SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 829

PROGRAMSKA IZVEDBA SUSTAVA ZA PRILAGODBU FREKVENCIJA OTIPKAVANJA I PROIZVOLJNO PRETIPKAVANJE SIGNALA

Alen Mešić

Zagreb, srpanj 2009.

Sadržaj

Uvod

Povezivanje dva asinkrona digitalna sustava koji rade na istim frekvencijama na prvi je pogled jednostavna zadaća. Idealno, trebalo bi ih se povezati samo preko jednog memorijskog elementa s dvodtrukim pristupom. No međutim, radi nesavršenosti takta može doći do razlika u frekvencijama (*jitter*) te ti isti uzorci više ne odgovaraju vrijednost signala u novom relativnom vremenu. Također, sustavi mogu raditi na različitim frekvencijama, te je u tom slučaju potrebno raditi interpolaciju orginalnih uzoraka da bi se dobile vrijednosti signala u zadanim relativnim vremenima. U svrhu prilagodbe frekvencija i pretipkavanja signala u ovom radu koriste se FIFO memorija i MOMS kubna interpolacija. Cijeli sustav implementiran je na procesoru za naprednu digitalnu obradu signala firme Analog Devices ADSP-bf537, porodica blackfin.

1. MOMS interpolacija

MOMS je interpolator koji zadovoljava svojstvo maksimalnog reda aproksimacije uz minimalnu dužinu pri čemu je red aproksimacije definiran kao stupanj smanjenja srednje kvadratne pogreške, tj. L^2 norme između originalne i rekonstruirane funkcije uz korak amplitudne kvantizacije koji teži k nuli. Postoji cijela klasa funkcija koje zadovoljavaju MOMS kriterij. Ta klasa interpolirajućih polinoma može se opisati kao

$$\phi(t) = \beta^N(t) + \sum_{k=1}^{N} p(k) \frac{d^k \beta^N(t)}{dt^k} \tag{1}$$

Budući da su B-spline funkcije konstruirane sukcesivnim konvolucijama sa pravokutnom funkcijom tako da se derivacija može izračunati kao

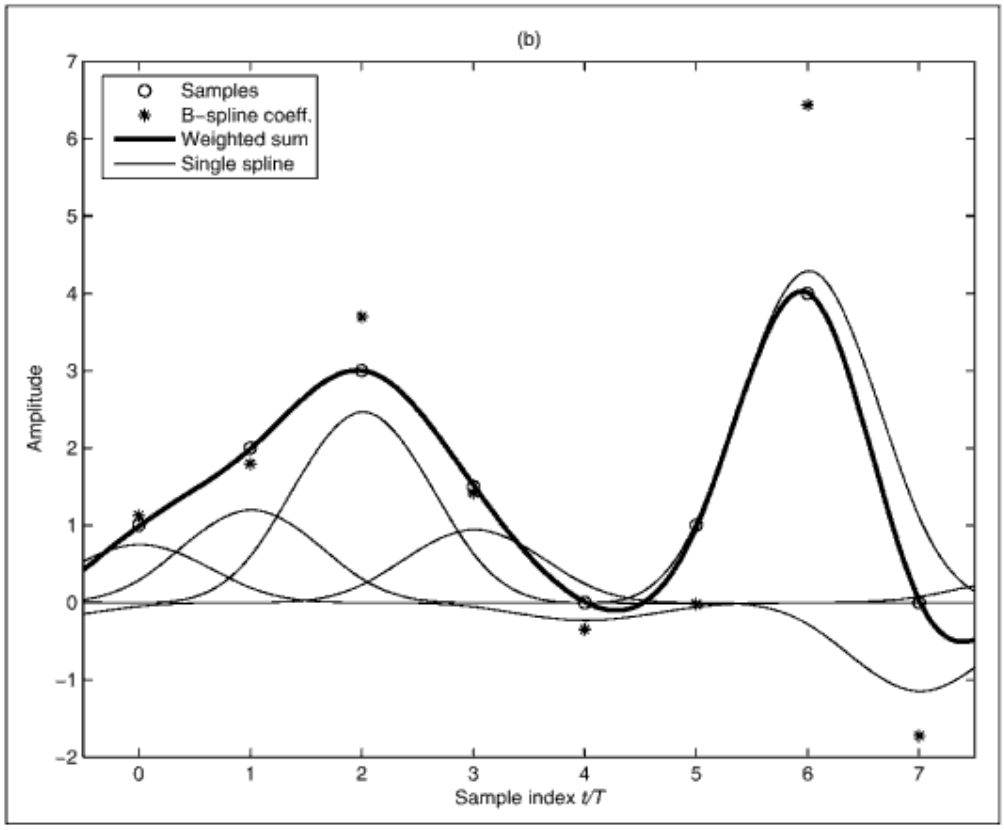
$$\frac{d\beta^{k+1}(t)}{dt} = \beta^k\left(t + \frac{1}{2}\right) - \beta^k\left(t - \frac{1}{2}\right) \tag{2}$$

Iz (1) je vidljivo da su parametri dizajna p(k) loji se biraju tako da se postignu željeni ciljevi dizajna, npr. za mnoge dizajne je poželjna simetrija interpolacijske baze, što znači da su svi neparni koeficijenti p(k)=0, pa se bira N=3, tj. tip kubične interpolacije i tada slijedi da je

