Univerzitet u Beogradu
Elektrotehnički Fakultet

Uroš Minoski, 2019/0135

Delta sigma modulator

Diplomski Rad

Mentor:
doc. dr Dušan Grujić

Beograd, septembar 2023.

Sažetak

U ovom radu je dato osnovno teorijsko objašnjenje delta sigma modulatora, kao što je uobličenje kvantizacionog šuma i korišćenje nadodabiranja zarad povećanja rezolucije konverzije. Pored teorijskog objašnjenja, napisam je kod u VHDL-u koji implementira delta sigma modulator u hardveru.

Ključne reči: ADC, DAC, Delta-Sigma, IIR, quantization, VHDL

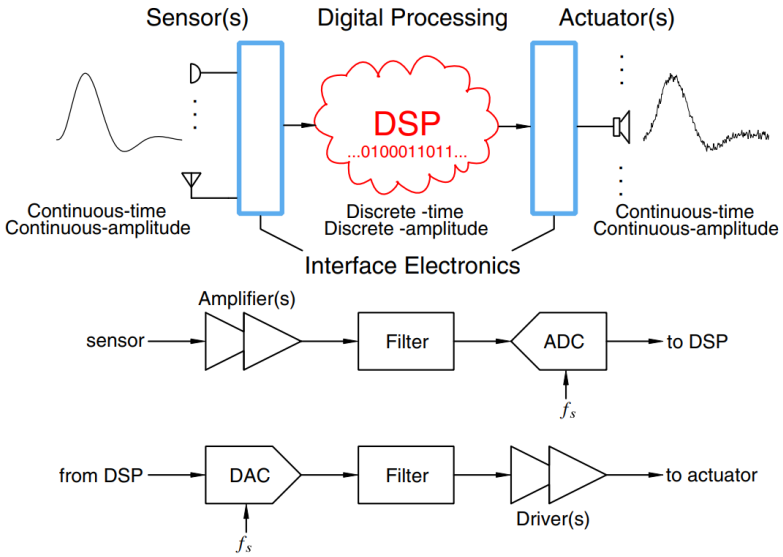
Sadržaj

Spisak slika

Glava 1

Uvod

Digitalna elektronika je danas u tolikoj meri rasprostranjena, da se u skoro svakom uređaju nalazi neka vrsta digitalnog sistema. Glavna karakteristika većine digitalnih sistema je da interaguju sa spoljašnjim svetom, ali postoji problem. Signali u spoljašnjem svetu su analogni, tj. kontinualni u vremenu i po amplitudi, dok su digitalni sistemi diskretni i po vremenu i po amplitudi. To znači da je potrebno izvršiti konverziju signala iz analognog u digitalni domen (A/D konvertor), i obrnuto (D/A konvertor).



Slika 1.1: *ADC i DAC interfejs prema spoljašnjem svetu. Slika je preuzeta iz [3].*

Postoje dve velike grupe konvertora, konvertori koji funkconišu po principu Nikvistove teoreme i konvertori koji koriste nadodabiranje (engl. over-

sampling) pri odabiranju [1, 3].

Obe grupe konvertora imaju prednosti i mane. Nikvistovi konvertori imaju jednostavnu hardversku realizaciju, ali su nepogodni u situacijama u kojima nam treba visoka rezolucija konverzije. Praktično ograničenje Nikvistovih konvertora je oko 12 bita [3], dok konvertori koji koriste nadodabiranje imaju mnogo zahtevniju hardversku implementaciju zarad visoke rezolucije konverzije.

Ono što omogućava visoku rezoluciju konverzije, u konvertorima koji koriste nadodabiranje, je kolo za uobličenje kvantizacionog šuma koje se zove $\Delta\Sigma$ (delta sigma) modulator.

Rad je organizovan u dve glave. Prva glava se bavi teorijskim osnovama potrebnim za razumevanje funkcionisanja $\Delta\Sigma$ modulatora, kao što su kvantizacija i kvantizacioni šum i osnovne odlike kola za uobličenje šuma. Druga glava se bavi problemima hardverske implementacije modulatora. Problemima koji nastaju usled konačne dužine reči u hardveru. Na kraju druge glave su dati rezultati simulacije modulatora u VHDL jeziku.

Glava 2

Teorijska osnova

Za razumevanje funkcionisanja $\Delta\Sigma$ modulatora, neophodno je osnovno razumevanje teorijskih koncepata kao što su kvantizacija i kvantizacioni šum i kolo za uobličavanje šuma.

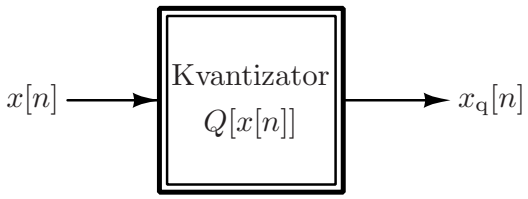
Osnovna uloga $\Delta\Sigma$ modulatora je da sa manje bita predstavi signale visoke rezolucije. Zbog činjenice da se za predstavu koristi mali broj bita (nego što je potrebno za datu rezoluciju) dolazi do kvantizacije, tj. pojave kvantizacionog šuma. Da bismo razumeli kako da smanjimo, tj. oblikujemo, kvantizacioni šum, prvo moramo da razumemo šta je kvantizacija i u kojim uslovima je možemo smatrati kao šum.

Kada razumemo šta je kvantizacija i kako utiče na signale, potrebno je razumeti osnovne tehnike za smanjenje uticaja kvantizacije na ulazni signal. Kolo koje obavlja tu funkciju se zove kolo za uobličenje šuma.

Na kraju se daje osnovni blok dijagram sistema za D/A konverziju koji eksploatiše $\Delta\Sigma$ modulator zarad velike rezolucije izlaznog signala iako je rezolucija samog D/A konvertora mala, tj. nedovoljna za tu rezoluciju kada ne bi bilo $\Delta\Sigma$ modulatora.

2.1 Kvantizacija i kvantizacioni šum

Signali u spoljašnjem svetu su analogni, tj. kontinualni su po amplitudi i u vremenu, pa ih je potrebno diskretizovati A/D (engl. analog to digital) konvertorom. U A/D konvertoru postoje dva procesa. Prvi je diskretizacija u vremenu, tj. odabiranje. Drugi proces je diskretizacija po amplitudi (sl. 2.1), s obzirom da je širina digitalne reči konačne dužine. Za razliku od diskretizacije po vremenu, koju je uvek moguće idealno rekonstruisati ako su zadovoljeni uslovi Teoreme o odabiranju [1], diskretizacija po amplitudi uvek unosi degradaciju u rekonstrukciji signala. Kvantizacija po amplitudi je neli-

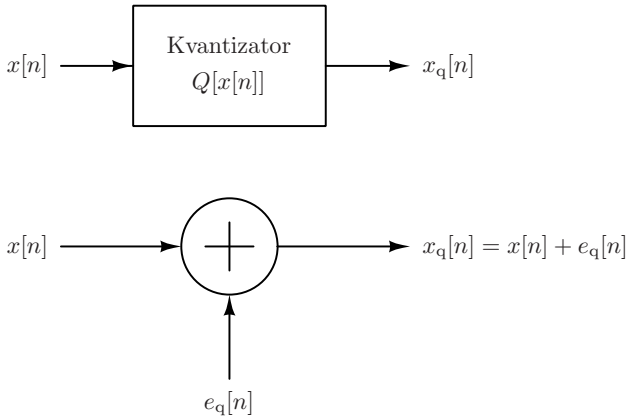


Slika 2.1: *Kvantizacija odbirka $x[n]$, po amplitudi. Slika je preuzeta iz [2].*

nearna operacija što anaizu kvantizacije, u opštem slučaju, čini vrlo teškom. To znači da se odbirci $x[n]$ ne mogu naći iz odbiraka $x_q[n]$ (sl. 2.1). Pod određenim uslovima, kvantizacija po amplitudi se može zameniti linearnim modelom (sl. 2.2), što u velikoj meri pojednostavlja analizu kvantizacije. Ti uslovi su [2]:

- Greška kvantizacije ima uniformnu raspodelu.
- Greška kvantizacije je stacionirani beli šum.
- Greška kvantizacije nije korelisana sa ulaznim signalom.
- Signal je stacioniran i njegova srednja vrednost je nula.

