



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У  
НОВОМ САДУ



Драган Бркин

# **ИМПЛЕМЕНТАЦИЈА БЛОКА АУДИО ОБРАДЕ ЗА ОГРАНИЧАВАЊЕ НИВОА ИЗЛАЗНОГ АУДИО СИГНАЛА НА ДСП ПЛАТФОРМИ**

**ДИПЛОМСКИ РАД**  
- Основне академске студије -

Нови Сад, 2016.



**КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА**

Редни број, <b>РБР:</b>		
Идентификациони број, <b>ИБР:</b>		
Тип документације, <b>ТД:</b>	Монографска документација	
Тип записа, <b>ТЗ:</b>	Текстуални штампани материјал	
Врста рада, <b>ВР:</b>	Завршни (Bachelor) рад	
Аутор, <b>АУ:</b>	<b>Драган Бркин</b>	
Ментор, <b>МН:</b>	<b>Доц. др Јелена Ковачевић</b>	
Наслов рада, <b>НР:</b>	<b>Имплементација блока аудио обраде за ограничавање нивоа излазног аудио сигнала на ДСП платформи</b>	
Језик публикације, <b>ЈП:</b>	Српски / латиница	
Језик извода, <b>ЈИ:</b>	Српски	
Земља публиковања, <b>ЗП:</b>	Република Србија	
Уже географско подручје, <b>УГП:</b>	Војводина	
Година, <b>ГО:</b>	<b>2016</b>	
Издавач, <b>ИЗ:</b>	Ауторски репринт	
Место и адреса, <b>МА:</b>	Нови Сад; Трг Доситеја Обрадовића 6	
Физички опис рада, <b>ФО:</b> (поглавља/страна/ цитата/табела/слика/графика/прилога)	<b>7/31/0/4/19/0/0</b>	
Научна област, <b>НО:</b>	Електротехника и рачунарство	
Научна дисциплина, <b>НД:</b>	Рачунарска техника	
Предметна одредница/Кључне речи, <b>ПО:</b>	<b>Дигитална обрада сигнала, Ограничавање нивоа излазног аудио сигнала</b>	
<b>УДК</b>		
Чува се, <b>ЧУ:</b>	У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, <b>ВН:</b>		
Извод, <b>ИЗ:</b>	У оквиру рада имплементирано је једно решење ДСП модула за ограничавање нивоа излазног аудио сигнала на ДСП процесору фирме Cirrus Logic.	
Датум прихватања теме, <b>ДП:</b>	<b>30. јун 2016.</b>	
Датум одбране, <b>ДО:</b>	7. јул 2016.	
Чланови комисије, <b>КО:</b>	Председник: <b>Доц. др Миодраг Ђукић</b>	
	Члан: <b>Доц. др Иван Каштелан</b>	Потпис ментора
	Члан, ментор: <b>Доц. др Јелена Ковачевић</b>	



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO</b> :	
Identification number, <b>INO</b> :	
Document type, <b>DT</b> :	Monographic publication
Type of record, <b>TR</b> :	Textual printed material
Contents code, <b>CC</b> :	Bachelor Thesis
Author, <b>AU</b> :	<b>Dragan Brkin</b>
Mentor, <b>MN</b> :	<b>PhD Jelena Kovačević</b>
Title, <b>TI</b> :	<b>Implementation of audio processing block for limiting output level audio signal on DSP platform</b>
Language of text, <b>LT</b> :	Serbian
Language of abstract, <b>LA</b> :	Serbian
Country of publication, <b>CP</b> :	Republic of Serbia
Locality of publication, <b>LP</b> :	Vojvodina
Publication year, <b>PY</b> :	<b>2016</b>
Publisher, <b>PB</b> :	Author's reprint
Publication place, <b>PP</b> :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, <b>PD</b> : (chapters/pages/ref./tables/pictures/graphs/appendixes)	<b>7/31/0/4/19/0/0</b>
Scientific field, <b>SF</b> :	Electrical Engineering
Scientific discipline, <b>SD</b> :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, <b>S/KW</b> :	<b>Digital signal processing, Output level audio signal limiting</b>
<b>UC</b>	
Holding data, <b>HD</b> :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, <b>N</b> :	
Abstract, <b>AB</b> :	This paper describes implementation of solution for limiting output level audio signal for Cirrus Logic digital signal processor.
Accepted by the Scientific Board on, <b>ASB</b> :	<b>30. jun 2016.</b>
Defended on, <b>DE</b> :	7. jul 2016.
Defended Board, <b>DB</b> :	President: <b>PhD Miodrag Đukić</b>
	Member: <b>PhD Ivan Kaštelan</b>
	Member, Mentor: <b>PhD Jelena Kovačević</b>
	Mentor's sign