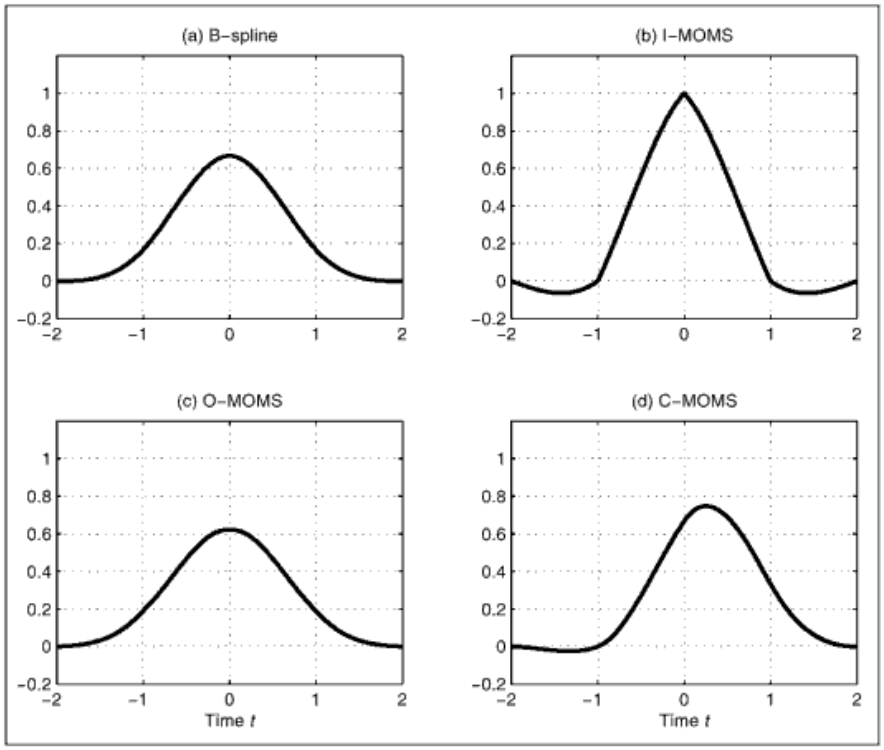
$$\begin{aligned} \phi(t) &= \beta^3(t) + p(2)\frac{d^2\beta^3(t)}{dt^2} \\ &= \beta^3(t) + p(2)(\beta^1(t+1) - 2\beta^1(t) + \beta^1(t-1)) \end{aligned} \tag{3}$$

Sada kao parametar dizajna ostaje odrediti samo p(2). Ako se želi dizajnirati interpolacijsku funkciju koja ne zahtijeva kompenzacijski filtar, proizlazi da p(2) treba biti -1/6. Taj slučaj se naziva I-MOMS, a rezultat interpolacije je jednak Lagrangeovoj interpolaciji, dakle ne daje optimalne rezultate interpolacije. Ako je cilj dizajna minimizirati srednju kvadratnu pogrešku, stavlja se p(2)=1/42 te se taj tip interpolacije naziva O-MOMS i daje manju pogrešku nego I-MOMS. Ukoliko se želi povećati odnos signal šum, bira se p(2)=1/28 koji daje za 1dB bolji rezultat nego O-MOMS. Budući da kompenzacijski filtar za O-MOMS ima nestabilni pol, mora se koristiti FIR aproksimacijski filtar, čime se gube prednosti male srednje kvadratne pogreške jer se FIR izvodi u aritmetici konačne

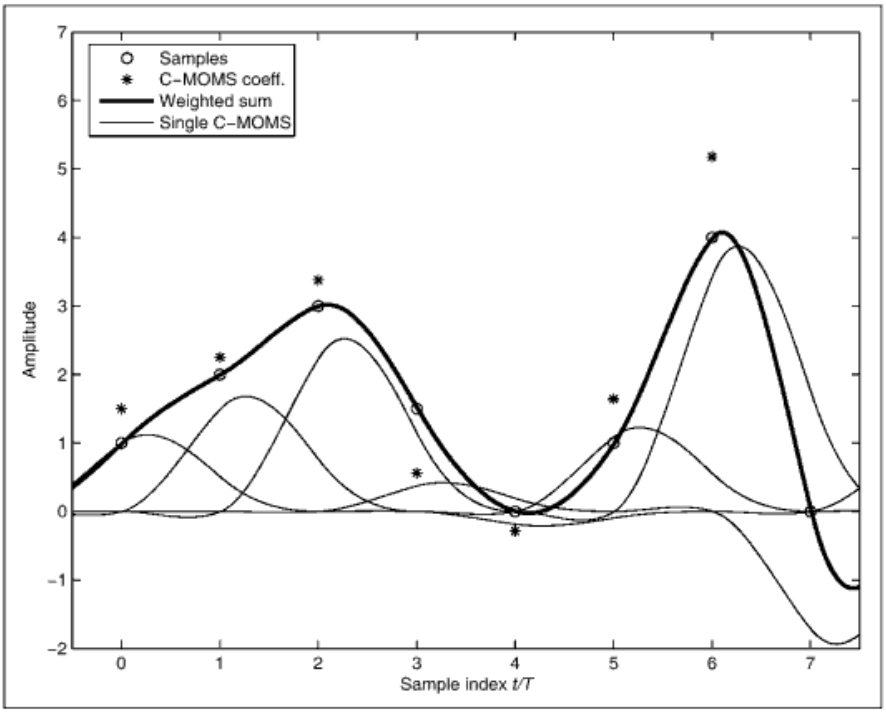
preciznosti. Ukoliko se odreknemo uvjeta simetričnosti funkcije baze, možemo dizajnirati MOMS funkcije takve da su interpolacijske funkcije otipkane u cjelobrojnim točkama φ(k) kauzalne funkcije, tj φ(-1)=0. To su tzv. C-MOMS funkcije koje su prilično glatke budući da je p(2)=p(3)=0. C-MOMS uvjet zahtijeva da je p(1)=-1/3 i dobije se asimetrična interpolacijska funkcija kojoj je glavna prednost da se za kompenzacijski filtar može upotrijebiti jednostavan stabilni jednopolni IIR filtar sa F(z)=1.5/(1+0.5z-1), za razliku od B-spline ili O-MOMS gdje je potrebno koristiti FIR aproksimacije. Zanimljivo je da se maksimumi funkcije i točke uzorkovanja ne poklapaju kod C-MOMS-a kao kod simetričnih baza, npr. B-spline-a što se vidi sa slika 1 i 3. Međutim, težinska suma C-MOMS-a prolazi kroz točke uzorkovanja što je karakteristično za spline interpolaciju. Iz slike 2 vidljivo je da C-MOMS nije simetrična oko nule.