Linearni model zamenjuje nelinearni proces kvantizacije, aditivnim kvantizacionim šumom. Ovim postupkom možemo naći neke statističke karakteristike kvantizovanog signala.



Slika 2.2: *Linearni model kvantizatora. Slika je preuzeta iz [2].*

Snaga kvantizacionog šuma je statistički parametar i dat je izrazom [2],

$$P_N = \frac{\Delta^2}{12}. \tag{2.1}$$

gde je Δ korak kvantizacije. Izraz za snagu kvantizacionog šuma (2.1) ima dve vrlo važne osobine. Prva je da nam omogućava da nađemo vezu između SQNR (engl. signal to quantization noise ratio) i rezolucije A/D konvertora, tj. koraka kvantizacije Δ.

Druga, možda još bitnija, osobina je da izraz za snagu kvantizacionog šuma (2.1) ne zavisi od učestanosti odabiranja, već samo od koraka kvantizacije Δ. Posledica toga je da je spektralna gustina snage kvantizacionog šuma konstantna i iznosi,

$$N_e = \frac{P_N}{f_s} = \frac{\Delta^2}{12 f_s}, \tag{2.2}$$

gde je f_s učestanost odabiranja A/D konvertora. Ova osobina spektralne gustine snage kvantizacionog šuma nam omogućava da smanjimo snagu kvantizacionog šuma nadodabiranjem, što će biti objašnjeno dalje u tekstu.

SQNR je parametar koji se koristi da pokaže koliko kvantovan signal nalikuje originalnom signalu. Što je SQNR veći, to je razlika između kvantovanog i originalnog signala manja. Pod originalnim signalom podrazumevam signal diskretan u vremenu i kontinualan po amplitudi.

Vrednost SQNR, pod pretpostavkom da je originalni signal sinusoida maksimalne amplitude, tako da ne dođe do odsecanja signala prilikom kvantizacije, je dat izrazom [2],

$$SQNR = 6.02N + 1.76 \text{ [dB]}, \tag{2.3}$$

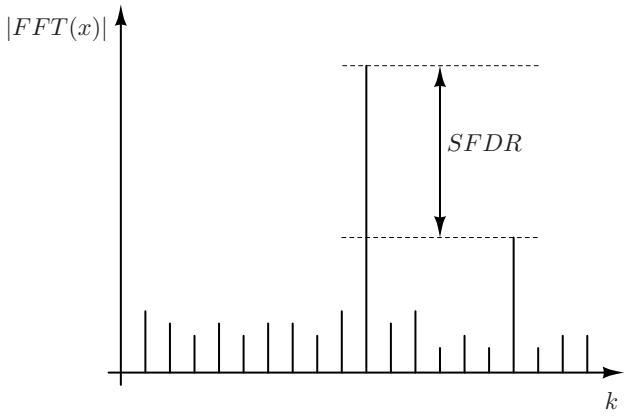
gde je N broj bita, tj. rezolucija A/D konvertora i korak kvantizacije $\Delta = 2^N$.

U sistemu se, pored kvantizacionog, mogu naći i drugi izvori šuma, kao što su termički, šum usled podrhtavanja signala takta (engl. jitter) i drugi. U nekim situacijama neće biti u potpusnoti zadovoljeni uslovi linearnosti kvantizatora. Ukoliko je korak kvantizacije dovoljno mali, kvantizacioni šum može biti u dovoljno velikoj meri korelisan sa ulaznim signalom što će smanjiti odnos snage signala i šuma. Pojaviće se, takozvani, spurovi u spektru. Sl. 2.3 pokazuje odstupanje od izraza (2.3). Zbog korelisanosti šuma kvantizacije sa ulaznim signalom, šum može u nekoj meri biti periodičan što prouzrokuje koncentrisanu energiju u spektru. Na SFDR (engl. spurious free dynamic range) utiču i drugi faktori pored korelisanosti šuma kvantizacije i ulaznog signala, kao što su neidealnosti analognih komponenata u A/D konvertoru.

Upravo zbog raznih drugih izvora šuma i neidealnosti se definiše SINAD (engl. signal to noise distortion [2]),

$$SINAD = \frac{P_{sig}}{P_N + P_{dist}}, \tag{2.4}$$

gde su P_{sig}, P_N i P_{dist} snage signala, ukupnog šuma i distorzije usled neidelanosti sistema. Kako je izraz (2.3) izveden samo u slučaju šuma kvantizacije



Slika 2.3: *SFDR. Slika je preuzeta iz [2].*

i bez distorzije, efektivan broj bita ENOB (engl. effective number of bits [2]) je definisan kao,

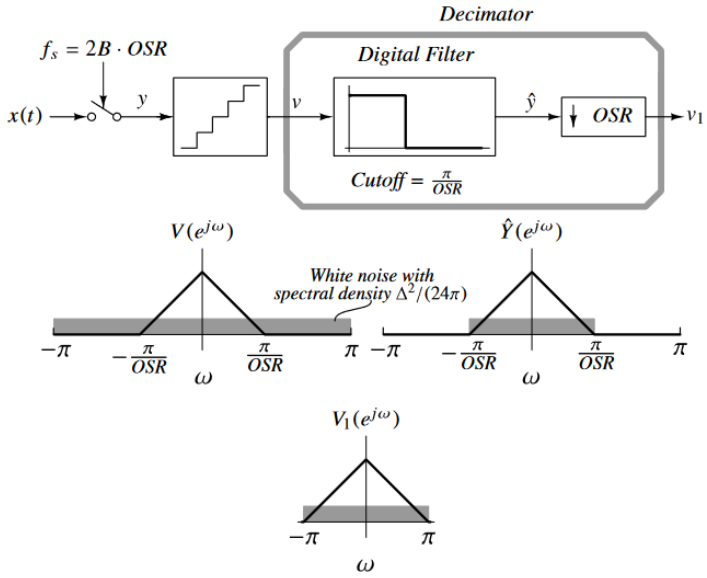
$$ENOB = \frac{SINAD - 1.76}{6.02}, \tag{2.5}$$

što nam govori da rezolucija kvantizovanog signala ne zavisi samo od koraka kvantizacije, već i od anbijentalnog šuma i stepena idealnosti konvertora.

Metod za poboljšanje SQNR, koji eksploatiše u velikoj meri i delta sigma modulator, je nadodabiranje. Nadodabiranje je povećanje učestanosti odabiranja zarad smanjenja kvantizacionog šuma, pri tome se ne povećava maksimalna učestanost signala koji se konvertuje. Ovo je moguće zbog činjence da je spektralna gustina snage kvantizacionog šuma konstantna (2.2). Sl. 2.4 pokazuje sistem koji koristi nadodabiranje i digitalno filtriranje signala kako bi povećao SQNR. S obzirom da je spektralna gustina snage kvantizacionog šuma obrnuto proporcionalna frekvenciji odabiranja, sa povećanjem f_s smanjuje se spektralna gustina snage. Ukoliko se takav signal digitalno filtrira, nakon A/D konverzije, snaga kvantizacionog šuma će se smanjiti OSR (engl. oversampling ratio, odnos frekvencije odabiranja i Nikvistove frekvencije [1]) puta.

Na sl. 2.4 se nalazi još jedan blok na izlazu. On služi da smanji učestanost odabiranja, kako bi se u digitalnom sistemu koristilo manje RAM (engl. random access memory), s obzirom da se na visokim frekvencijama ne nalazi nikakva informacija.

S obzirom da je glavni izvor šuma u $\Delta\Sigma$ odulatoru kvantizacija, tj. smatra se da su ostali izvori šuma dovoljno mali, u daljem takstu neće biti razmatrani drugi izvori šuma sem kvantizacije.



