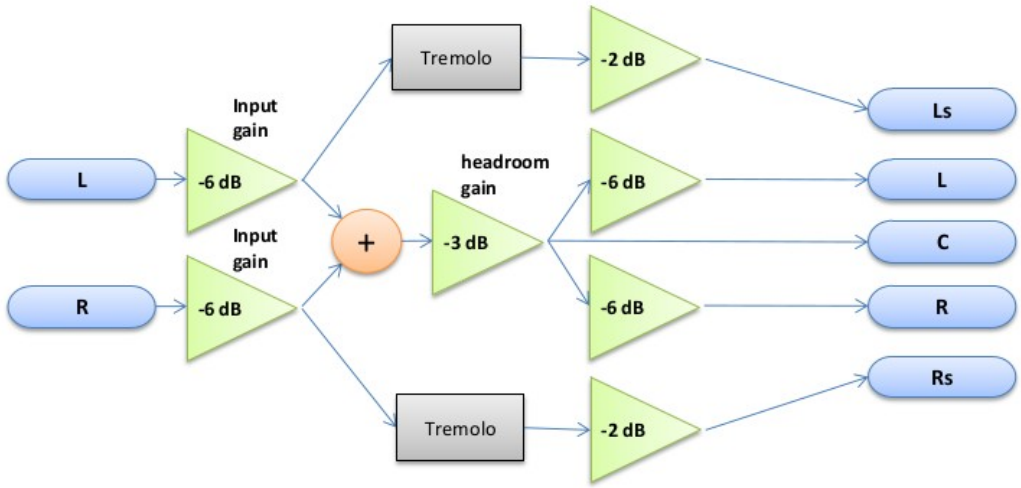
Algoritmi i arhitekture DSP-a 2

Projekat: Realizacija algoritma kombinovanja kanala na Cirrus Logic DSP platformi

Bojan Uljarević RA15-2017

Opis zadatka:

Cilj zadatka je realizovati kombinovanje kanala na osnovu date šeme i tabele. Projekat se realizuje kroz 4 modela urađena u skladu sa metodologijom predloženom na vježbama.



control	Enable	Input gain	Headroom gain	Output Mode
values	On/Off	From 0 to -∞ dB	From 0 to -∞ dB	2_0_0, 2_0_0, 0_2_0, 0_2_0, 3_2_0
default value	On	-6 dB	-3 dB	2_0_0

Tabela 1 – Korisničke kontrole

Model 0:

Model 0 predstavlja referentni kod na osnovu koga se vrše postepene modifikacije da bi program funkcionisao na namjenjenoj platformi. U skladu sa zadatim zadatkom, unesene su sljedeće modifikacije u priloženi Visual Studio projekat:

- Dodata je datoteka *tremolo.cpp* uz odgovarajuće zaglavlje. U ovoj datoteci se nalazi funkcija za primjenu tremolo efekta na ulazni niz kao I pomoćne funkcije.
- Dodata je funkcija *processing* u datoteku *main.cpp* koja primjenjuje pojačanja I efekte u skladu sa zadatim režimom.

- Dodato je zaglavlje *common.h* unutar koga se nalaze makro definicje za jasnije rukovanje kanalima, režimima, te pretpostavljene vrijednosti ulaznih parametara. Takođe je definisana funkcija *db2gain* za konvertovanje decibela u linearno pojačanje .
- Izmjene unutar funkcije *main* se sastoje iz:
    a) provjere režima i postavljanje odgovarajućeg broja kanala u zaglavlje izlazne *wav* datoteke.
    b) dodavanja promjenjive *offset* koja se u slučaju režima 0_2_0 postavlja na vrijednost 3 da bi se na izlazu upisivalo u kanale LS i RS umjesto kanale L i R.

Model 1:

Model 1 predstavlja funcionalno poboljšanje modela 0 i prevodi se na istom prevodiocu. Modifikacije koje su izvršene u ovom koraku su:

- Preimenovanje *int* I *double* tipova u *DSPint* I *DSPfract*.
- Izmiještanje izlaznih bafera iz *processing* funkcije u globalni opseg radi manjeg opterećenja steka.
- Smanjenje broja argumenata funkcije *processBlock* unutar *tremolo.cpp* datoteke.
- Uvođenje pokazivača za brže iteriranje kroz ulazno-izlazni bafer *sampleBuffer* i pomoćni *tremoloBuffer*.
- Pokazivači na *sampleBuffer* su objedinjeni unutar globalne strukture *samples*.