Slika 1. Interpolacija kubičnom B-spline i FIR kompenzacijskim filtrom



Slika 2. Moguće MOMS baze duljine četiri

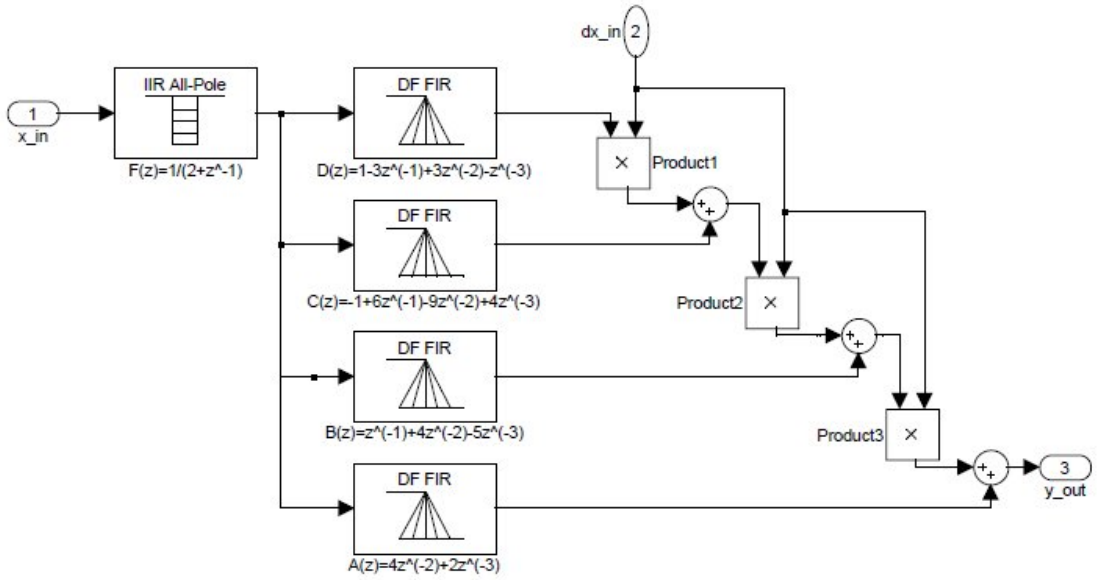


Slika 3. C-MOMS interpolacija koristeći IIR kompenzacijski filtar

U ovom radu korištena je upravo C-MOMS interpolacija za interpolator s frakcionalnim kašnjenjem. Eksperimentalno je dokazano da je C-MOMS bolja od B-spline, a nešto lošiji od O-MOMS kada je ova implementirana u punoj preciznosti. Implementacija će biti Farrow-ljevom strukturom koja prikazana matricom izgleda ovako :

$$\begin{array}{lcccc} d^0: & 0 & & +2x(n)/3 & +1x(n-1)/3 = c_0 \\ d^1: & 0 & +x(n+1)/6 & +2x(n)/3 & -5x(n-1)/6 = c_1 \\ d^2: & -x(n+2)/6 & +x(n+1) & -3x(n)/2 & +2x(n-1)/3 = c_2 \\ d^3: & x(n+2)/6 & -x(n+1)/2 & +x(n)/2 & -x(n-1)/6 = c_3 \end{array}$$