Slika 2.4: *Sistem koji eksploatiše nadodabiranje zarad povećanja SQNR. Slika je preuzeta iz [2]*

2.2 Uobličavanje šuma kvantizacije

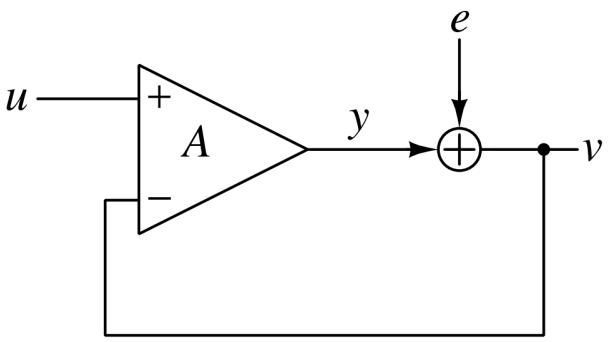
Već smo videli da $\Delta\Sigma$ modulator omogućava da sa A/D, tj. D/A, konvertorom manje rezolucije, možemo dobiti signal jako velike rezolucije. Postavlja se pitanje kako je to moguće?

Delta sigma modulator pored nadodabiranja koristi prednosti negativne povratne sprege [5]. Na sl. 2.5 je prikazan jednostavn pojačavač sa negativnom povratnom spregom. Pod pretpostavkom linearnog modela kvantizacije, kvantizaciju možemo smatrati šumom e. Prenosna funkcaija sistema sa sl. 2.5 je,

$$v = \left(\frac{A}{1+A}\right) u + \left(\frac{1}{1+A}\right) e, \tag{2.6}$$

gde se vidi da sa porastom pojačanja A, uticaj šuma e na izlaz v se smanjuje, u graničnom slučaju $A \to \infty$, $v \to u$.

Na prvi pogled, izraz (2.6) bi mogao da bude rešenje našeg problema. Ukoliko dovoljno povećamo pojačanje pojačavača u sistemu sa sl. 2.5 možemo potisnuti kvantizacioni šum u celom spektru. Nažalost ovo nije tačno! Da bismo videli zašto ovo ne odgovara stvarnosti, potreban nam je bolji matematički model (sl. 2.6). Svaki pojačavač sa povratnom spregom ima neko kašnjenje u povratnoj putanji. Ukoliko modelujemo i to kašnjenje, dobijamo

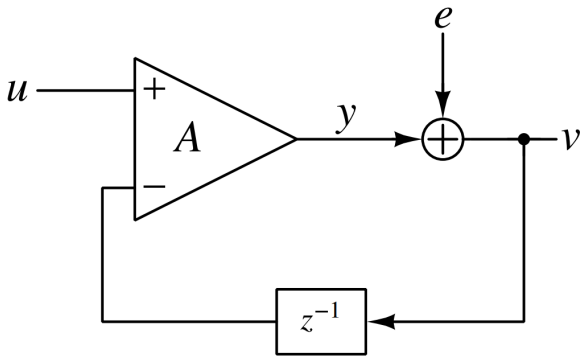


Slika 2.5: *Jednostavan pojačavač sa negativnom povratnom spregom. Slika je preuzeta iz [3].*

prenosnu funkciju,

$$V(z) = \left(\frac{A}{1 + Az^{-1}}\right) U(z) + \left(\frac{1}{1 + Az^{-1}}\right) E(z). \tag{2.7}$$

Problem koji ovakva struktura ima se može videti analizom polova prenosne funkcije. Obe prenosne funckije STF (engl. signal transfer function) i NTF (engl. noise transfer function) imaju iste polove. Prostom analizom se vidi da su polovi za pojačanja $|A| > 1$ izvan jediničnog kruga, tj. sistem je nestabilan za pojačanja veća od 1 [5]. U prethodnoj analizi smo zaključili da pojačanje treba da bude što veće, kako bismo što bolje potisnuli kvantizacioni šum.



Slika 2.6: *Bolji matematički model pojačavača sa negativnom povratnom spregom. Slika je preuzeta iz [3].*

Sistem sa sl. 2.5 je konstruisan tako da potisne kvantizacioni šum na celom spektru. Ukoliko iskoristimo prednost nadodabiranja, možemo potisnuti kvantizacioni šum samo na niskim učestanostima u spektru, dok visoke učestanosti nećemo dirati. Zbog nadodabiranja, informacija od značaja, tj.

signal, se nalazi samo na niskim učestanostima, tj. tamo gde je kvantizacioni šum potisnut. Najjednostavnija struktura koja obavlja ovu funkciju je integrator,

$$A = \frac{1}{1 - z^{-1}}. \tag{2.8}$$

Pa je prenosna funkcija u tom slučaju,

$$V(z) = 1 \cdot U(z) + (1 - z^{-1})E(z), \tag{2.9}$$

STF je sada 1, dok je NTF $(1 - z^{-1})$, što je propusnik visokih učestanosti prvog reda, baš ono što nam treba.

Snaga šuma u opsegu od interesa IBN (engl. in-band noise) je data izrazom [3],

$$IBN = \frac{\Delta^2}{36} \frac{\pi^2}{OSR^3}. \tag{2.10}$$

Ukoliko bismo povećali red integratora, npr. $NTF = (1 - z^{-1})^L$, dobili bismo veće potiskivanje šuma u opsegu od interesa [3],

$$IBN = \frac{\Delta^2}{12\pi(2L + 1)} \left(\frac{\pi}{OSR}\right)^{2L+1}. \tag{2.11}$$

Ovo sugeriše da bismo mogli da dobijemo proizvoljno potiskivanje šuma, u opsegu od interesa, ukoliko izaberemo dovoljno veliko L. Zvuči previše dobro da bi bilo istinito!

2.2.1 Dinamički opseg ulaznog signala

Pre nego što nastavimo, da se podsetimo sledećeg:

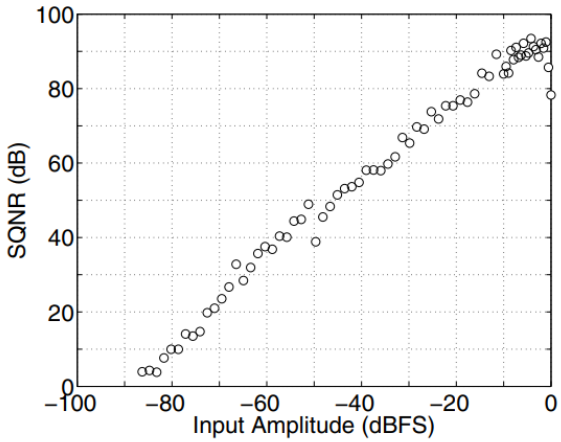
- Izraz za snagu šuma u opsegu od interesa (2.11) je izveden pod pretpostavkom da je kvantizacija aditivni šum sa uniformnom raspodelom. To je u velikoj meri ispunjeno ukoliko ulazni signal ne prelazi $[-M, M]$ gde je M nivo kvantizatora.

- Ulaz u kvantizator se sastoji od dva dela. Prvi je ulazni signal $U(z)$ na niskim frekvencijama, a drugi je uobličen kvantizacioni šum $(1 - z^{-1})^L E(z)$, koji zauzima ceo spektar.

Problem korišćenja velikog reda integratora je u tome što kvantizacioni šum ima komponente na visokim frekvencijama, koji se uobličavanjem pojačavaju. Što je veći red L to je pojačanje šuma na visokim frekvencijama veće (2.9). Zbog toga i za mali DC (engl. direct current) ulaz, kvantizator može da uđe u zasićenje, ukoliko je L dovoljno veliko. Ukoliko se to desi,

pretpostavka linearnog modela kvantizatora je narušena. Kvantizator ulazi u zasićenje i SQNR drastično opada.

Na sl. 2.7 se vidi da SQNR raste sa porastom amplitude ulaznog signala, sve dok amplituda ne poraste do određene granice kada kvantizator ulazi u zasićenje, tada SQNR počinje naglo da opada. Ukoliko je red L integratora dovoljno mali, granica amplitude ulaznog signala je nivo kvantizatora M. Ukoliko je red L veći, granica amplitude ulaznog signala će biti manja.