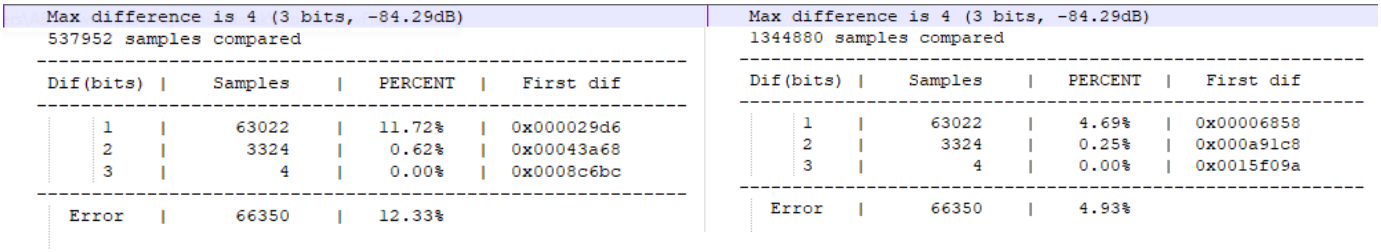
Model 2:

Model 2 se prevodi na istom kompjaleru kao i prva dva modela, uz razliku da se koristi aritmetika u nepokretnom zarezu. Izvršene izmjene u ovom modelu su:

- Ubačeno je zaglavlje *stdfix_emu.h* za emulaciju aritmetike u nepokretnom zarezu.
- Tip *DSPfract* zamijenjen je klasom *fract*.
- Ulazno-izlazni bafer se inicijalizuje pomoću direktive *FRACT_NUM* umjesto funkcije *memset*.

- Vrijednosti su skalirane u skladu sa aritmetikom nepokretnog zareza.

Izlaz iz ovog modela je bit-identičan sa prethodnim modelima u režimu 2_0_0 (bez tremolo efekta), a u suprotnom greška na nivou bita se primjećuje u 4-12% odbiraka pri čemu nikada nije veća od 4 bita po odbirku.



Max difference is 4 (3 bits, -84.29dB)
537952 samples compared

Dif(bits)	Samples	PERCENT	First dif
1	63022	11.72%	0x000029d6
2	3324	0.62%	0x00043a68
3	4	0.00%	0x0008c6bc
Error	66350	12.33%	

Max difference is 4 (3 bits, -84.29dB)
1344880 samples compared

Dif(bits)	Samples	PERCENT	First dif
1	63022	4.69%	0x00006858
2	3324	0.25%	0x000a91c8
3	4	0.00%	0x0015f09a
Error	66350	4.93%	

Rezultati PCMCompare alata za režime 0_2_0 I 3_2_0

Model 3:

Funkcionalnost modela 3 nije postignuta u priloženom projektu.

Model 3 je realizovan unutar *CLIDE* okruženja i treba predstavljati kod u formatu u kome se spušta na ciljnu platformu. Posebna pažnja pri formiranju ovog modela se posvećuje profilisanju koda, kao i optimizaciji prevođenja kao i direktnom upotrebom asemblerskog jezika. Izlaz iz ovog modela treba biti identičan izlazu modela 2.

Optimizacije za prevođenje CCC2 prevodiocem:

Ovaj tip optimizacije u datom projektu se svodi na sledeće:

- Eksplicitno naglašavanje memorijskih zona globalnih promjenjivih.
- Podjela processing funkcije na podfunkcije *load_tremolo* koja puni *tremoloBuffer* iz ulaznog niza I transformiše ga, i *calculate_gain* koja primjenjuje zadata pojačanja.
- Hardverske petlje su primijenjene na svim mogućim mjestima, izuzetak je petlja za iteraciju kroz izlazne kanala, zato što njihov broj zavisi od režima.

- Optimizacija uslovnih petlji (a > b u a – b > 0) je izvršena na mjestima gdje asembler vrši nepotrebne provjere izlaza iz opsega. Ubrzanje nije zabilježeno.
- Optimizacija uslovnog iskaza koji čini tijelo funkcije *lfo* (niskofrekventni oscilator) u *tremolo.h* datoteci nije pokazala rezultate.

Optimizacije upotrebom asemblerskog jezika:

U ovoj fazi funkcije *load_tremolo* I *calculate_gain* su realizovane u zasebnim asemblerskim datotekama. Polazna tačka su bile funkcije unutar generisanog izlaznog asemblerskog koda. Instrukcije u asemblerskom formatu su reorganizovane tako da se vrši što manje bespotrebnih *nop* instrukcija, suvišnih provjera i uslovnih skokova. Ovom fazom optimizacije je postignuta značajna redukcija broja ciklusa unutar pomenutih funkcija. Rezervisana programska memorija je smanjena za 20 adresa. U memorijama za podatke nema značajnijih promjena. Podaci prikupljeni tokom profilisanja različitih faza modela 3 se mogu vidjeti u priloženoj tabeli.

		bez optimizacije	**kompajlerske opt.**	**asemblerske opt.**
2_0_0	load_tremolo	16	16	16
	calculate_gain	944	896	857
0_2_0	load_tremolo	581	580	560
	calculate_gain	944	896	857

Prosječan broj instrukcija po izvršenju funkcije