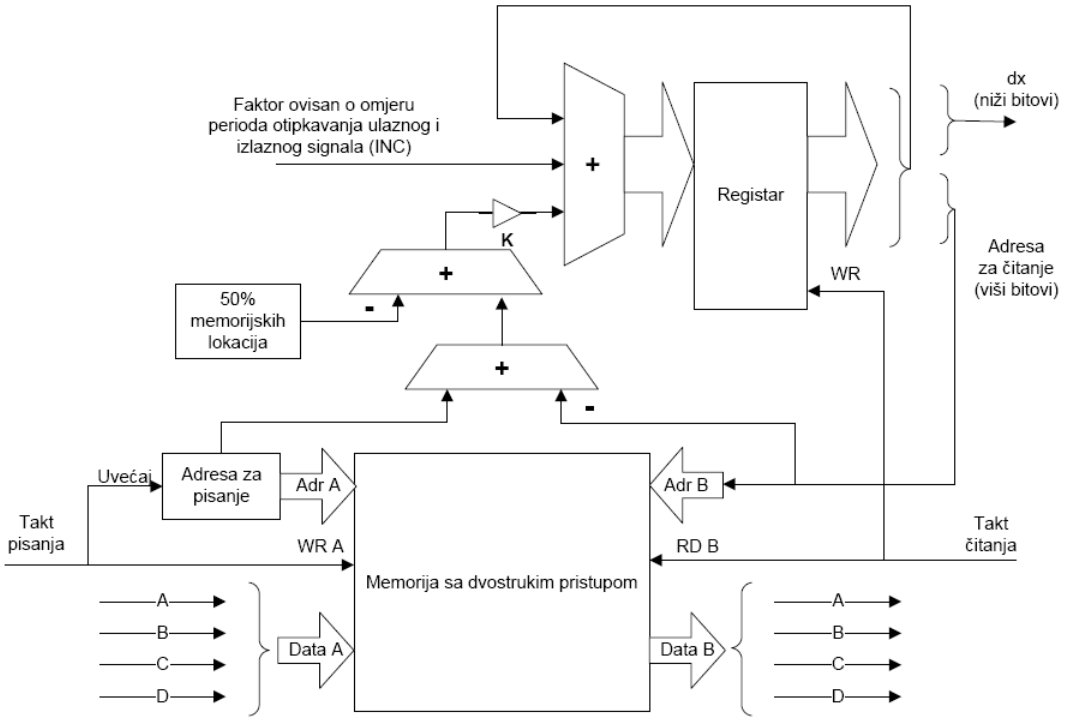
Kako ne bi došlo do pogreške prilikom računanja FIR filtara uzrokovane frakcionalnom aritmetikom, svi koeficijenti FIR filtara su pomnoženi sa 6 (množenje se svodi na posmak ulijevo i zbrajanje).



Slika 4. Blok shema MOMS sustava za interpolaciju

U svrhu integriranja interpolatora i sustava za prilagodbu frekvencija koeficijenti FIR filtra spremaju se u FIFO memoriju čija popunjenost se pokušava regulirati tako da uvijek iznosi približno 50% (npr. dupliciranjem ili preskakanjem podataka), sa ciljem izbjegavanja podpunjenosti ili prepunjenosti FIFO spremnika. FIFO međuspremnik je integriran u cjevovod interpolatora, i ima mogućnost upravljanja interpolatorom (brzinom pražnjenja). To omogućuje precizniju regulaciju brzine pražnjenja međuspremnika od grubog preskakanja ili dupliciranja

podataka opisanog u pojednostavljenom primjeru. Zbog preciznije regulacije brzine pražnjenja međuspremnika i upravljanja interpolatorom, frekvencije pisanja i čitanja ne moraju biti jednake, ali njihov omjer mora biti fiksiran. No zbog podrhtavanja bilo koje od frekvencija taj omjer u praksi varira, pa je potrebno vršiti kompenzaciju zadane brzine pražnjenja međuspremnika. Ugrađeni podsustav za kompenzaciju razlika frekvencija otipkavanja generira točke evaluacije polinoma. To je potrebno zbog mogućnosti precizne regulacije popunjenosti FIFO spremnika, odnosno zbog izbjegavanja izravnog preskakanja ili dupliciranja izlaznih uzoraka. Ulazna veličina u sustav je omjer perioda otipkavanja ulaznog i izlaznog signala. Ovo je samo procijenjena veličina, jer ulazni i izlazni taktovi će odstupati od nazivnih. Ukoliko se pojavi raskorak, logika sustava za kompenzaciju će vršiti korekcije ovog faktora. Blok-shema podsustava za kompenzaciju razlika frekvencija otipkavanja je prikazana na slici 3.



Slika 5. Podsustav za kompenzaciju razlika frekvencija otipkavanja

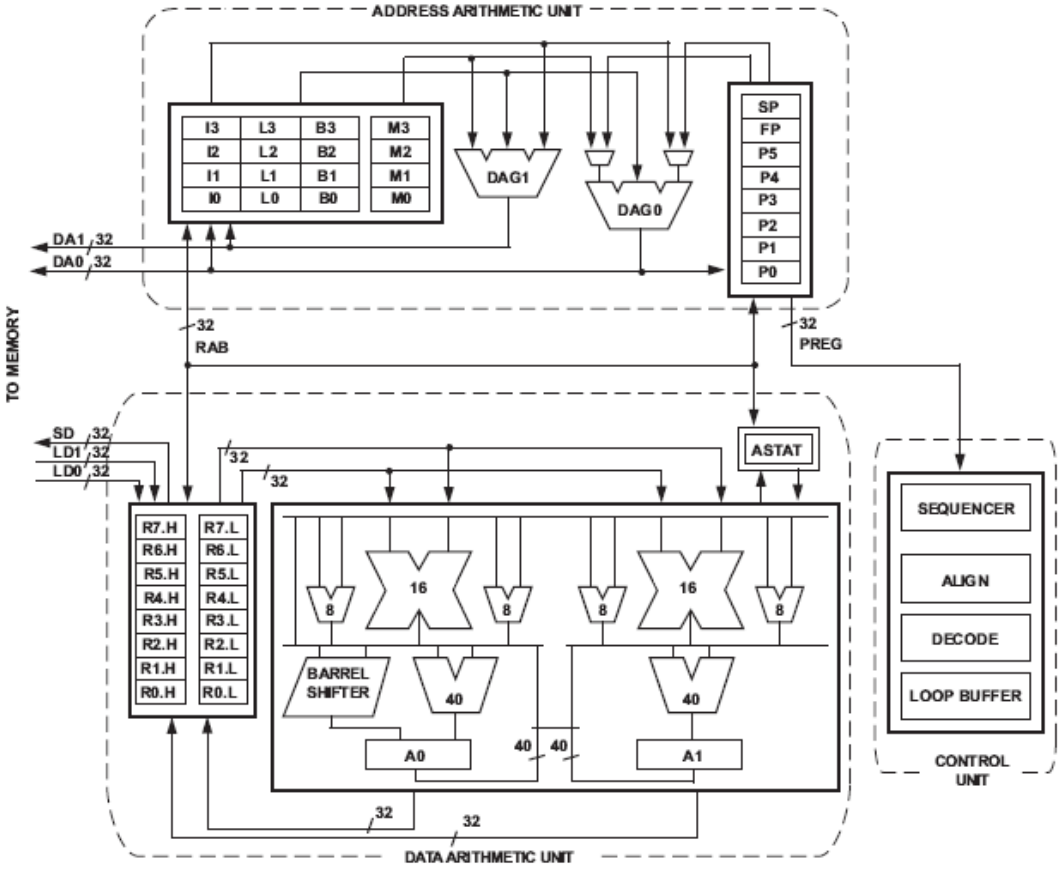
FIFO memorija je izvedena pomoću memorije sa dvostrukim pristupom i dvije adresne kazaljke. Adresna kazaljka za pisanje je neovisna o ostatku sustava i uvećava se na takt pisanja. Osim uvećanja adresne kazaljke pisanja, na takt