Slika 2.7: *SQNR u zavisnosti od amplitude ulaznog signala. Slika je preuzeta iz [3].*

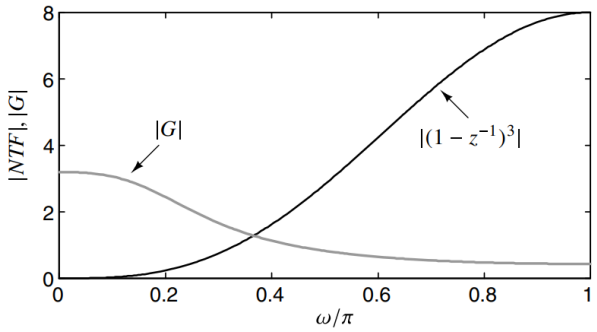
Maksimalnu stabilnu amplitudu ulaza MSA (engl. maximum stable amplitude) je vrlto teško naći ekzaktno, pa je to najbolje uraditi simulacijom, tj. emulacijom. Ono što znamo da ulaz u kvantizator ne varira mnogo kada kvantizator nije u zasićenju, međutim, kada je u zasićenju, ulaz naglo počinje da raste. Ono što bismo mogli da uradimo je da na ulaz dovedemo sporu rampu i da gledamo kada će ulaz u kvantizator naglo početi da raste. U tom trenutku očitamo koja je vrednost ulaznog signala i to predstavlja maksimalnu stabilnu amplitudu MSA.

2.2.2 Poboljšanje maksimalne stabilne amplitude ulaznog signala

Problem premalog MSA je preveliko pojačanje NTF na visokim frekvencijama. Rešenje bi bilo da nekako smanjimo pojačanje na visokim frekvencijama, a da to što manje utiče na pojačanje na niskim frekvencijama.

Posmatrajmo $NTF = (1 - z^{-1})^3$, koji ima previše veliko pojačanje na visokim frekvencijama. Dinamički opseg ovakvog sistema je vrlo mali, što

je bilo objašnjeno u prethodom pododeljku. Ukoliko NTF pomnožimo filtrom propusnikom niskih učestanosti G, dobili bismo smanjenje pojačanja na visokim učestanostima (sl. 2.8).



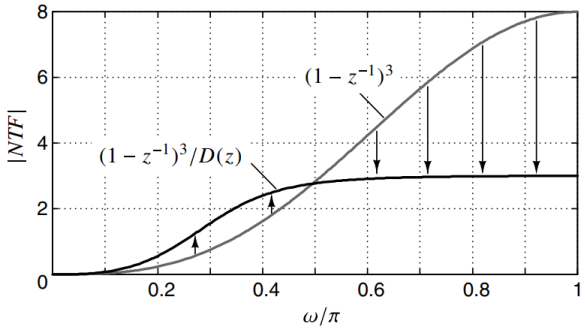
Slika 2.8: *Prenosne funkcije NTF i G. Slika je preuzeta iz [3].*

Polovi takvog filtra propusnika niskih učestanosti G, moraju biti što bliži $z = 1$, kako bi pojačanje na visokim učestanostima bilo što manje, ali ne smeju biti previše blizu $z = 1$, kako ne bi previše narušili pojačanje konačnog NTF na niskim učestanostima.

Konačan NTF je dat izrazom,

$$NTF = \frac{(1 - z^{-1})^L}{D(Z)}, \tag{2.12}$$

gde je $D(z) = \Pi_k(1 - z^{-1}p_k)$. Sl. 2.9 prikazuje $G(z) \cdot NTF(z)$ za $NTF = (1-z^{-1})^3$. Može se videti da je pojačanje na visokim učestanostima popriličnо smanjeno, ali da je i pojačanje na niskim učstanostima povećano, ali je oblik w^{-3} i dalje ostao.



Slika 2.9: *Prenosno funkcija konačnog NTF. Slika je preuzeta iz [3].*

2.2.3 NTF sa optimalnim razmakom između nula prenosne funkcije

Množenjem NTF sa G pogoršali smo prenosnu funkciju u opsegu od interesa, tj. na niskim učestanostima. To je zbog toga što polovi funckije G moraju da budu bliži $z = 1$ nego $z = -1$. Snaga kvantizacionog šuma u opsegu od značaja je [3],

$$IBN = \frac{\Delta^2}{12} \int_0^{\frac{\pi}{OSR}} k\, w^{2L} dw, \tag{2.13}$$

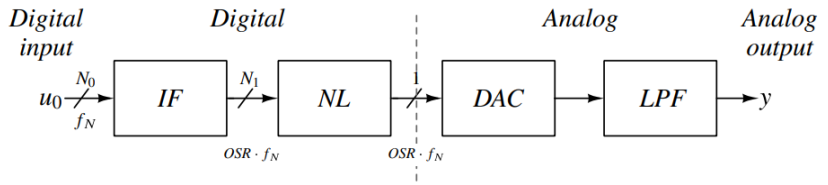
gde je $k = \frac{1}{d}$, a d je udaljenost pola od $z = 1$, G ima jedan konjugovano kompleksni par polova.

Mogli bismo da rasporedimo nule po jediničnom krugu, oko $z = 1$ tako da integral (2.13) bude minimalan. Na taj način ćemo rasporediti nule prenosne funkcije na veći opseg, tj. na opseg učestanosti od značaja i smanjiti varijaciju na niskim učestanostima.

2.3 Delta sigma D/A konvertor

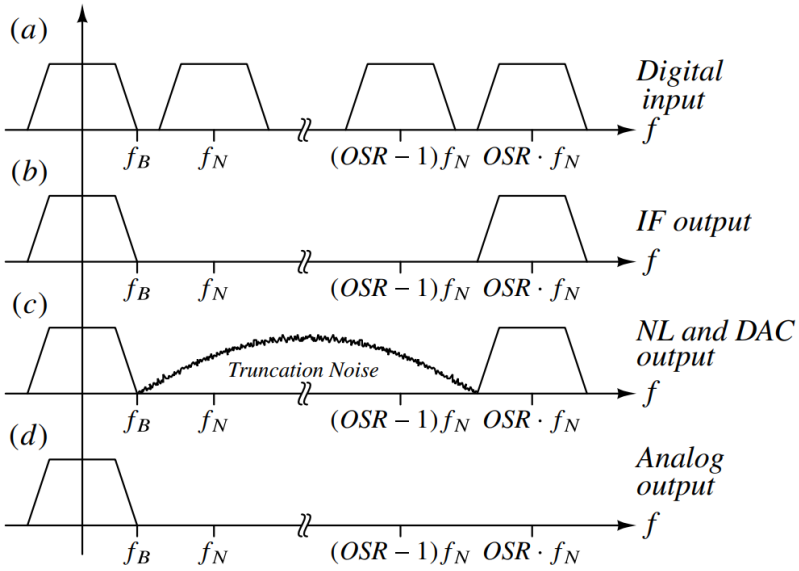
Dosadašnja analiza se podjednako odnosila na $\Delta\Sigma$ modulator i u A/D i u D/A konvertoru. U ovom delu će pažnja biti usmerena na D/A primenu modulatora.

Na sl. 2.10 je prikazan osnovni blok dijagram $\Delta\Sigma$ D/A kovertora. Sistem se sastoji od 4 bloka. Prvi služi da poveća frekvenciju odabiranja digitalnom signalu. To radi interpolacijom [2]. Drugi blok je blok za uobličenje šuma, o kome je bilo diskusije u prethodnim odeljcima. Treći blok je D/A konvertor, čija je rezolucija uglavnom od 1-5 bita. Poslednji blok je analogni filtar propusnik niskih učestanosti. Njegova uloga je da otkloni kvantizacioni šum na visokim frekvencijama iz signala.