## **Zahvalnost**

Zahvaljujem se porodici i prijateljima na pruženoj podršci tokom studiranja.

Zahvaljujem se i DSP timu RT-RK instituta na profesionalnoj podršci prilikom izrade ovog rada.

## SADRŽAJ

1. Uvod.....	1
2. Teorijske osnove .....	2
2.1 Soundbar sistemi .....	2
2.2 Audio signal i njegova snaga .....	5
2.3 Osnovne karakteristike procesora CS49XXX.....	5
2.4 Razvojno okruženje CLIDE .....	9
2.5 Programsko okruženje (framework).....	10
3. Koncept rešenja.....	14
3.1 Kratak opis algoritma .....	14
3.2 Tok razvoja implementiranog rešenja .....	15
3.2.1 Realizacija referentnog modula .....	16
3.2.2 Implementacija modula za ciljnu platformu .....	16
3.2.3 Optimizacija implementiranog modula .....	17
4. Implementacija algoritma za ciljnu platformu .....	18
4.1 Model 0 .....	18
4.2 Model 1 .....	18
4.2.1 LimiterOsInterface.....	19
4.2.2 LimiterDataStruct .....	19
4.2.3 LimiterProcessing .....	20
4.2.4 LimiterDataVars i limiterGlobalConst .....	20
4.3 Model 2 .....	20
4.4 Iskorišćenost resursa ciljne platforme .....	21
5. Ispitivanje.....	23

---

5.1	Slušni testovi .....	23
5.2	Analiza signala u vremenskom domenu.....	24
5.3	Bit identični testovi .....	26
5.3.1	Automatski bit identični testovi.....	27
6.	Zaključak .....	29
7.	Literatura.....	31

## SPISAK SLIKA

Slika 2.1 5.1 Sistem kućnog bioskopa .....	2
Slika 2.2 7.1 sistem kućnog bioskopa .....	3
Slika 2.3 Ilustracija soundbar sistema .....	4
Slika 2.4 Ilustracija dobijanja surround efekta.....	4
Slika 2.5 Pozicija limitera jednom primeru obrade audio signala .....	5
Slika 2.6 CS49XXX procesor .....	6
Slika 2.7 Blok dijagram Crystal 32 DSP arhitekture .....	7
Slika 2.8 Blok dijagram Crystal 32 jezgra .....	8
Slika 2.9 Izgled razvojnog okruženja CLIDE .....	9
Slika 2.10 Cirrus Logic programsko okruženje .....	10
Slika 2.11 Blok dijagram sprege modula sa operativnim sistemom .....	11
Slika 3.1 Blok dijagram algoritma .....	14
Slika 3.2 Pojačanje signala male amplitude .....	15
Slika 3.3 Signal pre i posle limitera .....	15
Slika 5.1 Jasan prikazi greške generisane pri implementaciji.....	24
Slika 5.2 Statistika signala dobijena oduzimanjem izlaza .....	25
Slika 5.3 Poređenje izlaza iz referentnog modela i modela implementiranog u assembleru	26
Slika 5.4 Rezultati poređenja izlaza generisanih za maksimalan broj kanala.....	27
Slika 5.5 Rezultat jednog od testiranja.....	28

