Case projekt 3 – Audio filter

Baggrund

Man kan ofte have et ønske om at fjerne en enkelt tone/frekvens. Det kan fx. være 50 Hz fra lysnettet, som forstyrrer et målesignal eller det kan være en kraftig forvrængnings-komponent pga. en ikke-linearitet i ens system.

Opgaven

I skal designe, analysere og implementere et 2. ordens IIR filter til at fjerne en enkelt, forstyrrende tone i et audio signal. Filtret er et notch-filter, dvs. det har et hak (notch) ved en enkelt frekvens. Højere ordens IIR filtre vil man ofte implementere som fx. en kaskade af 2.ordens filtre, så det er godt at have styr på 2.ordens filtre.

Proceduren vil altså være som følger :

- Analysere det givne lydsignal og undersøge hvilken frekvens, som ønskes fjernet.
- Designe IIR filtret vha. placering af poler/nuller
- Implementere IIR filtret på Blackfin processor
- Teste funktionen af filtret vha. både tonegenerator/oscilloskop og den givne lydfil

Kom-godt-igang software

I har fået lydfilen med den forstyrrende rentone (”tale_tone_48000.wav”). Desuden ligger der et template projekt til CrossCore miljøet til programmering af Blackfin processoren, som I kan bruge i opgave 3.

Opgave 1 – Analyse af inputsignal

I skal finde frekvensen af den tone, som I ønsker at fjerne. Indlæs lydfilen i fx. Matlab og analyser signalet her - med analyse menes at se på tids-signalet, amplitude-spektrum, spektrogram, ..

Opgave 2 – IIR notch filter design

I skal designe et 2. ordens IIR notch filter vha. pol/nul-punkts placering i z-domænet. Nuller/poler har en radius og vinkel, da de er komplekse tal - fx. $z0 = r \exp(j\omega)$ hvor r er nul-radius og ω er nul-vinkel.

Fra opgave 1 kender I vinkel af de to nuller og poler - husk at nuller/poler altid kommer i par af to, dvs. for hver nul/pol vil der være en tilsvarende med samme radius, men med negativ fortegn på vinklen (den kompleks konjugerede). Nul-radius kan I sætte til 1 og pol-radius kan I sætte til fx. 0,9.

Start med at eksperimentere i Micromodeler (http://www.micromodeler.com/dsp/) eller Matlab ved at kalde funktionen ”fdatool”. I Fdatool (=filter design and analysis tool) skal I vælge ”pol-nulpunkts-analyse” (ikon i nedre venstre bjælke). Her kan I indsætte poler/nuller og se effekten i fx. frekvens-responset - det kan give en god intuitiv forståelse. Prøv at flytte pol-radius til andet end 0,9 - fx. 0,7 eller 0,999. Hvad er betydningen af pol-radius ?

1. Skitser pol-nul-punkts diagram for filtret
2. Udregn systemfunktionen H(z) og opskriv differensligningen herudfra. OBS: DETTE SKAL I GØRE MANUELT, DVS. "I HÅNDEN" OG BESKRIVES I RAPPORTEN.. I MÅ IKKE BLOT AFLÆSE FRA MATLAB !
3. Tegn signalgrafen for filtret på direkte form I.
4. Bestem og plot frekvens-responset $|H(e^{j\omega})|$ for filtret udfra H(z) - Er det et ideelt filter ? (hvorfor/hvorfor ikke)
5. Test at filtret virker ved at filtrere lydfilen i Matlab (I kan benytte ”filter” kommando i Matlab)

Opgave 3 – Implementation på Blackfin processor

I kan tage udgangspunkt i template-projektet (”AudioNotchFilter.zip”). I skal forbinde en lydkilde (PC eller tonegenerator) til kittets input og en output kanal (fx. et par hovedtelefoner eller højttaler) til output stikket. (OBS: kittet har minijack stereo inputs og outputs - det står med småt på printet. J1 = line in og J2 = head phones out).

Opgaven består herefter i at få omsat differensligningen fra opgave 2 til et fungerende filter i C-kode. Template-projektet er interrupt-styret, dvs. hver gang at der samples (dvs. med en periode på 1/fs hvor fs = 48kHz) dannes et interrupt og herefter kaldes funktionen ”AudioNotchFilter” i ”AudioNotchFilter.c”. Bemærk at først kopiers samples til et array for henholdsvis venstre og højre kanal. I kan simpelthen skrive jeres kode ind i denne funktion.

Template projektet vil som udgangspunkt sende lyden direkte igennem - i " AudioNotchFilter " svarer dette til "PASS_THROUGH" case. Hvis I trykker på "SW2"-knappen på kittet vil i skifte over til "IIR_FILTER_ACTIVE" casen istedet - på den måde kan I høre virkningen af jeres filter.

De basale trin i interrupt-funktionen skal være (for jeres filter):

1. Læs den nye sample fra ADC-register og gem i en variabel - husk at i ”matematik-sprog” er denne sample x(n). Læsningen fra ADC sker udenfor " AudioNotchFilter " og værdien er skrevet til variablen "dataIn" (for left, right samples).
2. Beregn det nye output y(n) udfra x(n), x(n-1), .. samt y(n-1), y(n-2), .. og filter-koefficienterne. Og skriv output til DAC-register - dvs. I skal lægge output værdien i variablen "outLeft". OBS: outLeft og outRight er 32-bit variable med værdien skrevet i de 24 mindst betydende bits.
3. Opdater den såkaldte delay-line - dvs. næste gang der kommer et interrupt, vil den nuværende x(n) være blevet til x(n-1), nuværende x(n-1) blevet til x(n-2) etc.. dvs. opdateringen består i at sætte ”x1 = xn” hvis x1 er variablen for x(n-1). Tilsvarende gøres for y-værdierne - filtret kaldes netop rekursivt, da det nye output y(n) ”føres tilbage” og bruges igen næste gang.

I udgangspunktet er y(n) = x(n)/2 i templaten.

Hints :

- I skal benytte datatypen ”short” til variablene, da I arbejder på en fixed-point processor.
- Start med blot at teste standard-indstillingen y(n) = x(n), dvs. intet filter. Så kan I checke at opstillingen er o.k. - bemærk, I kan godt forvente at der er en lidt hørbar støj på output.
- Det er en meget god ide at lave debugging af koden. Det kunne f.eks. være ved at sætte et breakpoint, single-steppe igennem og holde øje med variabel-værdierne løbende. På den måde kan I checke om I har den rigtige sammenhæng imellem output-værdien y(n) og input-værdierne x(n), x(n-1), ..

Opgave 4 – Test af filtret

Til sidst testes filtret (og evt. debugges hvis det ikke virker..). I kan teste filtret på flere forskellige måder :

- Tonegenerator og oscilloskop i elektronik-lab.
- Afspilning af rentoner i Matlab, som sendes ind i kittet.
- Afspilning af frekvens-sweep (”chirp”-kommando) i Matlab, som sendes ind i kittet.
- Med den ”forurenede” lydfil.

I bør som minimum afprøve med tonegenerator/oscilloskop og lydfilen.

Hvis man skal være lidt mere grundig, bør man reelt set optage outputtet i forhold til inputtet, når man tester. På den måde laves en frekvens-karakteristik af systemet. Så kan karakteristikken sammenholdes med den beregnede i opgave 2.