Slika 2.10: *Osnovni blok dijagram $\Delta\Sigma$ D/A konvertora. Slika je preuzeta iz [3].*

Na sl. 2.11 su prikazani spektri signala u svakom delu sistema sa sl. 2.10. Lepo se vidi da bi bez nadodabiranja projektovanje $\Delta\Sigma$ modulatora bilo jako teško, tj. nemoguće. Potrebno je ostaviti mesta za kvantizacioni šum na visokim učestanostima. Upravo to radi nadodabiranje. Signal od značaja je transliran na opseg učestanosti $[0, \frac{\pi}{OSR}]$ (engl. oversampling ratio OSR, je faktor nadodabiranja) što ostavlja dovoljno mesta za kvantizacioni šum.



Slika 2.11: *Osnovni blok dijagram $\Delta\Sigma$ D/A konvertora. Slika je preuzeta iz [3].*

Glava 3

Hardverska implementacija

Pored teorijskih razmatranja, cilj ovog rada je i hardverska implementacija $\Delta\Sigma$ modulatora u VHDL jeziku. Centralni deo modulatora je IIR (engl. infinite impulse response) filtar, pa će najviše reči biti o njemu.

Veliki problem pri projektovanju modulatora u hardveru je projektovanje IIR filtra. S obzirom da u hardveru radimo sa konačnim širinama reči, neminovno će doći do odsecanja, tj. zaokruživanja, nekih međurezultata pri računanju izlaza digitalnog filtra. To odsecanje, tj. zaokruživanje čini digitalni filtar nelinearnim i može u potpunosti pokvariti prenosnu funkciju filtra, čak može učiniti filtar nestabilnim za neke ulazne signale, iako su polovi prenosne funkcije unutar jediničnog kruga.

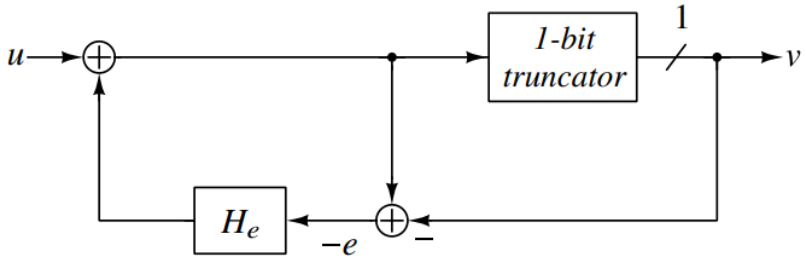
3.1 Error-Feedback struktura

Error-Feedback struktura predstavlja kolo za uobličenje kvantizacionog šuma u digitalnom domenu. Prenosna funkcija kola sa sl. 3.1 je,

$$V(z) = 1 \cdot U(z) + [1 - H_e(z)]\, E(z), \tag{3.1}$$

gde je $H_e(z)$ IIR filtar u povratnoj sprezi. Odbačeni LSB (engl. least segnificant bit) biti se vraćaju nazad i oduzimaju sa celim brojem, kako bi se našla greška kvantizacije koja se filtrira i sabira sa ulaznim signalom. Filtriranje greške kvantizacije e se vrši u povratnoj sprezi. Na sl. 3.1 je prikazano da je nivo kvantizatora 1, ali realizovan je kvantizator većeg nivoa, kako bi filtar H_e bio relaksiraniji.

Sa sl. 3.1 se vidi da se u sistemu nalaze sabirači koji zajedno sa aritmetikom u komplementu dvojke mogu napraviti veliki problem. Zbog konačne dužine reči lako može doći do prekoračenja. Ukoliko dođe do prekoračenja pozitivan broj će postati negativan, i obrnuto, što unosi veliku grešku (engl.



Slika 3.1: *Error-Feedback struktura. Slika je preuzeta iz [3].*

wraparound). Zbog toga treba uvesti limitere u sabirače. Ako se desi da rezultat prelazi dozvoljen opseg, rezultat se limitira na najveću, tj. najmanju vrednost.

Python kod koji nalazi koeficijente IIR filtra se nalazi u dodatku A.1. Potrebno je posebnu pažnju obratiti realizaciji IIR filtra, s obzirom da ćemo python kodom dobiti koeficijente filtra proizvoljne rezolucije, dok u hardveru radimo sa konačnom dužinom reči. Ukoliko nismo pažljivi, razni problemi mogu nastati prilikom predstavljanja koeficijenata IIR filtra na konačan broj bita.

VHDL kod koji implementira strukturu sa sl. 3.1 se nalazi u dodatku A.2.

3.2 Realizacija IIR filtra

Opšti oblik funkcije prenosa IIR filtra je dat količnikom dva polinoma [4],

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} a_k z^{-k}}{1 + \sum_{k=1}^{N} b_k z^{-k}} = \frac{A(z^{-1})}{B(z^{-1})}. \tag{3.2}$$

Izraz (3.2) nam govori da je za filtriranje jednog odbirka ulaznog signala filtrom N-tog reda potrebno $2N + 1$ operacija množenja i $2N$ operacija sabiranja sa dva sabirka.

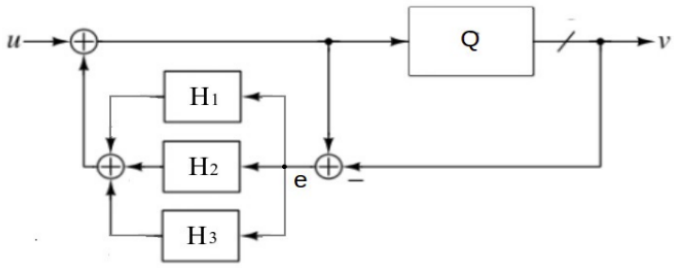
Postoji mnogo načina da se polazeći od funkcije prenosa formira realizaciona struktura koja pomoću blok dijagrama opisuje dati digitalni filtar. U ovom radu je odabrana paralelna struktura (sl. 3.2) bikvadratnih filtara realizovanih u direktnoj formi I i II.

Python kodom iz dodatka A.1 se dobijaju koeficijenti filtra,

$$H(z) = \frac{3.19382974 - 8.02002256z^{-1} + 8.73762976z^{-2} - 4.61756997z^{-3} + 0.97458298z^{-4}}{1 - 1.63632004z^{-1} + 1.47600867z^{-2} - 0.75840147z^{-3} + 0.2125798z^{-4} - 0.02541702z^{-5}}.$$

Filtar je podeljen na zbir tri bikvadratna filtra,

$$H(z) = H_1 + H_2 + H_3, \tag{3.3}$$



Slika 3.2: $\Delta\Sigma$ *modulator sa paralelnom strukturom IIR filtra u povratnoj sprezi.*

zbog efekta konačne dužine reči, što će biti objašnjeno u narednom podpoglavlju. Koeficijenti bikvadratnih filtara su,

$$H_1 = \frac{7.3765809}{1 - 0.3466036z^{-1}}, \tag{3.4}$$

$$H_2 = \frac{0.424071040 - 2.782608716z^{-1}}{1 - 0.66591402z^{-1} + 0.16260264z^{-2}}, \tag{3.5}$$

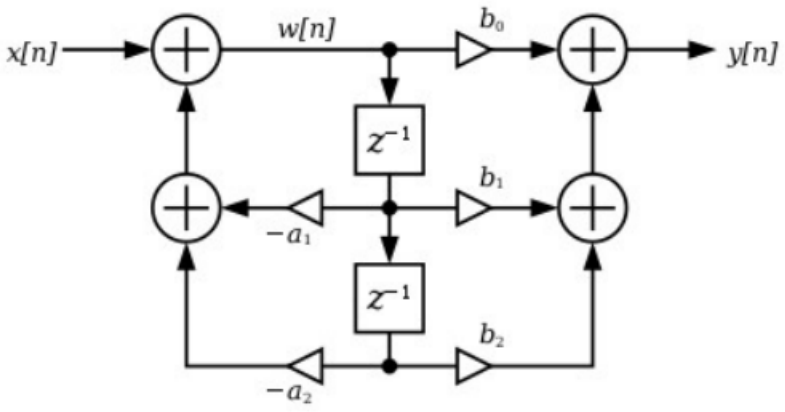
$$H_3 = \frac{-4.606822182 + 0.023331537z^{-1}}{1 - 0.62380242z^{-1} + 0.4509869z^{-2}} \tag{3.6}$$

Pored paralelne strukture (3.3) postoji i kaskadna struktura, gde je ukupna prenosna funkcija jednaka proizvodu filtara. Kaskadna struktura nije izabrana zato što ona ima veću kritičnu putanju, tj. kašnjenje u povratnoj putanji bi bilo veće nego kod paralelne strukture.