pisanja memorija sa dvostrukim pristupom preko pristupa A pohranjuje koeficijente polinoma na lokaciju na koju pokazuje adresna kazaljka za pisanje. Adresna kazaljka za čitanje je implementirana na složeniji način, jer je dio sustava za kontrolu brzine pražnjenja međuspremnika. Kazaljka je implementirana kao dio posebnog registra. Svakim čitanjem međuspremnika, preko pristupa B, registar se uvećava za iznos određen omjerom frekvencija čitanja i pisanja, te korigiran popunjenošću međuspremnika. Omjer frekvencija čitanja i pisanja ugrubo određuje hoće li se svaki upisani podatak **u prosjeku** iskoristiti točno jednom, više puta ili manje od jednom. Zbog odstupanja obje frekvencije od nazivnih, zadani faktor omjera frekvencija je potrebno korigirati, što se vrši razlikom kazaljke pisanja i kazaljke čitanja. Ako razlika iznosi točno 50% međuspremnika, ne vrši se dodatna korekcija. Ako je razlika veća od 50% vrši se korekcija uvećanja registra pozitivnim iznosom ovisnim o veličini raskoraka kazaljki,a ako je razlika manja od 50% vrši se korekcija uvećanja registra negativnim iznosom ovisnim o veličini raskoraka kazaljki. Razlika kazaljki čitanja i pisanja umanjena za 50% međuspremnika se ne koristi izravno za korekciju uvećanja registra, već je potrebno obaviti skaliranje tog iznosa kako bi se podpunjenost ili prepunjenost međuspremnika korigirala u što kraćem vremenu, ali uz izbjegavanje većih oscilacija u popunjenosti međuspremnika koje se mogu pojaviti ukoliko premala prepunjenost ili podpunjenost izazove preveliku reakciju. Kao adresna kazaljka čitanja koriste se značajniji bitovi registra. Manje značajni bitovi registra se prenose na izlaz dx – odabir trenutka za interpolaciju unutar periode otipkanog izlaznog signala, odnosno „decimalni dio“ pri „preskakanju uzoraka“. Dubina međuspremnika je 256 lokacija. Optimalna popunjenost međuspremnika je 128 lokacija. Ovakva konfiguracija unosi dodatno kašnjenje u rad sustava od 128 uzoraka. Sustav se tipično primjenjuje za pretipkavanje signala otipkanih sa frekvencijom otipkavanja u minimalno kHz području, stoga je kašnjenje od dodatnih 128 uzoraka nezamjetno za rad u stvarnom vremenu.

2. Blackfin procesor

2.1. Značajke procesora

Blackfin procesori imaju RISC protočnu arhitekturu Blackfin procesorske jezgre su namjenjene digitalnoj obradi signala. Posjeduju dva 16-bitna sklopa za množenje, dvije 40-bitne ALU s po jednim 40-bitnim akumulatorom, četiri 8-bitne video ALU te 40-bitni posmačni registar. Može obrađivati 8, 16 ili 32-bitne cijelobrojne ili frakcionalne podatke sa ili bez predznaka.



