade prevodovky [147], ktorá má $z_1 = 23$ a $z_2 = 44$ zubov (počty zubov nemajú spoločného celočíselného deliteľa) a požiadavke analyzovať jednotlivé harmonické až po tretiu *zubovú* (gear mesh) *frekvenciu* hriadeľa 1, je potrebné vypočítať spektrum po 69. harmonickú so *sub-harmonickým rozlíšením* 1/44. To vyžaduje určenie $44 \times 69 = 3036$ frekvenčných zložiek, čo vyžaduje DFT s rozmerom minimálne 6072. V prípade využitia

Hannovej oknovej funkcie $h[n]$ (ktorá ma z hľadiska spektrálnych vlastností pre túto aplikáciu veľmi výhodné vlastnosti) a analýze signálu

$$x[n] = h[n]v_{syn}[n] \tag{6.58}$$

pomocou FFT algoritmu s rozmerom, ktorý je mocninou dvoch, je potrebné použiť FFT s rozmerom 16384.

Štruktúra kompletného algoritmu ortogonálnej harmonickej analýzy je znázornená na obr. 6.13 [149], [150], [151].



Obr. 6.13 Ortogonálna harmonická analýza na báze FFT

6.1.5.2 HARMONICKÁ ANALÝZA S VYUŽITÍM DECIMOVANEJ FFT

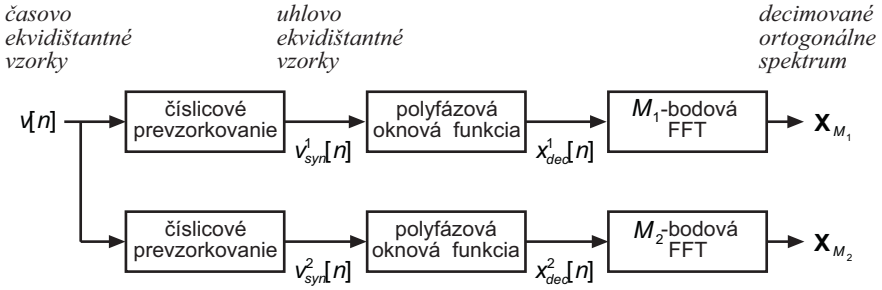
Z pohľadu praktickej implementácie je rozmer FFT (16384) vysoký a vyžaduje predovšetkým veľké pamäte. Rozmer použitej FFT je možné výrazne znížiť použitím tzv. *polyfázovej oknovej funkcie*, ktorá vychádza z modulo M redukcie signálu [152] a umožňuje počítať len požadované spektrálne zložky, t.j. harmonické jednotlivých hriadeľov s využítím FFT s podstatne menšími rozmermi. Metóda je založená na rozdelení signálu $x[n]$ (obr. 6.13) dĺžky $N = MB$ na B neprekrývajúcich sa blokov dĺžky M, sčítaní B blokov a vytvorení signálu

$$x_{dec}[n] = \sum_{i=0}^{B-1} x[iM + n], \qquad n = 0,1,\ldots,M-1 \tag{6.59}$$

Modulo M redukovaný signál $x_{dec}[n],\ n = 0,1,\ldots,M-1$ a pôvodný signál $x[n],\ n = 0,1,\ldots,N-1$ majú rovnakú M-bodovú DFT, t.j. platí

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi kn}{M}} = \sum_{n=0}^{M-1} x_{dec}[n]e^{-j\frac{2\pi kn}{M}} \qquad k = 0,1,\ldots,M-1 \tag{6.60}$$

V prípade, že sa využijú optimalizované FFT s rozmerom, ktorý je mocninou dvoch, je možné určiť harmonické zložky jednotlivých hriadeľov pomocou algoritmu, ktorého blokový diagram je znázornený na obr. 6.14 [151].

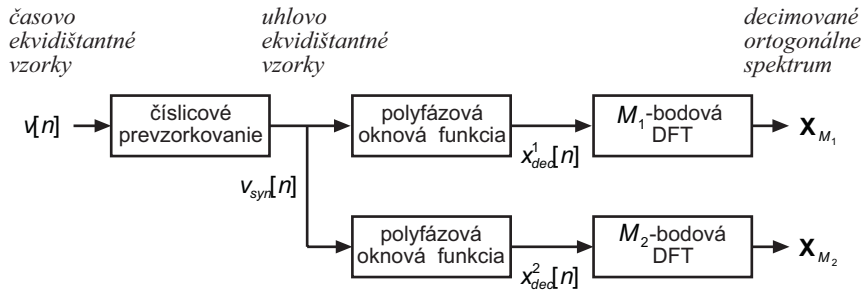


Obr. 6.14 Ortogonálna harmonická analýza na báze decimovanej FFT

Tento algoritmus výrazne redukuje rozmer potrebných algoritmov FFT (2×256 -bodová FFT), vyžaduje však dvojnásobné prevzorkovanie, pričom počet prevzorkovaných vzoriek $N_1 = 2 \times 44 \times 256 = 22528$ a $N_2 = 2 \times 23 \times 256 = 11776$ je väčší ako v prípade ortogonálnej harmonickej spektrálnej analýzy s využitím FFT ($N = 16384$).

6.1.5.3 HARMONICKÁ ANALÝZA S VYUŽITÍM DECIMOVANEJ DFT

Táto metóda umožňuje využiť DFT so všeobecným rozmerom, čo umožňuje zmeniť štruktúru celkového algoritmu, ktorého blokový diagram je znázornený na obr. 6.15 [151]. Algoritmus využíva len jedno prevzorkovanie, pričom počet prevzorkovaných vzoriek je $N = 23 \times 44 \times 3 \times 4 = 12144$, čo je výrazne menej ako v predchádzajúcej metóde. Tento počet je minimálny možný počet na zabezpečenie ortogonality jednotlivých spektrálnych zložiek a tak *nepriamo znižuje zložitosť algoritmu prevzorkovania*. Rozmer použitých DFT je oproti ortogonálnej harmonickej analýze taktiež výrazne redukovaný, čo sa prejavuje vo výrazne nižších pamäťových nárokoch algoritmu. Výpočet DFT je možné realizovať pomocou Chirp DFT algoritmu (6.47) – (6.55).



Obr. 6.15 Ortogonálna harmonická analýza na báze decimovanej DFT

6.1.5.4 SIMULAČNÉ VÝSLEDKY

Rýchlosť implementacie a vplyv zaokrúhľovacích chýb opísaného efektívneho algoritmu harmonickej analýzy bola testovaná na syntetickom signále $v_{syn}[n]$, pre frekvenciu otáčania 1. hriadeľa $f_{rot} = konšt$, $z_1 = 23$, $z_2 = 44$ a frekvenciu vzorkovania $f_s = 10 f_{rot} z_1$, pričom zložky tvoriace tento signál sú uvedené v tab. 6.5 [151].

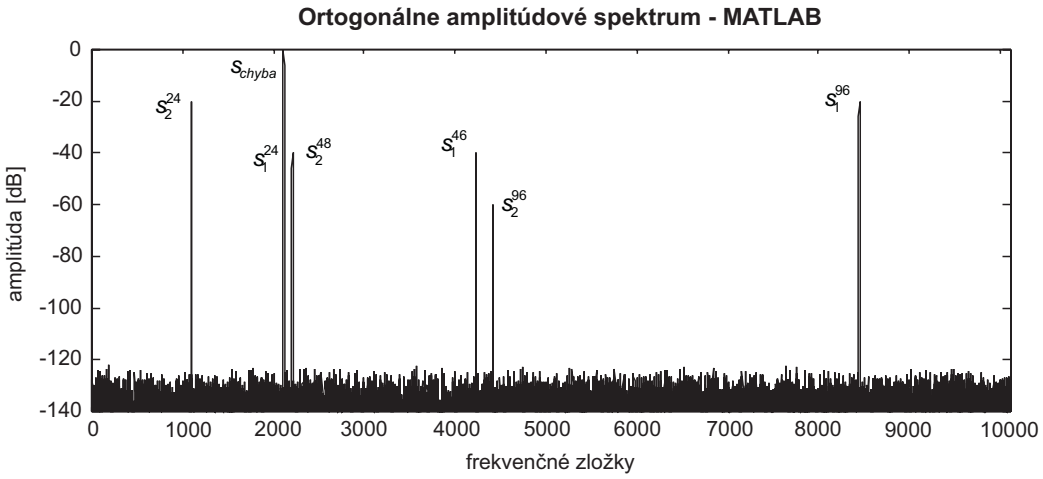
Testovací signál

$$v_{syn}[n] = s_1^{24}[n] + s_1^{48}[n] + s_1^{96}[n] + s_2^{24}[n] + s_2^{48}[n] + s_2^{96}[n] + s_{chyba}[n] \tag{6.61}$$

bol kvantovaný na 16 bitov (rozlíšenie typických sigma-delta prevodníkov), mapovaný do zlomkového formátu $\langle -1,1)$ a spracovaný ortogonálnou DFT s rozmerom $N = 20240$, pričom kompletné ortogonálne amplitúdové spektrum je zobrazené na obr. 6.16 [150], [151].

Tab. 6.5 Zložky testovacieho syntetického signálu v_{syn}

Zložka	Význam	Úroveň
$s_1^{24}[n] = 0{,}01\sin\left(\frac{2\pi 24}{z_1}\frac{n}{10}\right)$	24. harmonická hriadeľa 1	-40 dB
$s_1^{48}[n] = 0{,}1\sin\left(\frac{2\pi 48}{z_1}\frac{n}{10}\right)$	48. harmonická hriadeľa 1	-20 dB
$s_1^{96}[n] = 1{,}0\sin\left(\frac{2\pi 96}{z_1}\frac{n}{10}\right)$	96. harmonická hriadeľa 1	0 dB
$s_2^{24}[n] = 1{,}0\sin\left(\frac{2\pi 24}{z_2}\frac{n}{10}\right)$	24. harmonická hriadeľa 2	0 dB
$s_2^{48}[n] = 0{,}1\sin\left(\frac{2\pi 48}{z_2}\frac{n}{10}\right)$	48. harmonická hriadeľa 2	-20 dB
$s_2^{96}[n] = 0{,}01\sin\left(\frac{2\pi 96}{z_2}\frac{n}{10}\right)$	96. harmonická hriadeľa 2	-40 dB
$s_{chyba}[n] = 10\sin\left(\frac{2\pi(24+1/z_2)}{z_1}\frac{n}{10}\right)$	chybový signál, ktorý nie je súčasťou decimovaného ortogonálneho spektra	+20 dB



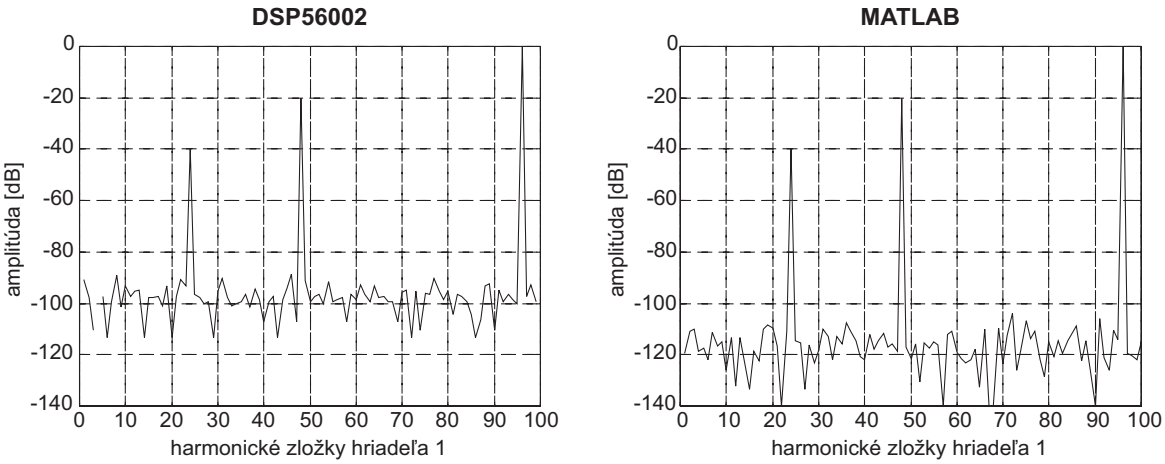
Obr. 6.16 Ortogonálne amplitúdové spektrum testovacieho signálu v_{syn} počítané 20240-bodovou DFT (64-bitová aritmetika v MATLAB-e)

Testovaný signál predstavuje signál so značným dynamickým rozsahom, pričom užitočný signál má hodnoty minimálne o 20 dB nižšiu ako je maximálny rozsah AD prevodníka.