Filtri H_2 i H_3 su realizovani kao direkta II forma (sl. 3.3) da bi imali što manje elemenata za kašnjenje. Filtar H_1 je realizovan kao direktna I forma, s obzirom da ima samo dva nenulta koeficijenta a_0 i b_1. VHDL kod koji implementira bikvadratne filtre je,

```
x0 := resize(b00*e + a01*x0d, x0'high, x0'low);
w1 := resize(e + a11*w1d - a12*w1dd, w1'high, w1'low);
x1 := resize(b10*w1 - b11*w1d, x1'high, x1'low);
w2 := resize(e + a21*w2d - a22*w2dd, w2'high, w2'low);
x2 := resize(b21*w2d - b20*w2, x2'high, x2'low);
```

gde su `x0`, `x1` i `x2` izlazi bikvadratnih filtara H_1, H_2 i H_3 respektivno, `w1` i `w2` su međurezutati filtara H_1 i H_2, respektivno, dok su `x0d`, `w1d`, `w1dd`, `w2d` i `w2dd` zakasneli odbirci signala `x0`, `w1` i `w2`, respektivno. U kodu se može videti funckija `resize()` koja menja broj bita za predstavu broja u aritmetici sa fiksnom tačkom.



Slika 3.3: *Direktna II forma.*

Ostaje da se odredi izlaz celog filtra. Prema izrazu (3.3) dobija se kod, koji je jedan na jedan preslikan sa izrazom,

```
y_iir := resize(x0 + x1 + x2, y_iir'high, y_iir'low);
```

3.3 Uticaj konačne dužine reči na karakteristike IIR filtra

Implementacija digitalnih filtara može biti hardverska ili softverska. Bez obzira na to koji je tip implementacije u pitanju, koeficijenti i signali su predstavljeni u binarnom obliku sa konačnom dužinom reči.

Dakle, koeficijenti i signali su kvantovani, što za posledicu ima da diferencna jednačina koja opisuje digitalni filtar postaje nelinearna i veoma teška za analizu. Na sreću, ukoliko je korak kvantizacije mali u poređenju sa vrednostima koeficijena i amplitudama signala, analiza efekata kvantizacije se može značajno uprostiti [4], što znači da se mogu koristiti jednostavni statistički modeli.

Efekti konačne dužine reči mogu se svrstati u sledeće četiri kategorije [4]:

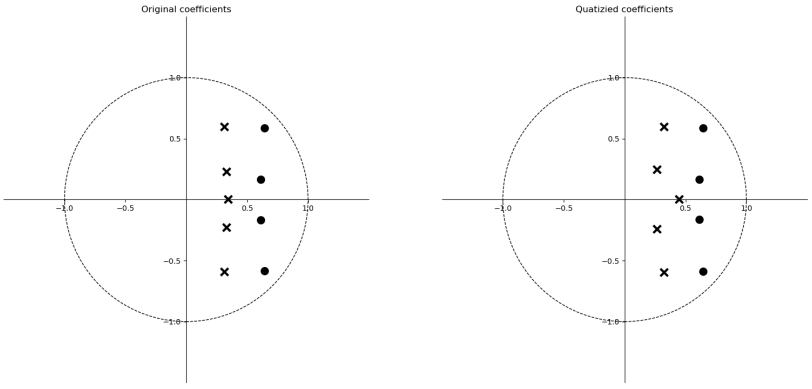
1. Greške usled kvantizacije koeficijenata.
2. Šum usled kvatizacije proizvoda.
3. Prekoračenje dinamičkog opsega.
4. Granični ciklusi.

Kvantizacioni šum je već obrađen u prvoj glavi. Ovde je akcenat na tome kako konačna dužina reči, tj. kvantizacija, utiče na performanse digitalnih filtara, konkretno na IIR filtre.

3.3.1 Greške usled kvantizacije koeficijenata

Kvantizaciojom, koeficijenti filtra se menjaju, što menja položaj nula i polova, tj. menja se prenosna funkcija filtra. Ta promena može biti mala ili velika, zavisi od toga koliko je filtar zahtevan i od toga kolika je dužina reči.

Kvantovanjem koeficijenata, IIR filtar može čak postati i nestabilan. U situacijama kada se polovi filtra nalaze vrlo blizu jediničnog kruga, kvantovanjem se može desiti da polovi izađu van jediničnog kruga. Ovo se ne dešava kod FIR filtara s obzirom da oni nemaju polove, tj. FIR filtri su uvek stabilni.



Slika 3.4: *Nule i polovi originalnog filtra i filtra sa kvantovanim koeficijentima.*

Na sl. 3.4 se nalaze nule i polovi filtra dobijeni python kodom iz dodatka A.1, i filtra sa kvantovanim koeficijentima na 4 bita za ceo deo i 8 bita za razlomljeni deo. Kao što se može videti sa slike, polovi ostaju unutar jediničnog kruga nakon kvantovanja, ali se njihov raspored malo promenio što dovodi do većih odstupanja u nepropusnom opsegu funkcije NTF, tj. na niskim učestanostima gde se nalazi signal od značaja.

Primer sa sl. 3.4 je izveden za direktnu strukturu IIR filtra. Ukoliko filtar razložimo na bikvadratne filtre u paralelnoj strukturi, ova greška kvantizacije će biti još manja. Što je red filtra manji, to će prenosna funkcija kvantovanog filtra više ličiti originalnoj prenosnoj funkciji. Kako se polovi javljaju u

konjugovano kompleksnim parovima, najmanji red je 2. Zbog toga je filtar razložen na bikvadratne filtre (3.3).

Veliki problem prilikom kvantovanja koeficijenata IIR filtra je u tome što se ne može egzaktno, unapred naći odgovarajući broj bita za predstavu celog i razlomnjenog dela. Jedini način je simulacijom, što iziskuje puno vremena i resursa prilikom projektovanja.

3.3.2 Šum usled kvantizacije proizvoda

Kada se pomnože dva binarna broja od kojih je jedan dužine B_1 bita a drugi B_2 bita, njihov proizvod će imati dužinu $(B_1 + B_2)$ bita. Smanjivanje broja bita u proizvodu odbacivanjem manje značajnih bita zove se kvantzacija proizvoda. Za kvantizaciju proizvoda se može primeniti zaokruživanje ili odsecanje (funkcija `resize()`, iz dodatka A.2, radi zaokruživanje). U praksi se češće koristi zaokruživanje jer je u tom slučaju srednja vrednost greške jednaka nuli.

Prema diskusiji iz prethodne glave o kvantizacionom šumu, i ovde je snaga šuma kvantizacije proizvoda,

$$\sigma^2 = \frac{\Delta^2}{12}. \tag{3.7}$$

Da bi se izračunala ukupna snaga kvantizacije na izlazu iz filtra, potrebno je uticaj svakog množača u filtru modelirati prema modelu linearnog kvantizatora iz prethodne glave.

U mnogim slučajevima je hardver koncipiran tako da se operacija množenja realizuje kao množenje i akumulacija, a rezultat se smešta u registar sa dvostrukom dužinom. Ovim načinom je moguće kvantovati izraz nakon svih množenja i sabiranja, čime se greška kvantovanja smanjuje. U VHDL implementaciji je uzeto da svi međurezultati unutar filtra budu na duplo većoj širini reči kao što se može videti u sledećoj sekciji VHDL koda,

```
-- IIR's output, quantizier input and quantization error.
variable y_iir, e            : sfixed(4 downto -19);
variable y_i                 : sfixed(4 downto -19);
-- H0.
variable x0, x0d             : sfixed(4 downto -19);
-- H1.
variable x1, w1, w1d, w1dd   : sfixed(4 downto -19);
-- H2.
variable x2, w2, w2d, w2dd   : sfixed(4 downto -19);
```

Izložena je sekcija koda koja pripada procesu unutar koga je implementirana struktura sa sl. 3.2. Međurezultati su predstavljeni na 24 bita sa 5 bita za ceo deo. Ulazni signal i koeficijenti su predstavljeni na 12 bita. Ceo kod implementacije je dat u dodatku A.2.