Slika 6. Blackfin procesorska jezgra

Jezgra sadrži i osam 32-bitnih podatkovnih registara koji u radu sa 16-bitnim podacima služe kao 16 nezavisnih 16-bitnih registara. Svaki sklop za množenje podržava množenje dva 16-bitna podatka po ciklusu, s mogućnošću

akumulacije u 40-bitni akumulator. ALU-i obavljaju i sve uobičajene aritmetičke i logičke operacije kao i neke posebne, npr. modulo, zaićenje, zaokruživanje. Pri operacijama sa 16-bitnim podacima omogućeno je obavljanje paralelnih operacija nad parom registara,npr. zbrajanje gornjie polovice registra R1 s gornjom polovicom R0 te istodobno oduzimanje donje polovice R0 od donje polovice R1. postojanje dviju ALU omogućava paralelno izvršavanje bilo kojih dviju operacija, pa je tako moguće obaviti i do četiri instrukcije nad parom registara. Adresna jedinica daje dvije adrese za dva simultana dohvata iz memorije. Također sadrži 4 skupine registara za ostvarivanje kružnih buffera (FIFO memorija). U svakoj grupi postoji registar za baznu adresu, za duljinu buffera, uvećanje adrese, te pokazivač na iduću memorijsku lokaciju. Procesor ima harvardsku memoriju s hierarhijskom strukturom L1, koja je smještena blizu jezgre procesora pa radi na punoj brzini procesora, te L2 memoriju koja ima Von-Neumann-ovu arhitekturu. L2 memorija je znatno sporija, ali većeg kapaciteta od L1 memorije. Memorijski prostor se može izvana proširiti do 4Gb.

2.2. Implementacija programa

Programska podrška za razvoj asemblerskog koda za blackfin procesore je Visual DSP. Unos asemblerskog koda je slobodnog formata, naredbe se mogu unositi u bilo koji dio reda, mogu se protezati kroz više redova, može se više naredaba nalaziti u jednom redu, ne razlikuju se velika i mala slova,a naredbe moraju završiti znakom točka-zarez. Za ostvarivanje FIFO memorije koriste se dvije skupine registara za kružne buffere sa istom bazom i duljinom, od kojih prva služi za upis podataka u memoriju a druga za čitanje. Ulazni pokazivač se uvećava upisom novih podataka u memoriju, dok se izlazni pri čitanju iz buffera uvećava za omjer frekvencija te razliku ulaznog i izlaznog pokazivača umanjenu za pola duljine buffera. Procesor sam vraća pokazivač na početak buffera kad dođe do kraja buffera. Po dolasku novog uzorka započinje ulazna prekidna rutina. Zapisuje se vrijednost uzorka te se računaju koeficijenti za MOMS interpolaciju koji se spremaju u FIFO memoriju. Za potrebe računanja koeficijenata, umjesto množenja uzoraka MAC jedinicama koristi se posmak i zbrajanje čime se dobiva na brzini izvođenja programa. Kad prijemni sustav zatraži sljedeći uzorak pokreće se izlazna prekidna rutina. Računa se nova adresa izlaznog pokazivača i vremenski trenutak uzorka dx kojim se izračunava vrijednost izlaznog uzorka.

2.3. Programski kod

Prekidi se omogućuju upisivanjem $FFC0010C_{16}$ na adresu $FFC0010C_{16}$, a upisom $FFFF_{16}$ na adresu $FFC01530_{16}$ port G se deklarira kao izlazni. Pokazivač P3 sadrži adresu porta F, a P2 porta G. Duljina kružnih buffera je 1024 memorijske lokacije što je dovoljno za 64 podatka, a počinju od adrese 1000_{16}. Omjer frekvencija i pomak izlaznog buffera potrebni su pri računanju trenutka interpolacije te se u tu svrhu spremaju na memorijsku lokaciju 2000_{16} i 2004_{16}. Izlazni pokazivač FIFO memorije uvećava se za početni pomak od polovice duljine buffera. Registri M0 i M1 služe uvećavanju pokazivača FIFO memorije. Prethodna tri i trenutni uzorak IIR filtra potrebni za izračunavanje FIR koeficijenata pamte se preko stoga, a početni se postavljaju na vrijednost 0. Dosadašnji dio koda služi samo za inicijalizaciju te na kraju dolazi naredba *idle* kojom se procesor stavlja u stanje mirovanja u svrhu štednje energije.

Dolaskom ulaznog ili zahtjevom za izlaznim podatkom pokreće se prekidna rutina te se izvršava potrebni dio koda. U ulaznom prekidnom potprogramu ulazni podatak se učitava te se računa novi IIR uzorak. Prethodni uzorci se čitaju sa stoga za potrebe računanja FIR koeficijenata, najstariji uzorak zamjenjuje se novim te se uzorci ponovno spremaju na stog. FIR koeficijenti računaju se kombinacijom pomaka i zbrajanja umjesto množenja radi ubrzanja izvedbe koda. Primjer računanja koeficijenta D prikazan je u nastavku. Uzorci su 16 bitni pa je moguće računati s polovicama registara. Koeficijenti se spremaju u FIFO memoriju preko pokazivača I0 koji se nakon svakog upisa uvečava za M0. Potprogram završava naredbom *reti* kojom se procesor vraća u stanje mirovanja.

```
r5.l=r0.l-r1.l(s);
r1.l=r1.l<<1;              //2r1
r5.l=r5.l-r1.l(s);
r5.l=r5.l+r2.l(s);         //r5(-=3*r1)+r2
r2.l=r2.l<<1;
r5.l=r5.l+r2.l(s);         //2r2
r5.l=r5.l-r3.l(s);
[i0++m0]=r5;               //upis "d" u buffer
```

U izlaznom prekidnom potprogramu učitavaju se vrijednosti pokazivača te se računa razlika izlaznog i ulaznog. Ako je razlika pozitivna oduzima joj se polovica duljine memorije, a ako je negativna, odnosno ako se izlazni pokazivač vratio na početak memorije, a ulazni još nije, razlici se pribraja polovica duljine memorije.

```
        r4=i0;
        r2=i1;
        cc=r4<r2;
        r0=r2-r4;                    //r0=delta
        r4=l0;
        r4=r4>>1;
        r0=r0+r4;
        if !cc jump manji;
        r0=r0-r4;
manji:
```