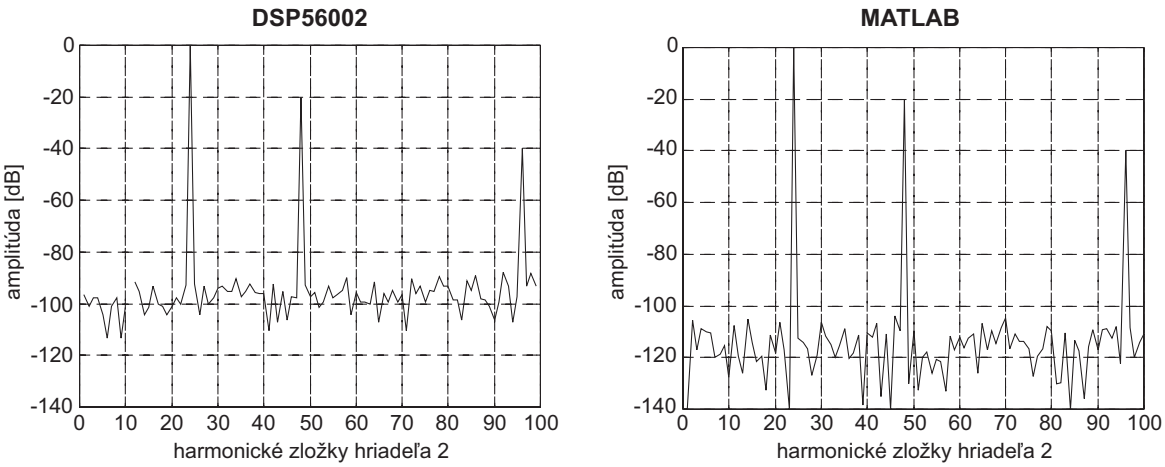
Decimované ortogonálne DFT spektrum bolo vypočítané pomocou optimalizovaného algoritmu na obr. 6.15 a Chirp DFT s parametrami $M_1 = 230$, $M_2 = 440$ a $K = 100$, pričom bola využitá Hannova oknová funkcia a algoritmy FFT s rozmermi $L_1 = 512$ a $L_2 = 1024$ z optimalizovanej knižnice *libfft* pre spektrálnu analýzu na procesore DSP5600x [137]. Decimované ortogonálne spektrá pre hriadeľ 1 a hriadeľ 2 spolu s referenčnými výsledkami vypočítanými v MATLAB-e sú zobrazené na obr. 6.17 a obr. 6.18 [150], [151].

Výsledky ukazujú, že úroveň zokrúhľovacích chýb 24-bitovej implementácie v pevnej rádovej čiarke je dostatočne nízka (aj keď vyššia ako v prípade 64-bitovej aritmetiky s pohyblivou rádovou čiarkou implementovanej v MATLAB-e). Nízka úroveň

zaokrúhľovacích chýb implementácie na DSP je dosiahnutá s použitím automatickej zmeny mierky v procese výpočtu FFT ako aj relatívne nízkym rozmerom použitých FFT, čo je priamym dôsledkom princípu decimovanej ortogonálnej spektrálnej analýzy.



Obr. 6.17 Decimované ortogonálne spektrum hriadeľa 1



Obr. 6.18 Decimované ortogonálne spektrum hriadeľa 2

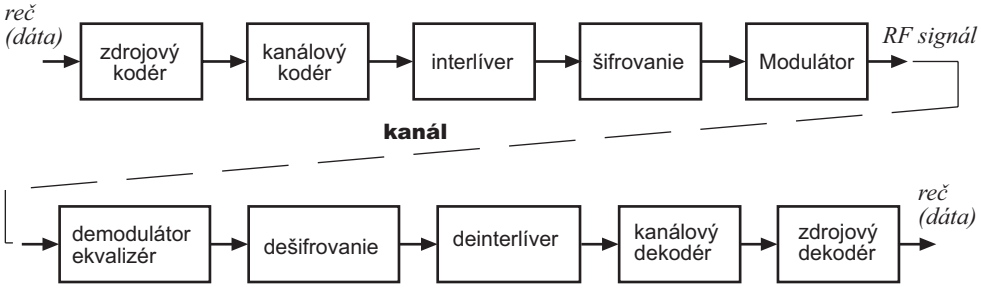
Implementácia kompletného algoritmu (bez algoritmu prevzorkovania) vyžaduje menej ako *5 % výpočtového výkonu* DSP56002/66 MHz a je príkladom kedy použitie principiálne zložitejšieho parciálneho algoritmu môže priniesť výrazné systémové výhody z hľadiská kompletného algoritmu. Zvyšná výpočtová kapacita je v reálnom systéme ANOVIS, pre ktorý bol tento algoritmus vyvinutý, využitá na číslicové prevzorkovanie signálu v reálnom čase pomocou interpolačného FIR filtra, ktorý v reálnej aplikácii spotrebuje viac ako 90 % výpočtového výkonu.

6.2 VYUŽITIE DSP KOPROCESOROV V SYSTÉME GSM

Digitálny mobilný systém pre rádiovú komunikáciu – GSM (Group Spécial Mobiles) [156] je typickým príkladom relatívne moderného telekomunikačného systému, ktorý široko využíva algoritmy ČSS. GSM využíva bunkový systém s viacnásobným prístupom na základe časového delenia kanálu (TDMA – Time Division Multiple Access) a digitálnu moduláciu GMSK (Gaussian Minimum Shift Keying) s konštantnou obálkou. Komunikácia je realizovaná pomocou paketov s presne definovanou štruktúrou. V ďalších častiach bude *naznačené využitie DSP koprocesorov* predovšetkým v linkovej vrstve systému GSM.

6.2.1 MODEL GSM KANÁLU

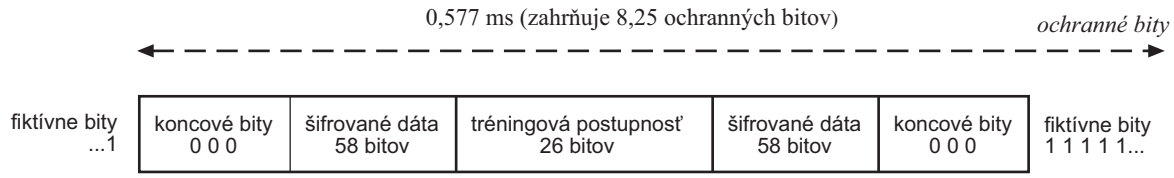
Základný model komunikačného kanálu GSM systému je znázornený na obr. 6.19 , pričom sú znázornené predovšetkým bloky súvisiace s linkovou vrstvou.



Obr. 6.19 Model GSM kanálu

GSM systém využíva z pohľadu šifrovania netradičné usporiadanie bloku šifrovania až za blokom kanálového kódovania (zabezpečenie konvolučným kódom). Toto usporiadanie je možné vzhľadom na *presnú synchronizáciu* a použitú *prúdovú šifru*, ktorá zabezpečuje správnu činnosť dešifrovania aj pri výskyte chybných bitov ktoré má opraviť kanálové dekódovanie.

V systéme GSM je definovaných niekoľko typov paketov, z ktorých najčastejšie využívaný a pre ďalší opis najdôležitejší je tzv. *normálny paket* (normal burst), ktorého štruktúra je znázornená na obr. 6.20 [158].



Obr. 6.20 Štruktúra normálneho paketu v systéme GSM

6.2.2 EKVALIZÁCIA KANÁLU

Ekvalizácia komunikačného kanálu patrí z pohľadu výpočtovej náročnosti telekomunikačných algoritmov medzi najzložitejšie algoritmy [154]. Úlohou ekvalizácie v systéme GSM je potlačenie vplyvu viaccestného šírenia signálu, ktorý typicky vzniká v mobilných rádiových sieťach. Dĺžka komunikačných paketov v systéme GSM je 0,577 ms a vzhľadom na použité frekvenčné pásmo (900 MHz) má GSM kanál charakter *kanálu s pomalým únikom* čo je zrejmé z doby za ktorú mobilná stanica prekoná polovicu vlnovej dĺžky (tzv. *koherenčný čas*) [153]

$$T_{koh} = \frac{\lambda / 2}{v} \tag{6.62}$$

pričom v je rýchlosť mobilnej stanice. Pre typický príklad rýchleho vlaku (v = 200 km/h) je $T_{koh} \approx 3$ ms, čo je niekoľkonásobne viac ako dĺžka komunikačného paketu. Vzhľadom na uvedené skutočnosti je možné považovať podmienky šírenia počas doby jedného paketu za kvázistacionárne.

Normálny paket obsahuje špeciálnu 26-bitovú *trénovaciu postupnosť* (je známa v prijímači aj vo vysielači), ktorá umožňuje v prijímači realizovať *odhad prenosového kanálu* a realizovať jeho ekvalizáciu. Efektívne algoritmy ekvalizácie GSM kanálu veľmi úzko súvisia s lineárnym modelom GMSK modulácie.

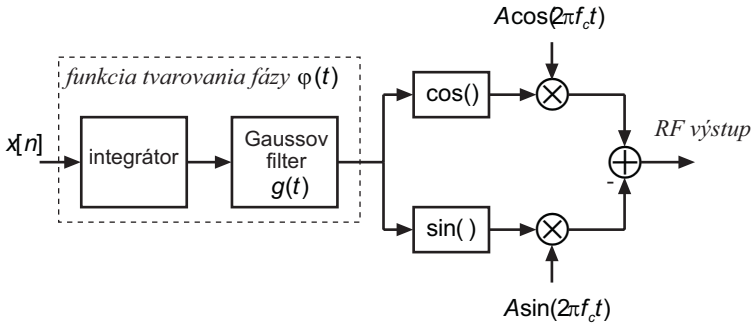
6.2.2.1 GMSK MODULÁCIA

GMSK modulácia je špeciálnym prípadom signálov so spojitou fázou (CPM – Continuous Phase Modulation) s modulačným indexom $h = 1/2$ a je opísaná v doporučení ETSI R.05.04 [158]. Binárny signál $x[n] = \pm 1$ modulovaný CPM modulátorom má nasledujúcu komplexnú reprezentáciu v základnom pásme

$$r_t(t) = Ae^{\left(j\pi h \sum_n x[n]\varphi(t-nT)+\varphi_0 \right)} \tag{6.63}$$

kde A je amplitúda, T je bitová perióda, φ_0 je počiatočná fáza a $\varphi(t)$ je funkcia fázového posuvu (phase shift function). CPM modulácie majú vo všeobecnosti veľmi dobré spektrálne vlastnosti (t.j. používajú úzke frekvenčné pásmo, čo je dané použitím riadenej medzisymbolovej interferencie) a sú vhodné (majú konštantnú obálku) pre nelineárne vysielače pracujúce v triede C.

Blokový diagram GMSK modulátora je zobrazený na obr. 6.21 [157].



Obr. 6.21 GMSK modulátor

Impulzova odpoveď Gaussovho filtra je určená vzťahom

$$g(t) = \frac{1}{2T}\left[Q\left(2\pi B \frac{t - \frac{T}{2}}{\sqrt{\ln 2}} \right) - Q\left(2\pi B \frac{t + \frac{T}{2}}{\sqrt{\ln 2}} \right) \right] \tag{6.65}$$

pričom $Q(t)$ je tzv. Q-funkcia

$$Q(t) = \int_{t}^{-\infty} \frac{1}{2\pi} e^{-\frac{r^2}{2}} dr \tag{6.66}$$

a funkcia fázového posunu je určená vzťahom

$$\varphi(t) = \int_{-\infty}^{t} g(t)dt \tag{6.67}$$

pričom GSMK modulácia v systéme GSM využíva hodnotu tzv. normalizovanej šírky (súčinu šírky pásma pre 3 dB pokles × dĺžka bitového intervalu) pásma $BT = 0{,}3$ – čo zodpovedá vplyvu jedného dátového bitu na časový interval približne troch bitových intervalov. Bitový interval je $T = 3{,}69\ \mu s$ a zodpovedá prenosovej rýchlosti 271 kbit/s.