3.3.3 Granični ciklusi

Dosadašnja analiza realizacije digitalnih filtara se zasnivala na uprošćenim linearnim modelima. Kvantizacija aritmetičkih operacija sabiranja i množenja, međutim, dovodi do toga da je digitalni filtar u praksi nelinearan sistem. Usled nelinearnosti se može dogoditi da filtar za određene ulazne signale postane nestabilan, iako se polovi filtra nalaze unutar jediničnog kruga. Ta nestabilnost se ogleda kao oscilatorni signal na izlazu koji se naziva granični ciklus (engl. limit cycle). Granični ciklusi se javljaju samo kod IIR filtara zbog njihove rekurzivne prirode [4].

Najznačajnija su dva tipa graničnih ciklusa:

1. Granični ciklusi zbog prekoračenja opsega pri sabiranju.

2. Granični ciklusi zbog zaokružavanja proizvoda.

Granični ciklusi zbog prekoračenja opsega pri sabiranju se neminovno javljaju zbog aritmetike u komplementu dvojke. Prilikom zbira dva broja, koja su po modulu manja od maksimalne dozvoljene vrednosti, rezultat može biti veći od te maksimalne vrednosti. S obzirom da je dinamički opseg u realizaciji filtra ograničen tom vrednošću, rezultat će postati negativan. Ta promena znaka će uneti veliku grešku, koja može upotpunosti narušiti performanse digitalnog filtra.

Rešenje bi bilo ili saturacija, ili povećanje broja bita za rezultat. Saturacija znači da se rezultat, u slučaju da dođe do prekoračenja, ograničava na maksimalnu, tj. minimalnu vrednost. U implementaciji, u ovom radu, je korišćena prva metoda, saturacija. Funkcija `resize()` sama vodi računa o saturaciji.

Granični ciklusi zbog zaokruživanja proizvoda se javljaju kada su vrednosti signala u nekim čvorovima filtra veoma mali, što se dešava kada se završi pobudni signal na ulazu u filtar. Izlaz filtra, kada prestane pobuda, počinje da se smanjuje do nule, ukoliko je filtar stabilan. Ono što se može desiti prilikom zaokruživanja je da izlaz iz filtra ostane konstantan, ili da počne da osciluje sa malom amplitudom. I ovaj granični ciklus se može smanjiti, čak i eliminisati, ukoliko se međurezultatima dodeli duplo veća širina reči. Drugo rešenje bi bilo odsecanje manje značajnih bita, ali se to izbegava zato što taj postupak povećava grešku kvantizacije.

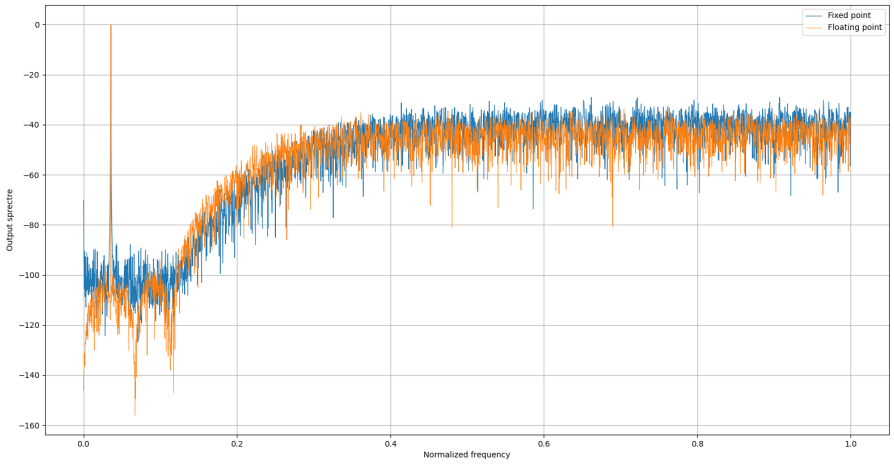
3.4 Rezultati simulacije u VHDL-u

Kako bi se sačuvali i crtali rezultati simulacije, napisan je VHDL test-bench kod (dodatak A.3). Zadata je sinusoida kao ulazni signal,

$$x[n] = 2^{M-1} \sin\left(2\pi \left\lfloor \frac{2}{7} \frac{N}{2\,OSR} \right\rfloor \frac{n}{N}\right), \tag{3.8}$$

gde su $M = 5$ rezolucija kvantizatora, $N = 8192$ broj odbiraka i $OSR = 8$ faktor nadodabiranja. Ulazni signal je kvantovan na 12 bita sa 5 bita za ceo deo. Koeficijenti IIR filtra su kvantovani tako da prioritet bude na razlomljenom delu, tj. kod koeficijenata koji su po aposlutnoj vrednosti manji od 1 svi biti su dodeljeni razlomljenom delu. Međurezultati su kvantovani na 24 bita, sa 5 bita za ceo deo.

Na sl. 3.5 su prikazani spektri izlaznog signala u realizaciji sa fiksnom tačkom i u poktretnom zarezu. Ulazni signal je sinusoida maksimalne amplitude (3.8). Sa slike se može videti da je sinusoida jasno izdvojena na opsegu učestanosti $f \in [0, \frac{1}{8}]$. Takođe se može videti da je, u istom opsegu učestanosti, SNR oko 88 dB.



Slika 3.5: *Spektri izlaznog signala u strukturi sa aritmetikom sa pokretnom i fiksnom tačkom.*

Teorijska vrednost snage kvantizacionog šuma u opsegu od značaja, prema (2.11),

$$IBN = \frac{\Delta^2}{132\pi} \left(\frac{\pi}{8^{11}}\right), \tag{3.9}$$

dok je snaga sinusoide,

$$P_s = 2^6 \cdot \Delta^2, \tag{3.10}$$

pa je teorijski SQNR,

$$SQNR = 10 \log_{10} \frac{P_s}{IBN} = 92\text{dB}. \tag{3.11}$$

Razlika između dobijene i teorijske vrednosti $SQNR$ je oko 4 dB.

Prenosna funkcija u aritmetici sa pokretnom tačkom, sa sl. 3.5, je dobijena korišenjem python funkcije `simulateDSM()` iz paketa `deltasigma` [6].

Visoke frekvencije, na kojima se nalazi kvantizacioni šum, se mogu lako otkloniti analognim filrom na izlazu D/A konvertora kao što je prikazano na blok dijagramu na sl. 2.10.

Glava 4

Zaključak

Delta sigma modulator igra ključnu ulogu u konverziji analognih i digitalnih signala sa viskom rezolucijom. Visoka rezolucija konverzije je potrebna u svim primenama koje koriste prikupljanje, tj. obradu zvučnog signala, radarskog signala itd.

U narednim modifikacijama ovog rada, mogla bi se implementacija u VHDL-u spustiti na FPGA (engl. field programmable gate array). U tom slučaju bi morala da se obrati pažnja na fizička ograničenja hardvera, kao što su različita kašnjenja signala po različitim putanjama, setup i hold vremena. Takođe bi mogao da se napravi algoritam za donekle sistematično određivanje broja bita za ceo i razlomljeni deo.

Sumirajući sve iznešeno u ovom radu kao krajnji zaključak nameće se da ovaj sistem sa svim svojim parametrima daje očekivane rezultate.