**SPISAK TABELA**

Tabela 2.1 Raspoloživa memorija Athena CS497004 procesora.....	8
Tabela 4.1 Iskorišćenost resursa ciljne platforme .....	21
Tabela 4.2 Potrošnja procesorskog vremena po podfunkcijama algoritma .....	22
Tabela 4.3 Potrošnja procesorskog vremena u zavisnosti od broja kanala ulaznog signala	22

## SKRAĆENICE

<b>DSP</b>	- Digital Signal Processing, Digitalna obrada signala
<b>MIPS</b>	- Million Instruction Per Second, milion instrukcija u sekundi
<b>MAC</b>	- Multiply And Accumulate, Množenje i dodavanje
<b>SRS</b>	- Shifter/Rounder/Saturator, Pomeranje, zaokruživanje, saturacija
<b>ALU</b>	- Arithmetic and Logical Unit, Aritmetičko logička jedinica
<b>CLIDE</b>	- Cirrus Logic Integrated Development Enviroment, Cirrus Logic integrisano razvojno okruženje
<b>OS</b>	- Operating System, Operativni sistem
<b>I/O</b>	- Input/Output – U/I Ulaz/izlaz
<b>ODT</b>	- Overlay Definition Table,
<b>MIF</b>	- Module Interface, Sprežni podsistem
<b>MCT</b>	- Module Call Table, Tabela poziva modula
<b>MCV</b>	- Module Control Vector,
<b>PC</b>	- Personal Computer, Personalni računar
<b>AVR</b>	- Audio/Video Receiver, Audio video prijemnik
<b>DMA</b>	- Direct Memory Access, Direktan pristup memoriji
<b>DTS</b>	- Digital Theater Systems,
<b>AC3</b>	- Audio Compression – 3, Audio kompresija - 3

## 1. Uvod

Tema rada je implementacija bloka audio obrade za ograničavanje nivoa izlaznog audio signala (*peak limiter*) na DSP procesoru firme Cirrus Logic za sound-bar sisteme. Rad obuhvata upoznavanje sa osnovama audio DSP obrade i upoznavanje sa ciljnom platformom.

Implementacija obuhvata prilagođenje referentnog C koda aritmetici ciljne DSP platforme i realizaciju u asemblerskom jeziku. Cilj rada je upoznavanje sa pisanjem programske podrške u realnom vremenu za audio DSP aplikacije na ciljnoj platformi. Zadatak se oslanja na rad u programskim jezicima C i Cirrus Logic assembler.

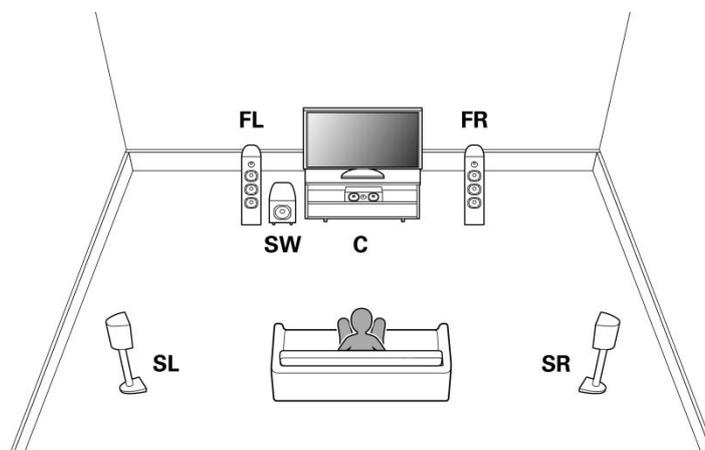
Rad se sastoji iz sedam poglavlja. U drugom poglavlju opisane su teorijske osnove, kao i ciljna platforma. Treće poglavlje predstavlja koncept rešenja, dok je implementacija opisana u četvrtom poglavlju. Peto poglavlje predstavlja opis načina ispitivanja, kao i prikaz rezultata datih testova. Šesto poglavlje sadrži zaključak celokupnog rada. Sedmo poglavlje sadrži literaturu korišćenu pri samoj izradi ovog rada.