6.2.2.2 LINEÁRNY MODEL GMSK SIGNÁLU

Funkcia fázového posunu $\varphi(t)$ pre GMSK moduláciu splňuje podmienku (po vhodnom posunutí na časovej osi a zanedbaní okrajových častí)

$$\varphi(t) = \begin{cases} 0 & t < 0 \\ konšt & t > (L-1)T \end{cases} \tag{6.68}$$

pričom $L = 4$. Aj keď GMSK modulácia je nelineárnou moduláciou, v práci [159] je ukázané, že ľubovoľný CPM signál s $h = 1/2$ je možné relatívne presne aproximovať ako súčet časovo a fázovo posunutých impulzov v tvare

$$r_t(t) \cong A\sum_n j^n a[n] q(t - nT) = A\sum_n A[n] q(t - nT) \tag{6.69}$$

pričom $A[n] = j^n a[n]$ sú *rotované binárne symboly* a súvislosť symbolov $a[n]$ a $x[n]$ je určená rekurzívnym vzťahom

$$a[n] = x[n]a[n-1] \tag{6.70}$$

Impulz $q(t)$ má dĺžku trvania menšiu alebo rovnú LT a je definovaný vzťahom [159]

$$q(t) = \prod_{i=0}^{L-2} \sin\left[\frac{\pi}{2} \psi(t + iT) \right] \tag{6.71}$$

pričom

$$\psi(t) = \begin{cases} \varphi(t) & t \le (L-1)T \\ 1 - \varphi(t - (L-1)T) & t > (L-1)T \end{cases} \tag{6.72}$$

Faktor j^n vo vzťahu (6.69) spôsobuje medzi jednotlivými symbolmi fázovú rotáciu vo fázovej rovine o uhol $\pi/2$, čo môže byť v prijímači odstránené tzv. *derotačnou technikou*, napr. násobením komplexnou postupnosťou j^{-n}. Za predpokladu, že modulovanou binárnou postupnosťou má byť postupnosť $a[n]$, vysielač musí v exponente vzťahu (6.63) použiť diferenčne kódovanú postupnosť [158], [159]

$$x[n] = a[n]a[n-1] \tag{6.73}$$

Základný význam uvedenej lineárnej aproximácie je v možnosti *využiť optimálne štruktúry* (prispôsobený filter) *a algoritmy* (Viterbiho algoritmus) pre lineárne modulácie [155] aj pre nelineárne modulované GMSK signály.

6.2.2.3 MODIFIKOVANÁ ŠTRUKTÚRA LINEÁRNEHO MLSE PRIJÍMAČA

Prijatý signál $r(t)$ po prechode disperzným prostredím s komplexnou impulzovou odpoveďou $c(t)$ má tvar

$$r(t) = A\sum_n A[n]h(t-nT) + šum(t) \tag{6.74}$$

pričom

$$h(t) = q(t) * c(t) \tag{6.75}$$

je celková komplexná impulzová odpoveď kanálu a $šum(t)$ je typicky modelovaný ako aditívny biely Gaussov šum (AWGN – Additive White Gaussian Noise). Po A/D prevode a filtrácii prispôsobeným filtrom je výstup $z[n]$ transverzálneho MF možné zapísať v tvare

$$z[n] = A\sum_k A[n-k]s[k] + šum[n] \tag{6.76}$$

pričom $s[k] = s(kT)$, $šum[n] = šum(nT)$ a

$$s(t) = h(t) * h^*(-t) = h(t) * h_{MF}(t) \tag{6.77}$$

je celková impulzová odpoveď kanálu a MF, pričom táto odpoveď je pre praktické kanály obmedzená na *K* predchádzajúcich a *K* nasledujúcich bitových intervalov

$$s[k] = 0 \qquad pre\ |k| > K \tag{6.78}$$

Za predpokladu, že prijímač môže byť modelovaný vzťahom (6.76), je možné využiť na potlačenie vplyvu kanálu VA s využitím Ungerboeckovej metriky [155]

$$J_n(\sigma[n]) = 2\,\mathrm{Re}(A^*[n]z[n]) + \max_{\{\sigma[n-1]\}\to\sigma[n]}\left\{J_n(\sigma[n-1]) - F(\sigma[n-1],\sigma[n])\right\} \tag{6.79}$$

pričom maximum je hľadané vo všetkých stavoch $\{\sigma[n-1]\}$ trellisu, ktoré vedú do stavu $\sigma[n]$, a tzv. tabuľka *F* je určená vzťahom

$$F(\sigma[n-1],\sigma[n]) = A^*[n]s[0]A[n] + 2\,\mathrm{Re}\left(A^*[n]\sum_{k=1}^{K} s[k]A[n-k]\right) \tag{6.80}$$

pričom rovnica (6.79) umožňuje počítať metriku rekurzívnym spôsobom. Pre GMSK moduláciu, ktorá je binárnou moduláciou, je možné vzťahy (6.79) a (6.80) zjednodušiť do tvaru [183] (vylúčením konštantného člena s $s[0]$ a vynechaním faktora 2)

$$J_n^{GMSK}(\sigma[n]) = a[n]\,\mathrm{Re}\left[(-j)^n z[n]\right] + \tag{6.81}$$

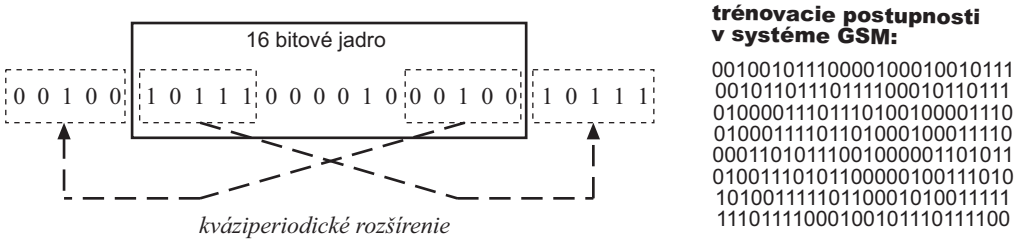
$$+ \max_{\{\sigma[n-1]\}\to\sigma[n]} \left\{ J_n^{GMSK}(\sigma[n-1]) - F_{GMSK}(\sigma[n-1],\sigma[n]) \right\}$$

$$F_{GMSK}(\sigma[n-1],\sigma[n]) = a[n]\sum_{k=1}^{K} \mathrm{Re}\left[j^{-k} s[k]\right]\ a[n-k] = a[n]\sum_{k=1}^{K} S[k]a[n-k] \tag{6.82}$$

pričom jedinou komplexnou operáciou vo vzťahoch (6.81) a (6.82) je derotácia výstupu prispôsobeného filtra a získanie reálnej hodnoty derotovanej veličiny. Ostatné veličiny, vrátane tabuľky $S[k],\quad k = 1, 2, \ldots, K$, sú reálne hodnoty, pričom tabuľka S môže byť pre známy kanál určená z koeficientov jeho impulzovej odpovede.

6.2.2.4 ODHAD KANÁLU S VYUŽITÍM TRÉNOVACEJ POSTUPNOSTI

Použite VA vyžaduje znalosť celkovej impulzovej odpovede prenosového kanálu $h(t)$ alebo aspoň jej odhadu $\tilde{h}(t)$. Tento odhad je možné realizovať pomocou 26-bitovej trénovacej postupnosti, ktorá je tvorená 16-bitovou postupnosťou, kváziperiodicky rozšírenou na obidvoch koncoch o 5 bitov s cieľom znížiť oscilácie autokorelačnej funkcie, čo je znázornené na obr. 6.22 .



Obr. 6.22 Princíp tvorby trénovacej postupnosti

V systéme GSM existuje 8 rôznych binárnych trénovacích postupností $\{tren[n]\}$ [160], ktoré boli nájdené počítačovou optimalizáciou a majú autokorelačnú funkciu (pre hodnoty $tren[n] = \pm 1$)

$$R[n] = \sum_{k=1}^{16} tren[k]tren[k+n] = \begin{cases} 16 & pre \quad n = 0 \\ 0 & pre \quad 0 < |n| \le 5 \end{cases} \tag{6.83}$$

pričom hodnota 5 súvisí s maximálnym oneskorením, pre ktoré bol systém GSM optimalizovaný. Z pohľadu lineárneho modelu GMSK signálov je možné trénovaciu postupnosť považovať za alternujúcu postupnosť ± 1 (pre párne symboly) a $\pm j$ (pre nepárne symboly).

Odhad prenosovej funkcie kanálu je možné realizovať pomocou vzťahu

$$h[k] = r[k] * (j^{-k} tren[-k]) \tag{6.84}$$

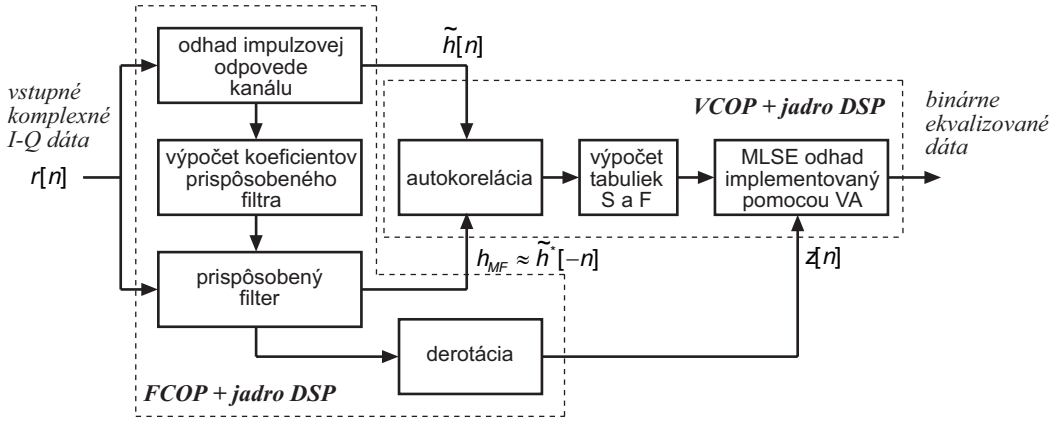
pričom $r[k] = r(kT)$ a druhý člen vzťahu (6.84) je prispôsobený (k trénovacej postupnosti) FIR filter s derotačnou kompenzáciou. V praktických implementáciách je odhad kanálu obmedzený na K-prvkový odhad, ktorý sa realizuje nájdením pozície K-prvkového pravouhlého okna s maximom energie [157] podľa vzťahu

$$\tilde{h}[n] = h[i_{opt} + n], \quad \left(E[i_{opt}] = \sum_{k=0}^{K-1} \left| h[i_{opt} + k] \right|^2 = \max \right) \Rightarrow i_{opt}, \quad n = 0,1,\ldots,K-1 \quad (6.85)$$

pričom hodnoty $E[i]$ sú určované pre všetky dostupné hodnoty i.

6.2.2.5 IMPLEMENTÁCIA S VYUŽITÍM FCOP A VCOP KOPROCESOROV DSP56305

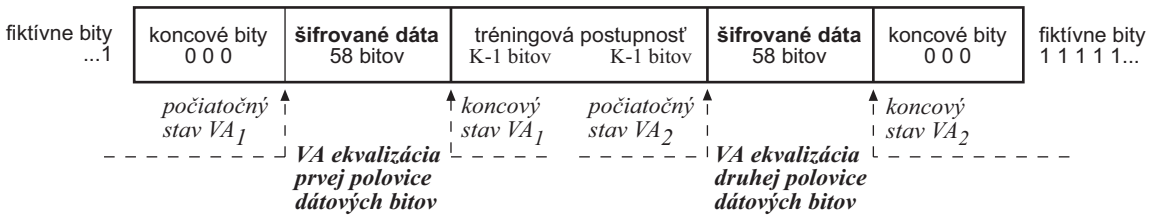
Bloková štruktúra výsledného algoritmu je zobrazená na obr. 6.23 [183]. Základ uvedeného algoritmu bol vytvorený pri analýze možností efektívnej implementácie GSM ekvalizéra na báze VA v simulačnom prostredí PTOLEMY [163] v rámci spolupráce s TU Ilmenau v Nemecku. Cieľom práce bolo navrhnúť implementáciu algoritmu, ktorá by sa blížila v praxi reálne implementovaným algoritmom s cieľom testovať vlastnosti celého GSM reťazca v prostredí Ptolemy. V dostupnej literatúre opísané algoritmy [157] boli z hľadiska implementácie pomocou DSP relatívne zložité a informácie o praktických DSP implementáciách neboli dostupné. Zverejnenie podrobných informácií o DSP56305 [56] v lete 1998 potvrdilo, že navrhnutá koncepcia algoritmu je vhodná pre implementáciu pomocou DSP.