Dodatak A

Kodovi

A.1 Koeficijenti IIR filtra

Python kod koji nalazi koeficijente IIR filtra je dat u ovom dodatku. Koristi se biblioteka `deltasigma` [6] koja je prevedena iz programskog paketa matlab.

```
from iir_filter import *
from scipy.signal import zpk2tf
from deltasigma import *

form = 'CRFBD'
f0 = 0.0
Hinf = 4
OSR = 8
M = 4
ORD = 5

H = synthesizeNTF(ORD, OSR, 1, Hinf, f0)
a, g, b, c = realizeNTF(H, form)
ABCD = stuffABCD(a,g,b,c, form)
ABCDs, umax, S = scaleABCD(ABCD, M+1)
[ntfs, stfs] = calculateTF(ABCDs)
ntfs_nd = zpk2tf(ntfs[0], ntfs[1], ntfs[2])
init_printing()
z = symbols('z')
Numerator = Poly(ntfs_nd[0],z)
Denominator = Poly(ntfs_nd[1],z)
NTFS = Numerator/Denominator
```

```
He = simplify(1-NTFS)
```

gde su `ORD` red filtra, `OSR` faktor nadodabiranja, `Hinf` je pojačanje NTF na visokim učestanostima i `f0 = 0` određuje da NTF bude propusnik visokih učestanosti.

Kod uzima u obzir sve navedeno u prethodnim poglavljima o odabiru nula i polova.

A.2 Implementacija delta sigma modulatora

VHDL kod koji implementira strukturu sa sl. 3.2 je dat u ovom dodatku. Fajl `fixed_generic_pkg` je potrebno importovati u projekat kako bi mogle da se koriste funkcije za aritmetiku sa fiksnom tačkom.

```
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.fixed_generic_pkg.all;
--------------------------------------------------------------------------
entity delta_sigma is
    port(
        clk, rst    : in std_logic;
        x           : in sfixed(3 downto -8);
        y           : out sfixed(3 downto 0)
    );
end entity;
--------------------------------------------------------------------------
architecture rti of delta_sigma is
-- Coefficients of H0.
constant b00  : sfixed(3 downto -8) := to_sfixed(7.3765809, 3, -8);
constant a01  : sfixed(0 downto -11) := to_sfixed(0.3466036, 0, -11);
-- Coefficients of H1.
constant b10  : sfixed(0 downto -11) := to_sfixed(0.424071040, 0, -11);
constant b11  : sfixed(2 downto -9) := to_sfixed(2.782608716, 2, -9);
constant a11  : sfixed(0 downto -11) := to_sfixed(0.66591402, 0, -11);
constant a12  : sfixed(0 downto -11) := to_sfixed(0.16260264, 0, -11);
-- Coefficients of H2.
constant b20  : sfixed(3 downto -8) := to_sfixed(4.606822182, 3, -8);
constant b21  : sfixed(0 downto -11) := to_sfixed(0.023331537, 0, -11);
```

```
constant a21  : sfixed(0 downto -11) := to_sfixed(0.62380242, 0, -11);
constant a22  : sfixed(0 downto -11) := to_sfixed(0.4509869, 0, -11);

begin
process(clk, rst)
-- IIR's output, quantizier input and quantization error.
variable y_iir, e      : sfixed(4 downto -19);
variable y_i   : sfixed(4 downto -19);
-- H0.
variable x0, x0d          : sfixed(4 downto -19);
-- H1.
variable x1, w1, w1d, w1dd   : sfixed(4 downto -19);
-- H2.
variable x2, w2, w2d, w2dd   : sfixed(4 downto -19);
-- Output.
variable v               : sfixed(4 downto 0);

begin
if(rst = '1') then
    y_iir   := to_sfixed(0, y_iir);
    y_i     := to_sfixed(0, y_i);
    e       := to_sfixed(0, e);
    x0      := to_sfixed(0, x0);
    x0d     := to_sfixed(0, x0d);
    x1      := to_sfixed(0, x1);
    w1      := to_sfixed(0, w1);
    w1d     := to_sfixed(0, w1d);
    w1dd    := to_sfixed(0, w1dd);
    x2      := to_sfixed(0, x2);
    w2      := to_sfixed(0, w2);
    w2d     := to_sfixed(0, w2d);
    w2dd    := to_sfixed(0, w2dd);
    v       := to_sfixed(0, v);
    tmp     := to_sfixed(0, tmp);
elsif rising_edge(clk) then
    y_i  := resize(x + y_iir, y_i'high, y_i'low);
    v   := resize(y_i, v'high, v'low);
    if(v(v'right) = '0') then
        v  := resize(v + to_sfixed(1, v), v'high, v'low);
    end if;
```

```
    e  := resize(y_i - v, e'high, e'low);

    x0  := resize(b00*e + a01*x0d, x0'high, x0'low);
    w1  := resize(e + a11*w1d - a12*w1dd, w1'high, w1'low);
    x1  := resize(b10*w1 - b11*w1d, x1'high, x1'low);
    w2  := resize(e + a21*w2d - a22*w2dd, w2'high, w2'low);
    x2  := resize(b21*w2d - b20*w2, x2'high, x2'low);

    y_iir  := resize(x0 + x1 + x2, y_iir'high, y_iir'low);

    x0d  := x0;
    w1dd  := w1d;
    w1d  := w1;
    w2dd  := w2d;
    w2d  := w2;
end if;
y <= v;
end process;
end architecture;
```

A.3 Simulacija implementacije

Kod u VHDL za simulaciju implementacije $\Delta\Sigma$ modulatora je dat u ovom dodatku. Potrebno je u promeljivu `x_in` upisati niz od 8192 elemenata ulaznog signala, i potrebno je podesiti putanju izlaznog tekstualnog fajla u koji će se upisati rezultati simulacije.

```
--------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use std.textio.all;
use work.fixed_generic_pkg.all;
--------------------------------------------------------------------------
entity delta_sigma_tst is
end entity;

architecture testbench of delta_sigma_tst is
    constant C_CLK_PERIOD   : time      := 1 ns;
```

```
    signal clk  : std_logic             := '1';
    signal rst  : std_logic             := '1';
    signal x  : sfixed(4 downto -8);    := (others => '0');
    signal y    : sfixed(4 downto 0);

    component delta_sigma_new
        port(
            clk, rst    : in std_logic;
            x           : in sfixed(4 downto -8);
            y           : out sfixed(4 downto 0)
        );
    end component;

    type x_input is array(1023 downto 0) of real;
    constant x_in : x_input := ();

begin
    I1 : delta_sigma_new
        port map(
            clk     => clk,
            rst     => rst,
            x       => x,
            y       => y
        );

    clk <= not clk after C_CLK_PERIOD / 2;

    ALWAYS : process
    begin
        wait for C_CLK_PERIOD;
        rst <= '0';
        wait;
    end process;

    INPUT : process(clk)
        variable i : integer := 0;
    begin
        if(rst = '1') then
            i := 0;
        elsif rising_edge(clk) then
```

```
            x <= to_sfixed(x_in(i), 3, -8);
            i := i + 1;
        end if;
    end process;

    OUTPUT : process(clk)
        file file_pointer   : TEXT OPEN WRITE_MODE is
                                ".\simulator_output.txt";
        variable line_el    : LINE;
        variable y_integer  : INTEGER;
    begin
    if rising_edge(clk) then
        y_integer := to_integer(signed(y));

        write(line_el, y_integer);
        writeline(file_pointer, line_el);
    end if;
    end process;

end architecture;
```

Literatura

[1] Miodrag Popović, *"Signali i sistemi"*, Akademska misao, Beograd, 2006.

[2] Dušan Grujić, *"Hardversko softverska obrada signala"*, 2023.

[3] Shanthi Pavan, Richard Schreier, Gabor C. Temes, *"Understanding Delta-Sigma Data Convertors"*, IEEE Press, 2017.

[4] Ljiljana Milić, Zoran Dobrosavljević, Jelena Ćertić, *"Uvod u digitalnu obradu signala"*, Akademska misao, Beograd, 2015.

[5] A. S. Sedra, K. C. Smit, *"Microelectronic circuits"* 7^{th} edition, Oxford University Press, New York, USA, 2015.

[6] Richard Schreier, *"The Delta-Sigma Toolbox"*, Matlab, 2020.