Rezultat se skalira faktorom K te mu se pribraja omjer frekvencija i prethodna reprezentacija vremena interpolacije unutar intervala. Frakcionalni dio dobivene vrijednosti predstavlja novu reprezentaciju vremena interpolacije unutar intervala, a cijeli dio množi se sa 16 i dodaje izlaznom pokazivaču. Množenje sa 16 osigurava da se pokazivač pomakne točno na početak FIR koeficijenata (4 koeficijenta po 4 bajta) odnosno na idući interval. Da ne bi dolazilo do stalnih oscilacija izlazne frekvencije male neusklađenosti pomaka pokazivača se zanemaruju, odnosno zadaje se petlja histereze. To se postiže pomoću dva uvjetna skoka.

```
        if !cc jump hist;
        cc=r1<-4;
        if cc jump hist;
        r1=0;
hist:
```

Zatim se iz iračunatih vrijednosti računa vrijednost izlaznog pokazivača te se ta i vrijednost reprezentacije vremena interpolacije unutar intervala pohranjuju na memorijsku lokaciju 2004_{16} za potrebe idućeg izlaznog prekidnog programa. Iz FIR koeficijenata i vrijednosti reprezentacije vremena interpolacije računa se interpolirana vrijednost na način prikazan slikom 4, odnosno formulom ((D*dx+C)*dx+B)*dx+A koja se šalje na port G. Potprporam završava naredbom *reti*.

```
r2=[i1++m1];
r1.l=r4.l*r2.l;
r2=[i1++m1];
r1.l=r1.l+r2.l(s);
r1=r1.l*r4.l;
r2=[i1++m1];
r1.l=r1.l+r2.l(s);
r1.l=r1.l*r4.l;
r2=[i1++m1];
r4.l=r1.l+r2.l(s);              //izracun vrijednosti u dx
[p2]=r4;
```

Programski kod je uvršten u rad kao dodatak A.

Zaključak

Pri povezivanju asinkronih sustava, sustava sa vlastitim izvorima takta, ne možemo ostvariti izravnu vezu, čak ni ako sustavi rade na istim frekvencijama, jer radi nesavršenosti izvora takta povremeno dolaziti do drhtanja frekvencija što bi dovelo do gubitka podataka. Također, sustavi mogu raditi na različitim frekvencijama. U oba slučaja potreban je međusklop koji bi se brinu da ispravni podaci stignu od odašoljača do prijemnika. Za većinu primjena korisno je imati ova dva sklopa integrirana. To se postiže primjenom MOMS interpolatora i FIFO memorije s dvostrukim pristupom. U praksi se pokazalo da je za digitalnu primjeni C-MOMS interpolacija najbolja. Ovakav međusklop unosio bi kašnjenje ovisno o dubini FIFO memorije, no uzevši u obzir da su najniže frekvencije današnjih asinkronih sustava reda kHz i sa memorijom dubine 1kb kašnjenje je zanemarivo.

VLASTORUČNI POTPIS

Literatura

[1] Karlović Luka: Sustav za pretipkavanje signala i kompenzaciju razlika frekvencija otipkavanja, seminarski rad, FER, 2009.

[2] Haus Mario: MOMS interpolator, seminarski rad, FER, 2009.

[3] blackfin_pgr.ref.man.rev1.3, *Blackfin® Processor Programming Reference* Analog Devices, 2008.

www.analog.com/static/imported-files/processor_manuals/blackfin_pgr.ref.man.rev1.3.pdf, 10.03.2009.

[4] 50_asm_man.rev3.1, *Assembler and Preprocessor Manual. 2008.*

https://sluga.fer.hr/exchweb/bin/redir.asp?URL=http://www.analog.com/static/imported-files/software_manuals/50_asm_man.rev3.1.pdf, 10.03.2009.

[5] ADSP-BF537_EZ-KIT_Lite_Manual_Rev_2_4.pdf, *ADSP-BF537 EZ-KIT Lite Evaluation System Manual, 2008* .https://sluga.fer.hr/exchweb/bin/redir.asp?URL=http://www.analog.com/static/imported-files/eval_kit_manuals/ADSP-BF537_EZ-KIT_Lite_Manual_Rev_2_4.pdf, 10.03.2009.

PROGRAMSKA IZVEDBA SUSTAVA ZA PRILAGODBU FREKVENCIJA OTIPKAVANJA I PROIZVOLJNO PRETIPKAVANJE SIGNALA

Sažetak

Asinkrone sustave nije moguće izravno povezati radi uporabe više različitih izvora takta kod kojih dolazi do podrhtavanja frekvencija. Za potrebe prilagodbe frekvencija i pretipkavanje koristi se MOMS interpolacija s FIFO memorijom. Praksa je pokazala kako je C-MOMS kubična interpolacija najoptimalnija za digitalnu izvedbu. Za potrebe interpolacije koristi se IIR filtar te četiri FIR filtra kojima se računaju koeficijenti potrebni za interpolaciju. Interpolirana vrijednost računa se kombinacijom zbrajanja koficijenata i množenja s relativnim vremenom u kojem se računa interpolirana vrijednost. Program je implementiran u procesoru za naprednu obradu digitalnih signala BF537 firme Analog Devices.