## 2. Teorijske osnove

U datom poglavlju opisane su teorijske osnove na kojima je rad zasnovan. Takođe su opisane karakteristike same platforme, razvojno okruženje CLIDE kao i programsko okruženje.

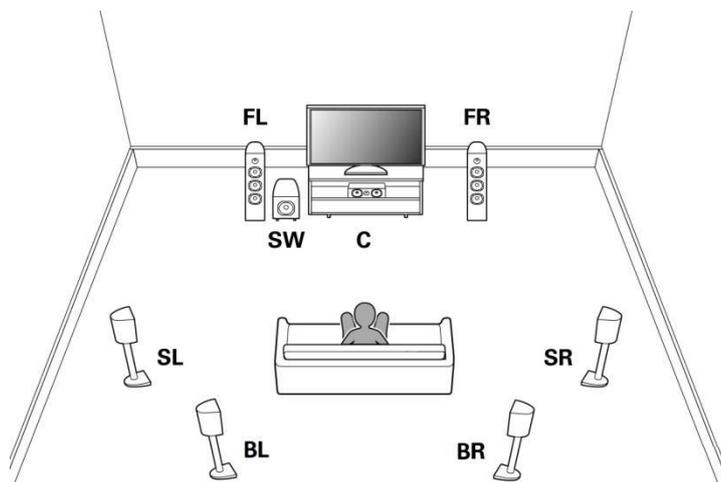
### 2.1 Soundbar sistemi

Sistemi kućnog bioskopa koji su najviše prisutni su takozvani 5.1 sistemi. Broj pet označava broj kanala, odnosno zvučnika punog opsega (eng. Full Range) koji se nalaze na određenim pozicijama oko slušaoca, tj posmatrača, dok broj jedan pripada zvučniku ograničenog opsega (eng. Band-Limited). U najvećem broju slučajeva to je bas zvučnik (eng. Subwoofer) koji reprodukuje niže frekvencije, kao što su eksplozije ili drugi nisko frekventni efekti.



Slika 2.1 5.1 Sistem kućnog bioskopa

Postoji i 7.1 sistem, koji se u odnosu na prethodni navedeni razlikuje po tome što su dodata još dva zvučnika punog opsega, tako da ovakav sistem koristi po dva kanala, zvučnika, sa strane i dva pozadinska kanala radi boljeg surround efekta.



Slika 2.2 7.1 sistem kućnog bioskopa

Poslednji trend u 3D audio, kao što je Dolby Atmos, dodaje još jedan zvučni element koji pruža još impresivnije audio iskustvo.

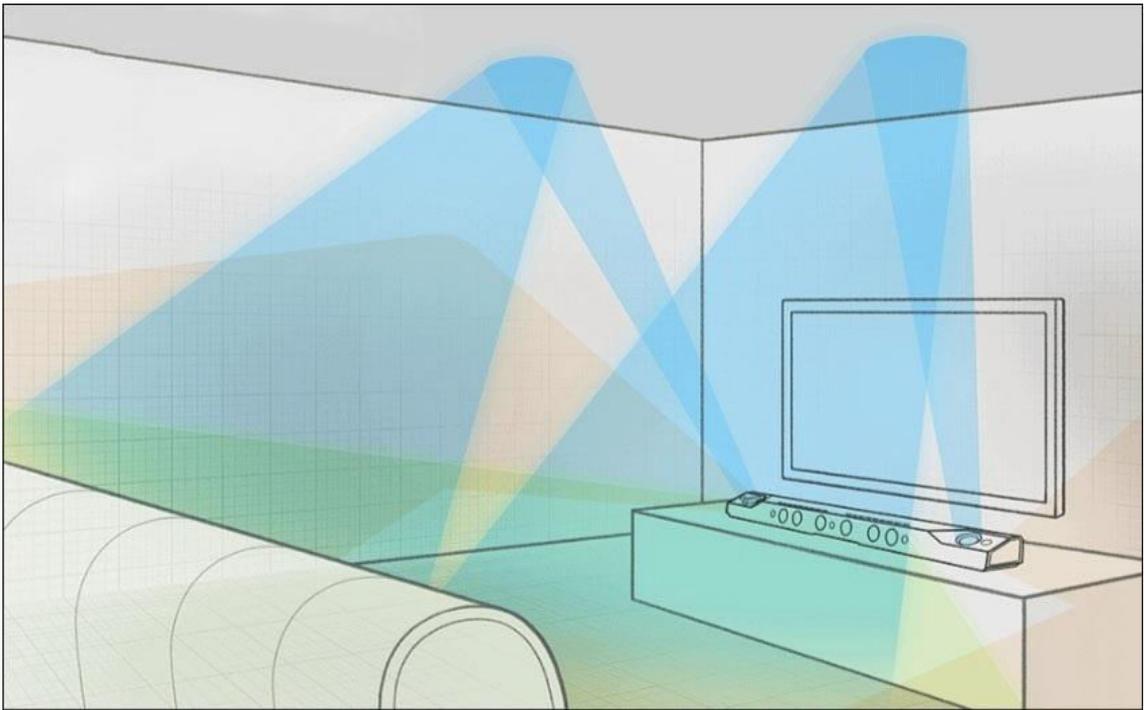
Ukoliko se gradi prostorija namenjena za kućni bioskop, tada postoji veća sloboda u izboru željenih zvučnika, bilo po pitanju veličine ili estetike, kao i njihovog pozicioniranja na idealno mesto u prostoriji da bi se dobile najbolje performanse.