Obr. 6.23 Štruktúra algoritmu ekvalizácie GSM kanálu

Naznačený algoritmus je možné efektívne implementovať pomocou filtračného a Viterbiho koprocesora DSP56305, ktorých základné možnosti boli naznačené na str. 34. Komplexnú koreláciu vstupnej komplexnej postupnosti (v tvare I a Q zložiek) s komplexnou rotovanou trénovacou postupnosťou (vzťah (6.84)) je možné realizovať pomocou FCOP v režime 3. Prispôsobený filter (vzťah (6.76)) je možné realizovať pomocou FCOP v režime 2. Operácie VA (vzťahy (6.81) a (6.82)) môžu byť realizované pomocou VCOP, pričom je však potrebné rozdeliť spracovanie paketu na dve polovice a oddelene spracovať prvú a druhú polovicu (zašifrovaných) dát[37]. S výhodou je možné využiť možnosť vnútiť počiatočný a koncový stav trellisu vo VCOP na základe známych koncových bitov čo je znázornené na obr. 6.24 [183].

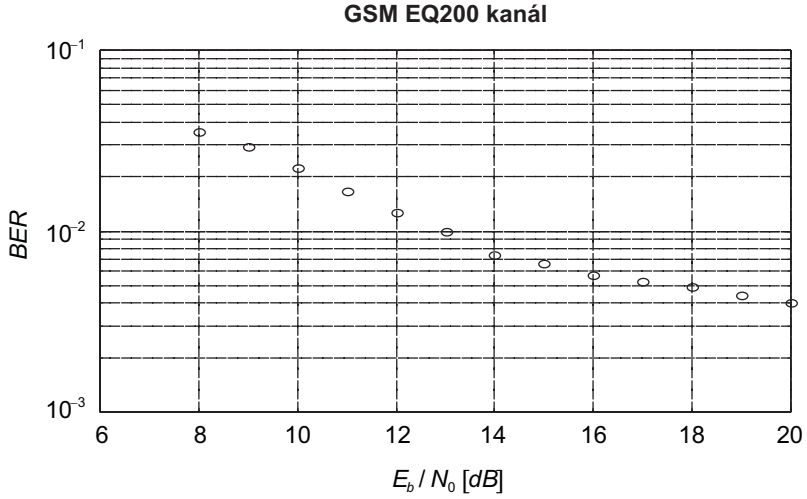
[37] V skutočnosti je zašifrovaných len $2 \times 57 = 114$ dátových bitov, dva bity sú využité na signalizáciu.



Obr. 6.24 Princíp oddeleného spracovania dátových bitov v normálnom pakete

Ostatné operácie je možné realizovať pomocou jadra DSP56305. S využitím kanálov DMA je možné realizovať spracovanie paketov pomocou súbežne pracujúcich FCOP, VCOP a jadra DSP, pričom simulačné výsledky ukazujú, že ekvalizácia jedného paketu trvá menej ako 7000 cyklov (80MHz verzia DSP56305 vykoná 80 miliónov cyklov/s).

Naznačený algoritmus bol implementovaný v simulátore DSP56305 ako aj v simulačnom prostredí MATLAB. Porovnanie výsledkov simulácie potvrdzuje, že 16-bitová implementácia nezhoršuje presnosť uvedeného algoritmu. Normovaná závislosť bitovej chybovosti (BER – Bit Error Rate) pre profil kanálu pre testovanie ekvalizérov, zahrňujúci 6 nezávislých ciest šírenia s Rayleighovým únikom a dopplerovským posunom pre prijímač s rýchlosťou 200 km/h (EQ200) [162] v závislosti na normovaných hodnotách E_b / N_0 pre AWGN šum so spektrálnou šumovou hustotou N_0 a energiou na bit E_b [161] je uvedená na obr. 6.25 [183].



Obr. 6.25 Závislosť bitovej chybovosti pre EQ200 kanál

Aj keď výrobcovia DSP typicky zverejňujú aplikačné príručky k rôznym typom algoritmov a aplikácií pre svoje DSP, autorovi práce nie je známa žiadna aplikačná príručka firmy Motorola opisujúca využitie uvedených koprocesorov v GSM aplikáciách a uvedená podrobnejšia analýza môže slúžiť aj na vyplnenie tejto medzery.

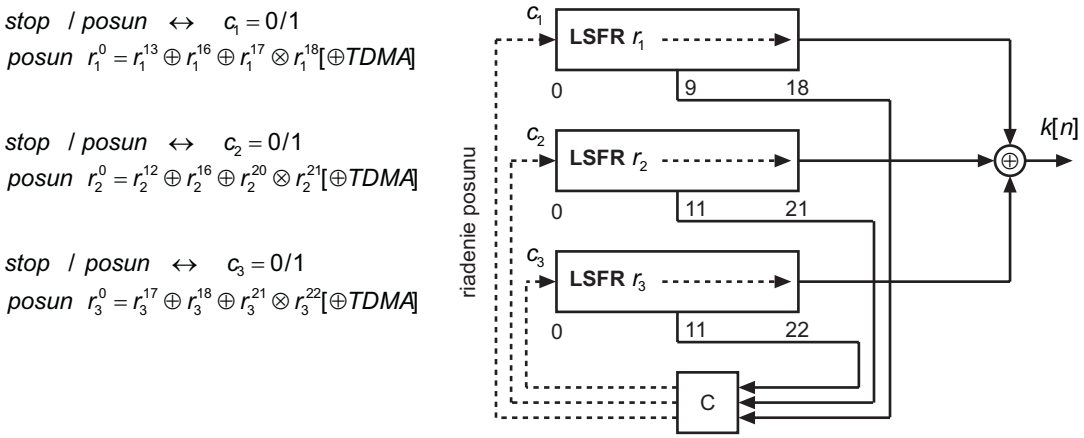
6.2.3 ŠIFROVANIE V SYSTÉME GSM

K ochrane rádiovej komunikácie v systéme GSM sa používajú kryptografické techniky, ktorých cieľom je sťaženie odpočúvania cudzích hovorov. Samotné šifrovanie je realizované len medzi mobilnou a bázovou stanicou. Ďalšia komunikácia medzi bázovou stanicou a zvyškom GSM siete je nešifrovaná. Samotný šifrovací algoritmus A5 (resp. jeho

varianty A5/1 a A5/2) je stále oficiálne utajovaný, na základe dohody o mlčanlivosti je poskytovaný výrobcom čipov a mobilných telefónov. Nepodpísanie (spôsobené administratívnou chybou) tejto dohody medzi Britskou telefónnou spoločnosťou a Dr. Simonom Stepherdom z Bradfordskej univerzity spôsobilo únik algoritmu na verejnosť. Aj keď články Dr. Stepherda o možných metódach kryptoanalýzy algoritmu A5 [164], [165] boli tajnou službou vyhlásené za neverejné, bolo už len otázkou času, kedy podobné metódy opísali iní.

6.2.3.1 ŠIFROVACÍ ALGORITMUS A5

Šifra A5 [168] je typ prúdovej šifry [167], pri ktorej sa pôvodná binárna postupnosť transformuje pomocou XOR operácie s pseudonáhodnou binárnou postupnosťou $k[n]$ generovanou algoritmom A5, ktorého štruktúra je znázornená na obr. 6.26 .



Obr. 6.26 Generovanie pseudonáhodnej postupnosti v algoritme A5

Štruktúra je zložená z *troch posuvných registrov* r_1 (19-bitový), r_2 (22-bitový) a r_3 (23-bitový) *s lineárnou spätnou väzbou*, pričom riadenie posunu jednotlivých posuvných registrov je realizované pomocou *nelineárnej trojvstupovej funkcie C*, ktorá má tri vstupy a tri výstupy. Výstup funkcie *C* určuje, či sa konkrétny register posunie ($c_i = 1$, „go“), alebo zostane stáť ($c_i = 0$, „stop“). Úlohou nelineárnej funkcie *C*, ktorej pravdivostná tabuľka pre systém GSM je v tab. 6.6 , je zvýšenie krytografickej bezpečnosti algoritmu. Z tabuľky je zrejmé, že v každom takte je vždy najviac jeden posuvný register neposunutý.

Tab. 6.6 Pravdivostná tabuľka funkcie C

Vstupy			Výstupy		
r_1 bit 10	r_2 bit 12	r_3 bit 12	c_1	c_2	c_3
0	0	0	1	1	1
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1

Počiatočné naplnenie je realizované 64-bitovou hodnotou (ktorá sa počas nadviazaného spojenia nemení) získanou zo SIM karty mobilného telefónu algoritmami A3 a A8 počas autentifikačnej fázy [166] a 22-bitovej identifikačnej hodnoty TDMA rámca, ktorá sa mení pre každý paket.

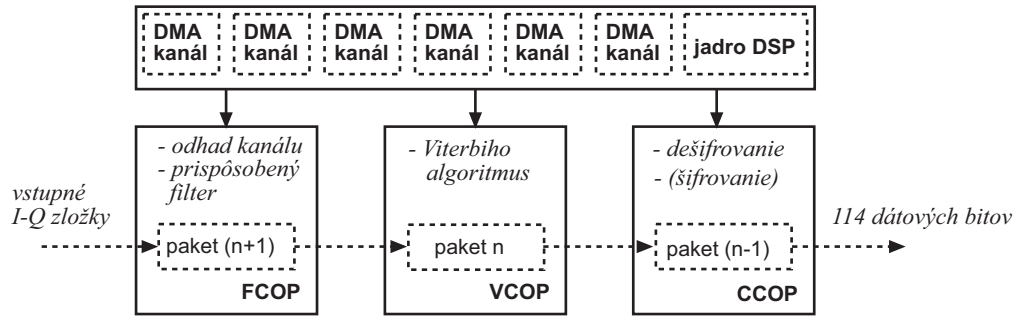
6.2.3.2 IMPLEMENTÁCIA S VYUŽITÍM CCOP KOPROCESORA DSP56305

CCOP poskytuje podporu pre všetky tri fázy algoritmu A5, ktoré sú tvorené vstupom bitov TDMA rámca do LSB bitov registrov r_1, r_2 a r_3 počas *incializačnej fázy*, *fázy nelineárneho zmiešavania* počas ktorej sa výstupné bity negenerujú a dochádza len k riadenému posunu posuvných registrov, tak aj vo *fáze generovania bitov šifry A5*. Tieto fázy sú podrobne opísané napr. v zdrojovom C kóde v [168]. FCOP umožňuje generovať jeden bit počas každého inštrukčného cyklu a na vygenerovanie 2×114 bitov kľúča (pre obidva smery komunikácie) z jedného TDMA rámca vyžaduje

$$22 \text{ (TDMA inicializácia)} + 2 \times (100 + 114) \text{ (premiešanie + generovanie)} = 450 \text{ cyk}$$

Pre 80MHz verziu DSP56305 to znamená, že vygenerovanie trvá menej ako 6 μs (aj vrátane nutnej inicializácie koprocesora). Aj keď jadro DSP56305 obsahuje vylepšenú podporu pre bitové operácie, je rýchlosť CCOP rádovo vyššia ako by bola implementácia len v jadre DSP. Dokonca aj v prípade možného súbežného spracovania GSM paketov, ktoré je principiálne znázornené na obr. 6.27 je CCOP najmenej vyťaženým koprocesorom.

Aj keď je CCOP optimalizovaný pre algoritmus A5, je CCOP podstatne univerzálnejší (obsahuje napr. 4 posuvné registre), plne reprogramovateľnú funkciu C a spätné väzby posuvných registrov [56]. Naviac CCOP umožňuje realizáciu aj ďalších algoritmov ako napr. generovanie CRC syndrómu, kódovanie a dekódovanie Fire kódov na zabezpečenie proti zhlukovým chybám, čo je možné využiť v ďalších fázach spracovania paketov.