Ključne riječi

Interpolacija, prilagodba frekvencija, pretipkavanje, MOMS, FIFO

SOFTWARE IMPLEMENTATION OF SISTEM FOR SAMPLING FREQENCY ADAPTATION AND ARBITRARY RESAMPLING

Summary

Direct linking of asynchronous systems is impossible because of usage of multiple clocks witch can jitter. For frequency adaptation and resampling purposes MOMS interpolation and FIFO memory are used. Usage has shown that C-MOMS cubic interpolation is the best for digital implementation. One IIR and four FIR filters are used witch calculate coefficients for interpolation. The interpolated value is calculated by adding and multiplying these coefficients by relative time in witch interpolated value is to be calculated. Software is implemented in Analog Devices processor for advanced digital signal processing BF537.

Keywords

Interpolation, frequency adaptation, resampling, MOMS, FIFO

Popis kratica i oznaka

MOMS – *maximum order minimum support*

FIFO – *first in first out*

ADSP – *advanced digital signal processing*

ALU – *aritmetic logic unit*

MAC – *multiply and acumulate*

RISC – *reduced instruction set computer*

Dodatak A

```
p1.l=0x010C;
p1.h=0xFFC0;
p0.l=0x1530;
p0.h=0xFFC0;
r5.h=0x1800;
r5.l=0;
[p1]=r5;                //omoguci prekide
r5.l=0xffff;
[p0]=r5;                //port g izlazni
p3.h=0xFFC0;
p3.l=0x0700;            //port f-ulaz
p2.h=0xFFC0;
p2.l=0x1500;            //port g-izlaz
l0=4096;
l1=l0;
r5.l=0x8000;
r5.h=0;
p1=0x2000;
[p1]=r5;                //omjer (0.5)
r5.h=2048;              //pocetni offset buffera
r5.l=0;
[p1+4]=r5;
r5=0;
[--sp]=r5;
[--sp]=r5;
[--sp]=r5;
[--sp]=r5;              //0. uzorak
b0=0x1000;              //pocetak buffera
b1=b0;
i1=0x1800;              //b1+offset
i0=b1;
m0=4;
```

```
        m1=4;                   //inicijalizacija buffera
        idle;

ulaz:
        r6=[p3];
        r0=[sp++];
        r1=[sp++];
        r2=[sp++];
        r3=[sp++];
        r5.l=r0.l>>>1;
        r5.l=r6.l-r5.l(s);
        r3=r2.l;
        [--sp]=r3;
        r2=r1.l;
        [--sp]=r2;
        r1=r0.l;
        [--sp]=r1;
        r0=r5.l;                //ucitavanje novog uzorka
        [--sp]=r0;

        r5.l=r0.l-r1.l(s);
        r1.l=r1.l<<1;           //2r1
        r5.l=r5.l-r1.l(s);
        r5.l=r5.l+r2.l(s);      //r5(-=3*r1)+r2
        r2.l=r2.l<<1;
        r5.l=r5.l+r2.l(s);      //2r2
        r5.l=r5.l-r3.l(s);
        [i0++m0]=r5;            //upis "d" u buffer

        r5.l=r0.l+r1.l(s);      //r0+2r1
        r1.l=r1.l<<1;           //4r1
        r2.l=r2.l<<2;           //8r2
        r5.l=r5.l+r1.l(s);
        r5.l=r5.l-r2.l(s);
        r2.l=r2.l>>>3;          //1r2
```

```
                r3.l=r3.l<<2;           //4r3
                r5.l=r5.l+r3.l(s);
                r5.l=r5.l-r2.l(s);
                [i0++m0]=r5;            //upis "c" u buffer

                r1.l=r1.l>>>2;          //1r1
                r2.l=r2.l<<2;           //4r2
                r5.l=r1.l+r2.l(s);
                r5.l=r5.l-r3.l(s);
                r3.l=r3.l>>>2;          //1r3
                r5.l=r5.l-r3.l(s);
                [i0++m0]=r5;            //upis "b" u buffer

                r3.l=r3.l<<1;
                r5.l=r2.l+r3.l(s);      //4r2+2r3
                [i0++m0]=r5;            //upis "a" u buffer
                reti;

izlaz:
                r4=i0;
                r2=i1;
                cc=r4<r2;
                r0=r2-r4;               //r0=delta
                r4=l0;
                r4=r4>>1;
                r0=r0+r4;
                if !cc jump manji;
                r0=r0-r4;
manji:
                r0=r0<<5;
                r5=[p1];
                r0=r0+r5;
                r1=[p1+4];
                r4=r1;
                r4.l=0;                 //rnd(staro)
```

```
        r1=r1+r0;               //r1=y
        r0=r1;
        r1.l=0;                 //r1=rnd(y)
        r2=r0-r1;               //dx
        r1=r1-r4;
        cc=r1<4(iu);
        if !cc jump hist;
        cc=r1<-4;
        if cc jump hist;
        r1=0;
hist:
        r1=r1<<4;
        r1=r1+r2;               //di.dx
        r0=i1;
        r0=r0+r1;
        r2=r0>>16;
        i1=r2;
        [p1+4]=r0;
        r4=r0.l;                //r4=dx

        r2=[i1++m1];
        r1.l=r4.l*r2.l;
        r2=[i1++m1];
        r1.l=r1.l+r2.l(s);
        r1=r1.l*r4.l;
        r2=[i1++m1];
        r1.l=r1.l+r2.l(s);
        r1.l=r1.l*r4.l;
        r2=[i1++m1];
        r4.l=r1.l+r2.l(s);      //izracun vrijednosti u dx
        [p2]=r4;
        reti;
```