Ukoliko ne postoji mogućnost da se praktično integriše višekanalni sistem u prostoriju, soundbar je veoma popularno rešenje.

Kao što sam naziv govori, ovaj zvučnik izgleda kao jedna šipka, zbog dizajna i iz razloga same upotrebe, kako bi se lakše postavio ispod ili iznad ekrana računara, TV uređaja ili kućnog bioskopa.

Soundbar je specijalna vrsta kućišta zvučnika koja od jedne tačke izvora, odnosno kućišta zvučnika, stvara surround efekat.

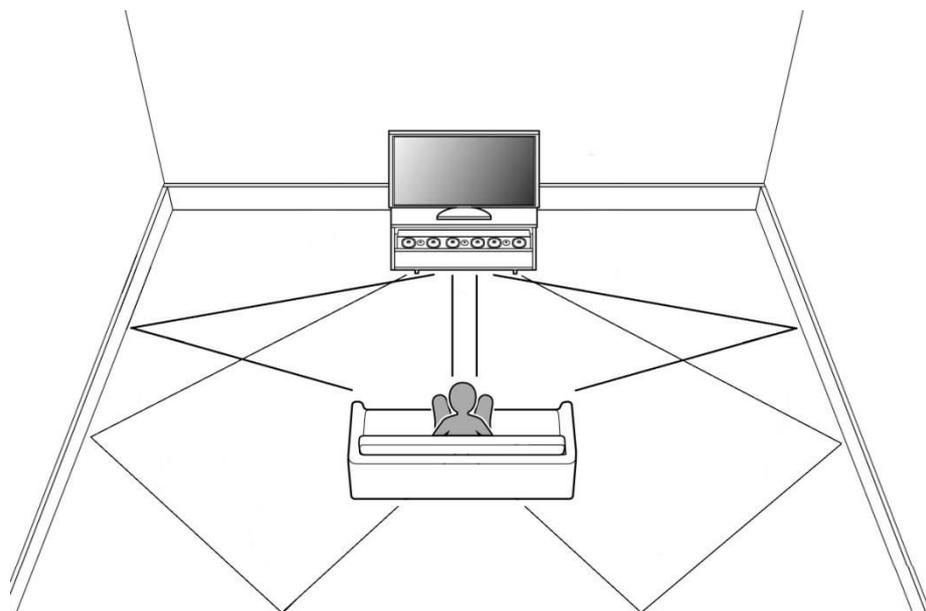
Unutar kućišta nalaze se zvučnici punog opsega, koji su pozicionirani na određen način, tako da emitovanjem audio signala i njegovim odbijanjem o zid prostorije dolazi do slušaoca, stvarajući surround efekat. Ilustracija principa rada jednog soundbar uređaja prikazana je na slici 2.3.