Obr. 6.27 Využitie DSP56305 pri súbežnom spracovaní paketov

6.3 ZHRNUTIE

Uvedená kapitola obsahuje dve navzájom nesúvisiace aplikácie. Ich základným spoločným znakom je skutočnosť, že popri znalosti architektúry cieľového DSP (ktoré sú z hľadiska programovacieho modelu podstatne jednoduchšie ako napr. architektúry výkonných moderných superskalárnych procesorov) je extrémne dôležitá aj znalosť algoritmov ČSS. Táto znalosť v konečnom dôsledku rozhoduje o úspešnosti alebo neúspešnosti mapovania algoritmu ČSS do DSP. Voľba vhodného algoritmu z pohľadu cieľového DSP tak často umožňuje niekoľkonásobné zrýchlenie implementácie a popri relatívne nepatrných rozdieloch medzi jednotlivými DSP z tej istej kategórie je podstatne dôležitejším faktorom. Tieto skutočnosti sú zrejme hlavným dôvodom, prečo je programovanie DSP (aj napriek priblíženiu metód ich programovania k programovaniu univerzálnych jednočipových procesorov) stále doménou predovšetkým ľudí, ktorí majú solídne základy z teórie číslicového spracovania signálov.

7 ZÁVER

Dnes, s odstupom dvoch desaťročí od vzniku DSP, je možné konštatovať, že história vývoja jednočipových DSP je v niektorých aspektoch veľmi podobná histórii mikroprocesorov. Ich spoločným hlavným prínosom bola integrácia základných stavebných blokov do monolitického čipu. V prípade signálových procesorov to bola integrácia základných stavebných blokov procesora pre číslicové spracovanie signálov. Samotná integrácia umožnila zmenšiť cenu, rozmery a príkon zariadení využívajúcich metódy ČSS na úroveň, kedy bolo možné využiť programovateľný DSP aj v cenovo kritických aplikáciách ako náhradu špeciálnych zákazníckych obvodov. Najznámejším a pravdepodobne prvým prípadom bolo využitie TMS320C10 v hračke „Julie doll" od firmy Worlds of Wonder [29], keď TMS320C10 vytlačil dovtedy využívaný zákaznícky čip. DSP naviac v mnohých aplikáciách, predovšetkým pri náhrade analógových zariadení umožnili vytvoriť kvalitatívne nové riešenia, ktoré posunuli úroveň techniky o veľký krok vpred.

Na rozdiel od všeobecnej popularity univerzálnych mikroprocesorov a mikropočítačov, pričom predovšetkým popularita jednočipových mikropočítačov stále rastie a má pevné miesto už v osnovách mnohých stredných škôl, je práca s DSP považovaná za značne zložitú. Táto situácia má v podstate dve základné príčiny.

V dobe vzniku DSP bolo ich programovanie skutočne zložité a pripomínalo skôr programovanie mikroprocesorových rezov ako programovanie univerzálnych mikroprocesorov. Tento nedostatok DSP sa však počas niekoľkých ďalších rokov prakticky vytratil a ďalšie generácie DSP už mali spôsob programovania veľmi blízky k spôsobu programovania univerzálnych mikroprocesorov a dnes je ich programovací model dokonca jednoduchší ako programovací model vysokovýkonnych univerzálnych mikroprocesorov. Ba aj architektúry najjednoduchších jednočipových mikroprocesorov sú dnes často založené na harvardskej architektúre a ich používateľom to vôbec neprekáža.

Druhým a zrejme hlavným dôvodom pre ich malú popularitu je veľmi tesná súvislosť medzi DSP a teóriou ČSS. Používatelia bez solídnych základov z teórie ČSS typicky narazia na ťažko prekonateľné problémy už pri pokuse o implementáciu IIR filtra a radšej použijú FIR filter a o implementáciu IIR filtra sa už viackrát nepokúsia (to je realita overená v praxi). Samozrejme implementácia zložitejších algoritmov je potom ďaleko nad ich možnosťami. Je preto optimálne, pokiaľ snahe o zvládnutie DSP predchádza štúdium metód ČSS, prípadne ak tieto dve činnosti prebiehajú paralelne. Demonštrácia aj jednoduchého algoritmu ČSS v reálnom čase môže byť pri výučbe metód ČSS veľmi ilustratívna a dokonca často aj motivujúca [169].

Existujú aj odlišné názory, ktoré vychádzajú z tvrdenia, že DSP je len výkonný programovateľný jednočipový procesor a jeho využitie je teda len otázkou programovania. Cieľom opísaných algoritmov v kapitole 6 bolo ukázať, že tento pohľad na programovanie DSP je veľmi zjednodušujúci a vzdialený od reality.

V súčasnosti sú DSP predovšetkým prostriedkom na riešenie praktických problémov. Je napr. všeobecne známe, že niektoré nové telekomunikačné štandardy boli vyvíjané

paralelne s ich implementáciou pomocou DSP, a umožnili tak ich testovanie priamo v reálnych podmienkach.

V dnešnej dobe, kedy je stavba špecializovaných procesorov ČSS stále častejšie nahradzovaná univerzálnymi programovateľnými technickými prostriedkami, sú DSP často optimálnym prostriedkom na riešenie praktických problémov. Doterajšie skúsenosti autora utvrdzujú v názore, že až praktické overenie a nasadenie algoritmov ČSS (ktoré na papieri často fungujú bez problémov) je schopné potvrdiť vhodnosť a vlastnosti implementovaného algoritmu. Konfrontácia s praxou tak často môže byť dokonca hnacou silou ďalšieho zdokonaľovania algoritmov.

Predkladaná habilitačná práca je pohľadom na problematiku v súčasnosti veľmi dynamického rozvoja jednočipových signálových procesorov a zhrnutie výsledkov a poznatkov, ktoré autor získal v tejto oblasti počas svojho pôsobenia na Katedre elektroniky a multimediálnych telekomunikácií v Košiciach. Práca nadväzuje na výskumné úlohy grantového a inštitucionálneho výskumu „Číslicové metódy predspracovania a prenosu signálov“ (č. 41141), „Spracovanie signálov v digitálnych komunikačných systémoch“ (č. 1/284/92), „Inteligentné spracovanie signálov v telekomunikáciách“ (č. 2322/95) a „Spracovanie a prenos multimediálnych dát telekomunikačnými sieťami“ (č. 4118). Hlavným zdrojom poznatkov bola účasť na medzinárodnom projekte „Copernicus CIPA-CT94-0220 – Innovative Methods of Noise and Vibration Analysis on Reciprocating Machinery for the Purpose of Quality Control and Diagnostics,“ spolupráca s partnerskými univerzitami a firmami v zahraničí (Technical University of Ilmenau, Technical University of Budapest a Medav GmbH) v rámci tohto projektu, ako aj študijný pobyt na Politecnico di Torino v rámci projektu TELECOMNET – Telecommunication Networks and Services (Tempus Project JEP-09326-95). Výrazný prínos mala aj spolupráca s priemyslom na báze hospodárskej činnosti (VSE Košice, TTC Telecom, Medav GmbH), ktorá umožnila nasadenie a overenie niektorých nových metód ČSS v náročných priemyselných podmienkach.

V rámci pedagogického procesu, ktorý by mal na vysokých školách technického smeru sledovať najnovšie technické trendy a výsledky vedy, autor využíva výsledky vedeckej a výskumnej činnosti v oblasti signálových procesorov aj v niektorých predmetoch vyučovaných na Katedre elektroniky a multimediálnych telekomunikácii v Košiciach. Hlavná časť výsledkov je študentom odovzdávaná v rámci predmetu Signálové procesory v telekomunikáciách. Časť týchto výsledkov je premietnutá do výučby aj v predmetoch Kódovanie a modulácia a Aplikovaná kryptografia a aktuálne problémy z oblasti signálových procesorov sú riešené aj v rámci diplomových prác.

PRÍLOHY

Zoznam príloh:

POCKET GUIDE TO POPULAR DSP PROCESSORS AND CORES

Revised 3/17/99

Courtesy of BERKELEY DESIGN TECHNOLOGY, INC.

2107 Dwight Way, 2nd Floor, Berkeley CA 94704
Tel: (510) 665-1600
WWW: http://www.bdti.com
Fax: (510) 665-1680
E-mail: info@bdti.com

Currently Available Processors

Manufacturer	Family	Chip, Core, or Both	Arithmetic	Data Width	Instruction Width	Instruction Clock Speed [1]	Relative DSP Perf. BDTImarks(TM) [2,5]	Max. Address Space [3,5] Program	Max. Address Space [3,5] Data	Voltage(s)[5]	Unit Price [4,5] (Qty. 10,000)	Notes
Analog Devices	ADSP-21xx	Both	Fixed-point	16 bits	24 bits	75 MHz	19	16 K	16 K	2.5/3.3, 3.3, 5.0	$5-$79	Many family members w/ assorted peripherals
	ADSP-2106x	Chip	Floating-pt.	40 bits	48 bits	60 MHz	17	16 M	4 G	3.3, 5.0	$20-$381	Strong support for multiprocessor designs
ARM	Piccolo	Core	Fixed-point	16 bits	16/32 bits	70 MHz	14	4 G	4 G	3.0	CV	DSP co-processor core for use with ARM7
DSP Group	PineDSPCore	Core	Fixed-point	16 bits	16 bits	40 MHz	n/a	64 K	64 K	3.3, 5.0	CV	First mainstream DSP core
	OakDSPCore	Core	Fixed-point	16 bits	16 bits	80 MHz	21	64 K	64 K	3.3, 5.0	CV	Pine's successor, widely licensed
Hitachi	SH-DSP	Chip	Fixed-point	16 bits	16/32 bits	66 MHz	17	4 M	128 K	3.0	$25	Hybrid DSP/microcontroller based on SH-2
IBM	C54XDSP	Core	Fixed-point	16 bits	16 bits	66 MHz	n/a	n/a	n/a	n/a	CV	Clone of TI's TMS320C54x
Lucent Technologies	DSP16xx	Chip	Fixed-point	16 bits	16 bits	120 MHz	22	64 K	64 K	3.0, 3.3, 5.0	$5-$75	Flash memory versions available for prototyping
	DSP16xxx	Chip	Fixed-point	16 bits	16/32 bits	100 MHz	37	1 M	1 M	3.3	$58	Dual-MAC architecture
Mentor Graphics	M320C50	Core	Fixed-point	16 bits	16 bits	CV	n/a	64 K	64 K	5.0	CV	Provided as synthesizable HDL
Motorola	DSP560xx	Chip	Fixed-point	24 bits	24 bits	47.5 MHz	13	64 K	128 K	5.0	$8-$21	24-bit data word
	DSP563xx	Chip	Fixed-point	24 bits	24 bits	100 MHz	25	16 M	32 M	3.3	$20-$53	PCI bus, DMA, can run '560xx code unmodified
	DSP566xx	Chip	Fixed-point	16 bits	24 bits	70 MHz	17	64 K	128 K	1.8, 2.5, 3	$15-$60	563xx architecture w/ 16-bit data word
	DSP568xx	Chip	Fixed-point	16 bits	16 bits	35 MHz	9	64 K	64 K	3.0-3.3	$6-$10	Contains many microcontroller-like features
Siemens (Infineon)	TriCore	Both	Fixed-point	32 bits	16/32 bits	80 MHz [6]	n/a	4 G (unified)		2.5/3.3	CV	Superscalar hybrid DSP/microcontroller core
Texas Instruments	TMS320C1x	Chip	Fixed-point	16 bits	16 bits	8.8 MHz	n/a	4 K	256	3.3, 5.0	CV	First commercially successful DSP
	TMS320C2x	Chip	Fixed-point	16 bits	16 bits	12.5 MHz	n/a	64 K	64 K	5.0	CV	TI's second generation fixed-point DSP
	TMS320C2xx	Both	Fixed-point	16 bits	16 bits	40 MHz	7	64 K	64 K	5.0	$5-$16	Low-cost cross between 'C2x and 'C5x
	TMS320C27xx	Core	Fixed-point	16 bits	16 bits	50 MHz	8	4 M	4 M	3.3	CV	Hybrid microcontroller/DSP
	TMS320C3x	Chip	Floating-pt.	32 bits	32 bits	40 MHz	9	16 M	16 M	3.3, 5.0	$10-$180	Cost-competitive with fixed-pt DSPs
	TMS320C4x	Chip	Floating-pt.	32 bits	32 bits	30 MHz	7	4 G (unified)		5.0	$69-$177	Intended for multiprocessor applications
	TMS320C5x	Chip	Fixed-point	16 bits	16 bits	50 MHz	10	64 K	64 K	3.3, 5.0	$11-$35	Strongly enhanced TMS320C2x
	TMS320C54x	Both	Fixed-point	16 bits	16 bits	100 MHz	25	64 K	64 K	2.5/3.3, 3.3, 5.0	$20-$27	Many specialized instructions
	TMS320C62xx	Chip	Fixed-point	16 bits	32 bits	200 MHz	99	16 M	16 M	1.8/3.3, 2.5/3.3	$90-$121	8-way VLIW
	TMS320C67xx	Chip	Floating-pt.	32 bits	32 bits	167 MHz	65	16 M	16 M	1.8/3.3	$143	Floating-point version of 'C62xx
Zoran	ZR386xx	Chip	Fixed-point	20 bits	32 bits	40 MHz	12	1 M	1 M	3.3, 5.0	$10-15	20-bit data word
ZSP Corporation	ZSP164xx	Chip	Fixed-point	16 bits	16 bits	200 MHz	70	64 K	64 K	2.5/3.3	$50	4-way superscalar architecture

Forthcoming Processors [6]

Analog Devices	ADSP-2116x	Chip	Floating-pt.	40 bits	48 bits	100 MHz	n/a	4 G	4 G	2.5/3.3	$138	SIMD-enhanced ADSP-2106x
	TigerSHARC	Chip	Fixed/Float	32 bits	32 bits	250 MHz	n/a	n/a	n/a	n/a	CV	4-way VLIW with SIMD capabilities
DSP Group	TeakDSPCore	Core	Fixed-point	16 bits	16 bits	130 MHz	n/a	n/a	n/a	n/a	CV	Dual MAC units, synthesizable core
	PalmDSPCore	Core	Fixed-point	16/20/24 bits	16/32 bits	150 MHz	n/a	1 M	n/a	n/a	CV	Selectable data width, dual MAC, synthesizable
Siemens (Infineon)	Carmel	Both	Fixed-point	16 bits	24/48 bits	120 MHz	n/a	8 M	64 K	2.5	CV	6-way VLIW, configurable instructions
StarCore	140	Core	Fixed-point	16 bits	16 bits	300 MHz	n/a	n/a	n/a	1.5	CV	6-way VLIW

NOTES:
[1] Instruction clock speed for fastest member of family. Most processors issue one instruction per clock cycle; VLIW and superscalar processors may issue multiple instructions.
[2] The BDTImark is a summary measure of DSP speed; higher is faster. See www.bdti.com for additional BDTImark scores.
[3] In native words. Some processors share address space between program and data memories.
[4] Unit prices quoted by manufacturers as of October, 1998.
[5] CV = Contact vendor; n/a = information not available
[6] Data is projected; processor has not yet been fabricated or is not yet available at speed shown.

ZOZNAM POUŽITEJ LITERATÚRY

[1] Wong, H.S.P. – Frank, D.J. – Solomon, P.M. – Wann, C.H.J. – Welser, J.J.: Nanoscale CMOS. *Proceedings of the IEEE*, Vol.87, No.4, April 1999, pp.537–569.

[2] Sarmiento, R. – Carballo, P.P. – Nuñéz, A.: High speed primitives of hardware accelerators for DSP in GaAs technology. *IEE Proceedings - G*, Vol.139, No.2, April 1992, pp.205-216.

[3] Deka, R.: A comprehensive study of digital signal processing devices. *Microprocessors and Microsystems*, Vol.19, No.4, May 1995, pp.201-209.

[4] Madisetti, V.K.: *VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Design Synthesis*. IEEE Press, Butterworth-Heineman, Boston 1995.

[5] Kalliojrvi, K.: Finite Word Length Effects in Floating-Point and Block-Floating-Point Digital Signal Processing Systems. PhD Thesis, Tampere University of Technology, Tampere, 1995.

[6] White, S.A.: Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review. *IEEE ASSP Magazine*, July 1989, pp.4-19.

[7] Marchevský, S. – Chmúrny, J.: Rýchly dvojrozmerný konvolútor na báze zvyškových číselných systémov. *Slaboproudy obzor*, 45, č.11, 1984, s.564-569.

[8] Adámek, J.: *Foundations of Coding: Theory and Applications of Error-Correcting Codes with an Introduction to Cryptography and Information Theory*. John Wiley & Sons, New York, 1991.

[9] Pollard, J.M.: The Fast Fourier Transform in a Finite Field. *Mathematics of Computations*, Vol.25, No.114, April 1971, pp.365-374.

[10] Reed, S.I. – Truong, T.K.: The Use of Finite Fields to Compute Convolutions. *IEEE Transactions on Information Theory*, Vol. IT-21, No.2, March 1975, pp.208-213.

[11] Taur, Y. – Buchanan, D. – Chen, W. – Frank, D.J. – Ismail, K.E. – Lo, S.H. – Sai-Halasz, G.A. – Viswanathan, R.G. – Wan, H.J.C. – Wind, J.S. – Wong, H.S.: CMOS scaling into the Nanometer Regime. *Proceedings of the IEEE*, Vol.85, No.4, April 1997, pp.486-502.

[12] Asai, S. – Wada, Y.: Technology Challenges for Integration Near and Below 0.1 μm. *Proceedings of the IEEE*, Vol.85, No.4, April 1997, pp.505-519.

[13] Davari, B. – Dennard, R.H. – Shahidi, G.G.: CMOS Scaling for High Performance and Low Power – The Next Ten Years. *Proceedings of the IEEE*, Vol.83, No.4, April 1995, pp.595-606.

[14] Semiconductor Industry Association (SIA): The National Technology Roadmap for Semiconductors [online]. Available: www.sematech.org/public.

[15] Wieder, W.A. – Neppl, F.: CMOS Technology Trends and Economics. *IEEE Micro*, August 1992, pp.10-19.

[16] Nishihara, A. – Magyar, A.: Implementation of DSP algorithms. *Elektrotechnický časopis*, Vol.36, No.6, June 1985, pp.449-463.

[17] Stevens, J.: DSPs in Communications. *IEEE Spectrum*, September 1998, pp.39-46.

[18] Westall, F.A. – Ip, S.F.A.: Digital Signal Processing in Telecommunications. Reprint from *BT Technol. J*., Vol.10, No.1, January 1992, pp.1-19.

[19] Kostic, Z. – Seetharaman, S.: Digital Signal Processors in Cellular Radio Communications. *IEEE Communications Magazine*, December 1997, pp.22-35.

[20] Page, J.H – Smee, D.A. – Pauley, D. – Hughes, P.J. – Williams, R.G.C.: An application of DSP to voiceband modems. Reprint from *BT Technol. J*., Vol.10, No.1, January 1992, pp.1-21.

[21] Chen, T. – Katsaggelos, A. – Kung, S.Y.: The Past, Present, and Future of Multimedia Signal Processing. *IEEE Signal Processing Magazine*, July 1997, pp.28-51.

[22] Tokhi, M.O – Hosain, M.A.: CISC, RISC and DSP processors in real-time signal processing and control. *Microprocessors and Microsystems*, Vol.19, No.5, June 1995, pp.291-299.

[23] Lapsley, P. – Bier, J. – Shoham, A. – Lee, E.A.: *DSP Processor Fundamentals: Architectures and Features*. IEEE Press, New York, 1997.

[24] Stewart, R.W. – Pfann, E.: Oversampling and sigma-delta strategies for data conversion. *Electronics & Communication Engineering Journal*, February 1998, pp.37-47.

[25] Jantzi, S.A. – Snelgrove, W.M. – Ferguson, P.F.: A Fourth-Order Bandpass Sigma-Delta Modulator. *IEEE Journal of Solid State Circuits*, Vol.28, No.3, March 1993, pp.282-291.

[26] Bazarjani, S. **–** Snelgrove, W.M.: A 160-MHz Fourth-Order Double-Sampled SC Bandpass Sigma-Delta Modulator. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol.45, No.5, May 1998, pp.547-555.

[27] Norsworthy, S.R. – Schreier, R. – Temes, G.C.: *Delta – Sigma Data Converters: Theory, Design, and Simulation*. IEEE Press, New York, 1997.

[28] Tuttlebee, W.H.W: Software Radio Technology: A European Perspective. *IEEE Communication Magazine*, February 1999, pp.118-123.

[29] Rabaey, J.M. – Broedersen, R. – Gass, W. – Nishitani, T.: VLSI Design and Implementation Fuels the Signal Processing Revolution. *IEEE Signal Processing Magazine*, January 1998, pp.22-37.

[30] Nishitani, T. – Maruta, R. – Kawakami, Y. – Goto, H.: A Single-Chip Digital Signal Processor for Telecommunications. *IEEE Journal of Solid-State Circuits*, SC-16, 1981, pp.372-376.

[31] Hagiwara, Y. – Kita, Y. – Miyamoto, T. – Toba, Y. – Hara, H. – Akazawa, T.: A Single Chip Digital Signal Processor and its Applications to Real-Time Speech Analysis. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.31, 1983.

[32] Magyar, A.: *Číslicové Zpracování Signálu II: Signálové procesory a jejich použití*. Skriptum TU v Košiciach, Košice 1991.

[33] Skalický, P.: *Digitální filtrace a signálové procesory*. Skriptum ČVUT, 1995.

[34] Frantz, G.A. – Lin, K.S. – Reimer, J.B. – Bradley, J.: The Texas Instruments TMS320C25 Digital Signal Microcomputer. *IEEE Micro*, December 1986, pp.10-28.

[35] Eichen, B.: NEC's μPD77230 Digital Signal Processor. *IEEE Micro*, December 1986, pp.60-69.

[36] Papamichalis, P. – Simar, R.*:* The TMS320C30 Floating-Point Digital Signal Processor. *IEEE Micro*, December 1988, pp.13-19.

[37] Sohie, G.R.L. – Kloker, K.L.: A Digital Signal Processor with IEEE Floating-Point Arithmetic. *IEEE Micro*, December 1988, pp.49-67.

[38] Simar, R. – Koeppen, P. – Leach, J. – Marshall, S. – Francis, D. – Mekras, G. – Rosenstrauch, J. – Anderson, S.: Floating-Point Processors Join Forces in Parallel Processing Architectures. *IEEE Micro*, August 1992, pp.60-69.

[39] Gluth, R.: Integrierte Signalprozessoren: Architekturen und technishe Realisierung. *Elektronik 18*, 5.9.1986, pp.112-125.

[40] Levy, M.: EDN's 1997 DSP – Architecture Directory. *EDN*, May 8, 1997, pp.43-107.

[41] Hlavička, J.: *Architektura počitačů*. Skriptum ČVUT, Praha 1996.

[42] Jelšina, M.: *Architektúry počítačov*. Skriptum TU v Košiciach, Košice 1992.

[43] Lee, E.A. – Messerschmitt, D.G.: Pipeline Interleaved Programmable DSP's: Architecture. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.35, No.9, September 1987, pp.1320-1333.

[44] Lee, E.A. – Messerschmitt, D.G.: Pipeline Interleaved Programmable DSP's: Synchronous Data Flow Programming. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.35, No.9, September 1987, pp.1334-1345.

[45] Marven, C. – Ewers, G.: *A simple approach to DIGITAL SIGNAL PROCESSING*. Texas Instruments, 1993.

[46] Lee, E.A.: Programmable DSP Architectures: Part I. *IEEE ASSP Magazine*, October 1988, pp.4-19.

[47] Lee, E.A.: Programmable DSP Architectures: Part II. *IEEE ASSP Magazine*, January 1989, pp.4-14.

[48] Blalock, G.: Microprocessors Outperform DSP 2:1, Unpredictable Execution, Poor Tools Complicate Use in Real-Time Applications. *Microprocessor Report*, Vol.10, No.17, December 30, 1996, pp.1-4 (article reprint).

[49] Schlett, M.: SH3-DSP: Take the Next Step in Advanced Embedded Computing. Proceedinngs of DSP Deutschland'98, Munchen, October 98, pp.45-52.

[50] DSP1609/ Flash DSP 1609F Digital Signal Processor. Preliminary Data Sheet, Lucent Technologies, Inc., February 1998.

[51] Geppert, L.: High-Flying DSP architectures. *IEEE Spectrum*, November 1998, pp.53-56.

[52] Lee, W. – Landman, P.E. – Barton, B. – Abiko, S. – Takahashi, H. – Mizuno, H. – Muramatsu, S. – Tashiro, K, - Fusumada, M. – Pham, L. – Boutaud, F. – Ego, E. – Gallo, G. – Tran, H. – Lemonds, C. – Shih, A. – Nandakumar, M. – Eklund, R.H. – Chen, I.C.: A 1-V Programmable DSP for Wireless Communications. *IEEE Journal of Solid State Circuits*, Vol.32, No.11, November 1997, pp.1766-1776.

[53] Mutoh, S. – Shigematsu, S. – Matsuya, Y. – Fukuda, H. – Kaneko, T. – Yamada, Y.: A 1-V Multithreshold-Voltage CMOS Digital Signal Processor for Mobile Phone Application. *IEEE Journal of Solid State Circuits*, Vol.31, No.11, November 1996, pp.1795-1802.

[54] Razavi, B.: Recent Advances in RF Integrated Circuits. *IEEE Communications Magazine*, December 1997, pp.36-43.

[55] Dipert, B.: FRAM: ready to ditch niche? *EDN*, April 10, 1997, pp.93-108.

[56] DSP56305, 24-Bit Digital Signal Processor User's Manual. DSP56305UM/AD, Motorola Inc., 1998.

[57] DSP56307, 24-Bit Digital Signal Processor User's Manual. DSP56307UM/AD, Motorola Inc., 1998.

[58] Maunder, C.M. – Tulloss, R.: Testability on TAP. *Microprocessors and Microsystems*, Vol.17, No.5, 1993, pp.260-275.

[59] Scannell, B.: Quad – SHARC DSP in Ceramic Quad Flatpack: Smaller. Faster, Cheaper AD14060, A 480 MFLOPS DSP Powerhouse. *Analog Dialogue*, Vol.31, No.2, 1997, pp.10.

[60] Viterbi, A.J.: Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, Vol.13, No.2, April 1967, pp.260-269.

[61] Lee, L.H.Ch.: *Convolutional Coding: Fundamentals and Applications*. Artech House, Boston, 1997.

[62] Taipale, D.: Implementing Viterbi Decoders Using the VSL Instruction on DSP Families DSP56300 and DSP56600. APR40/D, Motorola Inc., 1998.

[63] TMS320C54x, TMS320LC54x, TMS320VC54x Fixed Point Digital Signal Processors. SPRS039, Texas Instruments, Inc., February 1996.

[64] TMS320C54x User's Guide. Digital Signal Processing Products, Texas Instruments, Inc., 1995.

[65] Bursky, D.: Higher-Throughput DSP Chips Take on Complex Applications. *Electronic Design*, May 25, 1998, pp.80-88.

[66] Bier, J.: DSP16xxx Targets Communications Apps, New Lucent Design Extends Conventional Techniques to Improve Performance. *Microprocessor Report*, Vol.11, No.12, September 15, 1997, pp.1-6 (article reprint).

[67] Gao, F.: ADSL and V90 Modem: Standards, Principles, and Integration. Proceedings of DSP Deutschland'98, Munchen, October 98, pp.214-222.

[68] Fettweis, G. – Weiss, M. – Drescher, W. – Walther, U. – Engel, F. – Kobayashi, S. – Richter, T.: Breaking New Grounds over 3000 M MAC/s: A BROADBAND MOBILE MULTIMEDIA MODEM DSP. Proceedings of DSP Deutschland'98, Munchen, October 98, pp.30-34.

[69] Naccache, D. – Raihi, D.M.: Cryptographic Smart Cards. *IEEE Micro*, June 1996, pp.14-23.

[70] Faraboschi, P. – Desoli, G. – Fisher, J.A.: The Latest Word in Digital and Media Processing. *IEEE Signal Processing Magazine*, March 1998, pp.59-85.

[71] Sweeney, J.P.: Leverage Superscalar DSPs In Digital Communications Systems. *Electronic Design*, November 16, 1998, pp.49-62.

[72] Rathman, S. – Slavenburg, G.: Processing the New World of Interactive Media: The Trimedia VLIV CPU Architecture. *IEEE Signal Processing Magazine*, March 1998, pp.108-117.

[73] Pureell, S.: The Impact of Mpact-2. *IEEE Signal Processing Magazine*, March 1998, pp.102-107.

[74] Konstantinides, K.: VLIW Architectures for Media Processing. *IEEE Signal Processing Magazine*, March 1998, pp.16-19.

[75] Kočiš, I. – Šulko, I.: *Mikroprocesory a Mikropočítače*. ALFA Bratislava, 1986.

[76] Levický, D.: Analýza vlastností procesorov typu RISC pre číslicové spracovanie signálov. *Slaboproudý obzor*, 52 (1991), čís.2, s.50-51.

[77] Stallings, W.: Reduced Instruction Set Computer Architecture. *Proceedings of the IEEE*, Vol.76, No.1, January 1988, pp.38-55.

[78] Levy, M.: Microprocessor and DSP Technologies unite for Embedded Applications. *EDN*, March 2, 1998, pp.73-82.

[79] Seshan, N.: High VelocitTI Processing. *IEEE Signal Processing Magazine*, March 1998, pp.86-101.

[80] Wolf, O. – Bier, J.: StarCore Launches First Architecture, Lucent and Motorola Disclose New VLIW-Based Approach. *Microprocessor Report*, Vol.12, No.14, October 26, 1998, pp.1-4 (article reprint).

[81] THE STAR*CORE SC140 DSP CORE, Background Information. Motorola Inc., April 1999, www.mot.com, pp.1-14.

[82] Eyre, J. – Bier, J.: Carmel Enables Customizable DSP, User-Defined Instructions, Licensing Plan Set New Siemens Core Apart. *Microprocessor Report*, Vol.12, No.17, December 28, 1998, pp.1-6 (article reprint).

[83] Wolf, O. – Bier, J.: TigerSHARC Sinks Teeth Into VLIW, Analog Devices' High-End DSP Challenges Texas Instruments, StarCore. *Microprocessor Report*, Vol.12, No.16, December 7, 1998, pp.1-4 (article reprint).

[84] Main, I.: Factors to Consider when Choosing the Right DSP for the Job. *Electronic Design*, June 8, 1998, pp.67-72.

[85] Wright, M.: Media Processors target DIGITAL-VIDEO Roles. *EDN*, September 1, 1998, pp.59-75.

[86] Kung, S.Y.: *VLSI Array Processors*. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[87] Mikloško, J. – Klette, R. – Vajteršic, M. – Vrťo, I.: *Rýchle algoritmy a ich realizácia na špecializovaných počítačoch*. VEDA, Bratislava 1985.

[88] Drutarovský, M.: Systolická realizácia nelineárneho adaptívneho Volterrovho filtra. Zborník z celoštátneho seminára „Nové smery v spracovaní signálov". Račkova dolina, máj 1990, s.53-56.

[89] Kocur, D. – Drutarovský, M.: On Application of Recursive Modified Gram-Schmidt Algorithm For Adaptation of Adaptive Nonlinear Volterra Digital Filters. Proceedings of the 6th Scientific Conference of EF TU in Košice. September 1992, pp.174-179.

[90] Smith, M.J.S.: *Application – Specific Integrated Circuits*. Addison Wesley Longman, Berkeley, California, 1998.

[91] Mintzer, L.: FIR Filters with Field-Programmable Gate Arrays. *Journal of VLSI Signal Processing*, 6 (1993), pp. 119-127.

[92] Dick, C. – Harris, F.: Virtual signal processors. *Microprocessors and Microsystems*, 22 (1999), pp.135-148.

[93] Drutarovský, M.: Digital Signal Processing based on Field Programmable Logic Devices. Proceedings of the 3rd Tempus Telecomnet International workshop, Košice, September 1997, pp.91-94.

[94] Guštin, V.: An FPGA extension to ALU functions. *Microprocessors and Microsystems*, 22 (1999), pp.501-508.

[95] Drutarovský, M. – Galajda, P.: Implementation of DSP Functions in ALTERA Field Programmable Logic Devices. Proceedings of the 4th international conference DSP'99, Herľany, September 1999, pp.170-173.

[96] Deodhar, R.: Optimizing Compiler Technology Streamlines Complex Systems. *Electronic Design*, May 1, 1997, pp.153-160.

[97] Hwu, W.W. – Hank, R.E. – Gallagher, D.M. – Mahlke, S.A. – Levery, D.M. – Haab, G.E. – Gyllenhaal, J.C. – August, D.I.: Compiler Technology for Future Microprocessors. *Proceedings of the IEEE*, Vol.83, No.12, December 1995, pp.1625-1639.

[98] Papamichalis, P.: Digital Signal Processing Applications with the TMS320 Family, Volumes 1,2,3. Texas Instruments, Inc., 1990.

[99] Mar, A.: *Digital Signal Processing Applications Using the ADSP-2100 Family*, Volume 1. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[100] Babst, J.: *Digital Signal Processing Applications Using the ADSP-2100 Family*, Volumes 2. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[101] Ingle, K.V. – Proakis, J.G.: *Digital Signal Processing Laboratory Using the ADSP-2101 Microcomputer*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[102] El-Sharkawy, M.: *Real Time Digital Signal Processing Applications With Motorola's DSP56000 Family*. Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[103] Ombres, D.: C and C++ extensions simplify fixed-point DSP programming. *EDN*, October 10, 1996, pp.135-138.

[104] Levy, M.: C compilers for DSPs flex their muscles. *EDN*, June 5, 1997, pp.93-103.

[105] *Numerical C Extensions,* chapter 6 in ADSP-21000 Family C Tools Manual. Analog Devices, Inc., 1996.

[106] DSP56KCC, Motorola DSP56000 Family Optimizing C Compiler User's Manual. Motorola Inc., 1992.

[107] DSP563CCC, Motorola DSP56300 Family Optimizing C Compiler User's Manual. Motorola Inc., 1997.

[108] Čarnogurský, V.: Konvolučné metódy filtrácie na signálových procesoroch Analog Devices ADSP2181. Diplomová práca, KEMT FEI TU v Košiciach, Košice, apríl 1999.

[109] Kocur, D. – Drutarovský, M. – Galajda, P. – Marchevský, S. – Matúš, E. – Stanko, R.: Innovative Methods of Noise and Vibration Analysis on Reciprocating Machinery for the Purpose of Quality Control and Diagnostics. Proceedings of the 1st International Scientific Conference of the Faculty of Electrical Engineering and Informatics, TU Košice, February 1998, pp.39 - 40.

[110] Drutarovský, M.: ANOVIS FSK Demodulator: Design of Optimal Filters and Algorithm Parameters. Výskúmná správa projektu VHČ pre MEDAV GmbH, Košice, február 1999, s.1-9.

[111] Drutarovský, M.: Demodulator of FM and AM Signals – CORDIC DSP Implementation. Proceedings of the 4th international conference DSP'99, Herľany, September 1999, pp.166-169.

[112] Drutarovský, M.: C-Programming for DSP56001 Part of MINISYS. Proceedings of the 3rd Copernicus Workshop of Innovative Methods of Noise and Vibration Analysis on Rotating Machinery for Purpose of Quality Control, Monitoring and Diagnostics, Ilmenau (Germany), September 1995, pp.1-8.

[113] DFDP4/plus Digital Filter Design Package, Instruction Manual. Atlanta Signal Processors, Inc., Atlanta, 1996.

[114] Filter Design and Analysis System (FDASG) for the Motorola HC16. Reference guide, Momentum Data Systems, Inc., Costa Mesa, USA.

[115] QEDesign, Digital Filter Design System for Analog Devices ADSP-21000 Family Processors. Student version reference guide, Momentum Data systems, Inc., Costa Mesa, USA.

[116] DSP Design and Simulation Using the Simulink DSP Blockset. Simulink technical Computing Briefs, The MathWorks, Inc., 1995, pp.1-17.

[117] Lee, E.A. – Ho, W.H. – Goei, E.E. – Bier, J.C. – Bhattacharyya, S.: Gabriel: A Design Environment for DSP. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol.37, No.11, November 1989, pp.1751-1762.

[118] Keeling, N. – Ahluwalia, K.: SPOX: a high Level approach to DSP. *Electronic Engineering*, March 1990, pp.77-84.

[119] The Almagest, Vol.1 Ptolemy 0.6, User's Manual. University of California at Berkeley, June 1996.

[120] Jonuscheit, H.: Gearbox Testing Method. Proceedings of the 3rd International Conference DSP'99, Herľany, Slovakia, September 1997, pp.60-63.

[121] ANOVIS™ – Acoustic quality assessment and protective & condition monitoring for engines, gearboxes, components and their test benches. MEDAV Digitale Signalverarbeitung GmbH, Uttenreuth, Germany, 1999, (www.medav.de).

[122] Drutarovský, M.: Programming under Minisys DSP Operating System. Proceedings of the 4th Copernicus Workshop of Innovative Methods of Noise and Vibration Analysis on Rotating Machinery for Purpose of Quality Control, Monitoring and Diagnostics, Liberec, February 1996, pp.1-8.

[123] Drutarovský, M.: New Approach to Minisys DSP Software Development. Proceedings of the 6th Copernicus Workshop of Innovative Methods of Noise and Vibration Analysis on Rotating Machinery for Purpose of Quality Control, Monitoring and Diagnostics, Starý Smokovec, November 1996, pp.1-9.

[124] Drutarovský, M.: Universal Minisys DSP Operating System. Proceedings of the 8th Copernicus Workshop of Innovative Methods of Noise and Vibration Analysis on Rotating Machinery for Purpose of Quality Control, Monitoring and Diagnostics, Balatonfured, Hungary, October 1997, pp.1-5.

[125] Stupák, C.: Adaptívna filtrácia na signálových procesoroch Motorola DSP5600X. Diplomová práca, KEMT FEI TU v Košiciach, Košice, apríl 1997.

[126] Drutarovský, M. – Stupák, S. – Kocur, D.: Kodek rečového signálu na báze adaptívnej delta modulácie - popis technickej a programovej realizácie na signálovom procesore MOTOROLA DSP56002. Výskumná správa projektu VHČ pre TTC Telecom, s.r.o., Košice, január 1998, s.1-75.

[127] CS4215, 16-Bit Multimedia Audio Codec. DS76F2. Crystal Semiconductor Corporation, September 1993, pp.1-37.

[128] Lane, J. – Hillman, G.: Implementing IIR/FIR Filters with Motorola's DSP56000/DSP56001 Digital Signal Processors. Application Note APR 7/D Rev.2, Motorola, Inc., 1993.

[129] Frerking, M.E.: *Digital Signal Processing in Communication Systems*. Kluwer Academic Publishers, Boston, 1994.

[130] Lindquist, C.S.: *Adaptive & Digital Signal Processing with Digital Filtering Applications*. Stewards & Sons, New York, 1989.

[131] Thomä, R.: *Fensterfunktionen in der DFT – Spektralanalyse*. MEDAV GmbH, Uttenreuth, 1995, ISBN 3-9804152-0-1.

[132] Levický, D. – Marchevský, S. – Kocur, D. – Drutarovský, M. – Galajda, P.: Harmonický analyzátor sieťového napätia HARAN30-1, popis technického riešenia. Výskumná správa projektu VHČ pre VSE Košice, Košice, január 1995.

[133] Levický, D. – Drutarovský, M. - Galajda, P. - Kocur, D. - Marchevský, S.*:* Adaptive Goertzel's algorithm for harmonic analysis of power voltage. Proceedings of the 40. Internationales Wissenschaftliches Kolloquium, Ilmenau, Germany, September 1995, Band 1, ISSN 0943-7207, pp.473-478.

[134] Levický, D. - Drutarovský, M. - Galajda, P. - Kocur. D. - Marchevský, S.: Adaptive Goertzel's Algorithm for DFT Computation with Higher Accuracy. *Radioengineering*, Vol.5, No.1, April 1996, pp.1-6.

[135] Rektorys, K. a kol.: *Přehled užité matematiky*. SNTL Praha, 1981.

[136] Proakis, J.G. - Manolakis, D.G.: *Introduction to Digital Signal Processing*. Macmillan Publishing Company, New York, 1988.

[137] Drutarovský, M.: High Level Approach to Spectral Analysis on Motorola Digital Signal Processors. Proceedings of the 3rd International conference DSP'97, Herľany, September 1997, pp.53-56.

[138] Zetík, R. – Drutarovský, M.: Extended Library for Spectral Analysis on Motorola DSP5600X. Proceedings of the 8th Copernicus Workshop of Innovative Methods of Noise and Vibration Analysis on Rotating Machinery for Purpose of Quality Control, Monitoring and Diagnostics, Balatonfured, Hungary, October 1997, pp.1-4.

[139] Smith, W.W. – Smith, J.M.: *Handbook of REAL TIME Fast Fourier Transforms: Algorithms to Product Testing*. IEEE Press, New York, 1995.

[140] Guy, R.L. – Chen, S. – Chen, W.: Implementation of Fast Fourier Transforms on Motorola's Digital Signal Processors. Application Note APR 4/D Rev.3, Motorola, Inc., 1995.

[141] Duhamel, P.: Implementation of "Split – Radix" FFT Algorithms for Complex, Real, and Real - Symetric Data. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.34, No.2, June 1986, pp.285-295.

[142] Sorense, H.V. – Jones, D.L. - Heideman, M.T. – Burrus, C.S.: Real - Valued Fast Fourier Transform Algorithms. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.35, No.6, June 1987, pp.849-863.

[143] Lutkenhoner, B. – Ross, B.: Software manipulations to speed up a real – valued fast Fourier transform algorithm. *Computer Methods and Programs in Biomedicine*, 29 (1989), pp.129-142.

[144] Wu, Y.: New FFT Structures Based on the Brunn Algorithm. *IEEE Transactions on Accoustics, Speech and Signal Processing*, Vol.38, No.1, June 1990, pp.188-191.

[145] Drutarovský, M.: New Features of Minisys DSP Operating System. Proceedings of the 5th Copernicus Workshop of Innovative Methods of Noise and Vibration Analysis on Rotating Machinery for Purpose of Quality Control, Monitoring and Diagnostics, Budapest, June 1996, pp.1-5.

[146] Drutarovský, M. - Zetík, R.: The Library for Spectral Analysis on Motorola DSP5600X. Proceedings of the 7th Copernicus Workshop of Innovative Methods of Noise and Vibration Analysis on Rotating Machinery for Purpose of Quality Control, Monitoring and Diagnostics, Klánovice, Czech Republic, April 1997, pp.1-9.

[147] Strama, O. – Thomä, R. – Groppe, H.: Order Selective Vibration Analysis for Diagnosis and Quality Control of Gear Boxes. Proceedings of the Inter-Noise 97 conference, Budapest, August 1997, Vol.III, pp.1547-1550.

[148] Groppe, H. – Jonuscheit, H. – Strama, O. – Thomä, R.: Ordunganalyse. Contribution at MESSCOMP'96, Expert Verlag, Renningen-Malsheim, Germany, pp.122-127.

[149] Drutarovský, M.: Order Analysis based on Convolutional Approach. Proceedings of the 8th Copernicus Workshop of Innovative Methods of Noise and Vibration Analysis on Rotating Machinery for Purpose of Quality Control, Monitoring and Diagnostics, Balatonfured, Hungary, October 1997, pp.1-7.

[150] Drutarovský, M.: Polyphase Order Analysis based on Convolutional Approach. Proceedings of the 43. Internationales Wissenschaftliches Kolloquium, Ilmenau, Germany, September 1998, Band 1, ISSN 0943-7207, pp.393-398.

[151] Drutarovský, M.: Polyphase Order Analysis based on Convolutional Approach. *Radioeengineering*, Vol.8, No.2, June 1999, pp.43-48.

[152] Orfanidis, S.J.: *Introduction to Signal Processing*. Prentice Hall, New York, 1996.

[153] Skalar, B.: Rayleigh Fading Channels in Mobile Digital Communication Systems. *IEEE Communications Magazine*, September 1997, pp.136-155.

[154] Turletti, T. – Bentzen, H.J. – Tennenhouse, D: Toward the Software Realization of a GSM Base Station. *IEEE Journal on Selected Areas in Communications*, Vol.17, No.4, April 1999, pp.603-612.

[155] Ungerboeck, G.: Adaptive Maximum-Likelihood Receiver for Carrier-Modulated Data-Transmission Systems. *IEEE Transactions on Communications*, COM-22, No.5, May 1974, pp.624-636.

[156] Mouly, M. – Pautet, M.B.: *The GSM System for Mobile Communications*. 1992, ISBN 2-9507190-0-7.

[157] Steele, R.: *Mobile Radio Communications*. Pentech press publishers, London, 1992.

[158] Digital cellular telecommunications system; Modulation (GSM 05.04). ETSI standard ETS 300959, May 1997, pp.1-10.

[159] Laurent, P.A.: Exact and Approximate Construction of Digital Phase Modulations by Supperposition of Amplitude Modulated Pulses (AMP). *IEEE Transactions on Communications*, Vol.34, No.2, Feb1986, pp.150-160.

[160] Digital cellular telecommunications system (Phase 2); Multiplexing and multiple access on the radio path (GSM 05.02). ETSI standard ETS 300574, April 1997, pp.1-39.

[161] Blahut, R.E.: *Digital Transmission of Information*. Addison-Wesley, New York, 1990.

[162] Digital cellular telecommunications system (Phase 2+); Radio transmission and reception (GSM 05.05). ETSI standard ETS 300910, January 1998, pp.1-50.

[163] Kočan, V.: Simulačné prostredie PTOLEMY. Diplomová práca, KEMT FEI TU v Košiciach, Košice, apríl 1998.

[164] Shepherd, S. J: An Approach to the Cryptoanalysis of Mobile Stream Ciphers. IEE Colloquium on Security and Cryptography Applications to Radio Systems, Digest No. 1994/141, Savoy Place, London, 3 June, 1994.

[165] Shepherd, S. J: Cryptoanalysis of the GSM A5 Cipher Algorithm. IEE Colloquium on Security and Cryptography Applications to Radio Systems, Digest No. 1994/142, Savoy Place, London, 3 June, 1994.

[166] Mehrotra, A. – Golding, L.S.: Mobility and Security Management in the GSM System and Some Proposed Future Improvements. *Proceedings of the IEEE*, Vol.86, No.7, July 1998, pp.1480-1497.

[167] Massey, J.L: An Introduction to Contemporary Cryptology. *Proceedings of the IEEE*, Vol.76, No.5, May 1988, pp.533-549.

[168] Schneier, B.: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, New York, 1996.

[169] McClellan, J.H. – Schafer, R.W. – Yoder, M.A.: A Changing Role for DSP Education. *IEEE Signal Processing Magazine*, May 1998, pp.16-18.

[170] Bursky, D.: Advanced CPUs, Multimedia ICs Deliver Top Throughputs. *Electronic Design*, February 19, 1996, pp.55-74.

[171] Weiss, S.: Scalar Supercomputer Architecture. *Proceedings of the IEEE*, Vol.77, No.12, December 1989, pp.1970-1982.

[172] Eyre, J. – Bier, J.: DSP processors hit the main-stream. *Computer*, August 1998, pp.51-59.

[173] Kuo, B.J. – Lou, J.H.: *Low- voltage CMOS VLSI Circuits*. John Wiley & Sons, New York, 1999.

[174] Eyre, J. – Bier, J.: Infineon's TriCore Tackles DSP, Superscalar Hybrid Competes With Other Hybrids, DSPs. *Microprocessor Report*, Vol.13, No.5, April 19, 1999, pp.1-4 (article reprint).

[175] Ridder, R.J.: Efficient Programming Techniques for Digital Signal Processing. Proceedings of DSP Deutschland'98, October 1998, pp.184-191.

[176] Zetík, R.: FFT na signálových procesoroch Motorola DSP5600X. Diplomová práca, KEMT FEI TU v Košiciach, Košice, apríl 1997.

[177] Juraško, R.: Číslicová filtrácia na signálových procesoroch Motorola DSP5600X. Diplomová práca, KEMT FEI TU v Košiciach, Košice, apríl 1998.

[178] Kocur, D. – Kacvinský, J.: Order Approach to Signal Processing of Signals Generated by Reciprocated Machines. Proceedings of the 4th international conference DSP'99, Herľany, September 1999, pp.70-173.

[179] Wicker, S.B.: *Error Control Systems for digital Communication and Storage*. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[180] DSP56000 Digital Signal Processor Family Manual. DSP56KFAMUM/AD, Motorola, Inc., 1992.

[181] Floch, B.L. – Alard, M. – Berrou, C.: Coded Orthogonal Frequency Division Multiplex. *Proceedings of the IEEE*, Vol.83, No.6, June 1998, pp.982-996.

[182] Weste, N. – Skellern, D.J.: VLSI for OFDM. *IEEE Communications Magazine*, October 1998, pp.127-131.

[183] Drutarovský, M.: GSM Channel Equalization Algorithm – Modern DSP Coprocessor Approach. *Radioengineering*, Vol.8, No.4, December 1999, v tlači.

[184] Rabiner, L.R. – Schafer, R.W. – Rader, C.M.: The Chirp z – Transform and Its Application. *The Bell System Technical Journal*, Vol.48, No.5, May-June 1967, pp.1249-1292.

REGISTER

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

Z