Slika 2.3 Ilustracija soundbar sistema

Kao što se sa slike može uočiti, da bi se dobio efekat levog i desnog surround zvučnika, audio signal se odbija o bočne zidove, dok signal odbijen od plafona stvara efekat pozadinskih zvučnika (eng. LB, RB - Left Back, Right Back).

Postoje i izvedbe sound bar uređaja (slika 2.4) koji surround efekat stvaraju reflektovanjem audio signala o zid prostorije, postavljanjem zvučnika punog opsega pod različitim uglovima unutar kućišta.



Slika 2.4 Ilustracija dobijanja surround efekta

## 2.2 Audio signal i njegova snaga

Pojačavači su ograničeni signalom koji mogu da pojačaju, dok su zvučnici ograničeni signalom koji mogu da pretvore u zvuk bez distorzije audio signala ili fizičkog oštećenja zvučnika.

Kako bi se zvučnici zaštitili, potrebno je omogućiti ograničavanje signala iz izlaznog modula audio obrade. Najjednostavnija tehnika predstavlja jednostavno odsecanje signala na najvišu dozvoljenu vrednost (eng. Clipping). Međutim, ovakvo ograničenje štiti zvučnik od fizičkog oštećenja, ali dovodi do velikih distorzija izlaznog signala. Da bi se izbegle ovakve promene izlaza u odnosu na ulaz, uvodi se limiter koji vrši obradu izlaznog signala u obliku dinamičke kompresije.

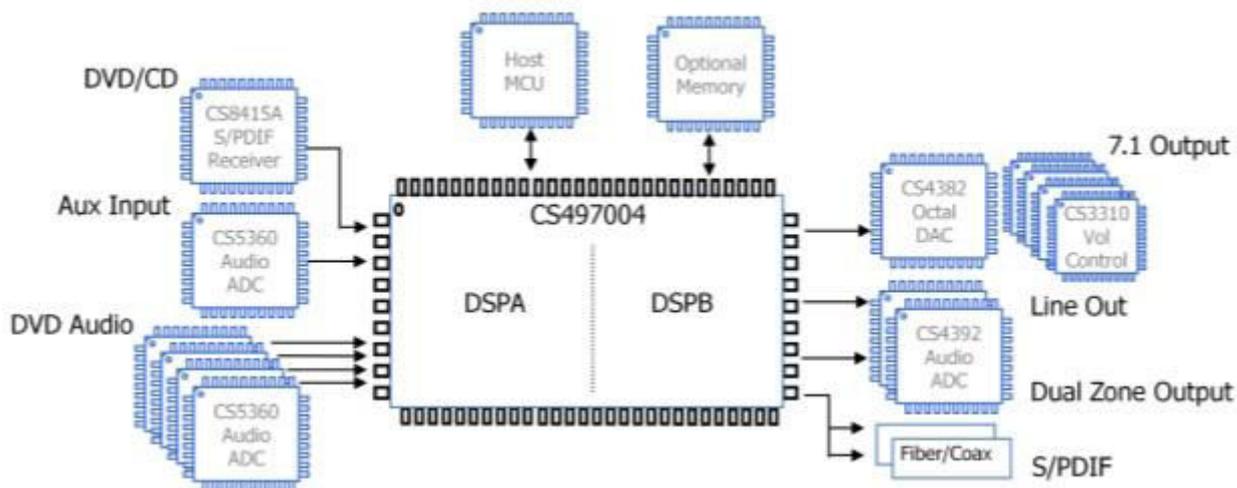
Zadatak ovakvog limitera je da omogući da izlazni signal bude isti kao i ulazni, s tim da određene vrednosti signala koje prelaze unapred zadate vrednosti, budu skalirane tako da uneta promena bude svedena na minimum. Mesto limitera u jednoj digitalnoj obradi signala, prikazano je na slici 2.5



Slika 2.5 Pozicija limitera jednom primeru obrade audio signala

## 2.3 Osnovne karakteristike procesora CS49XXX

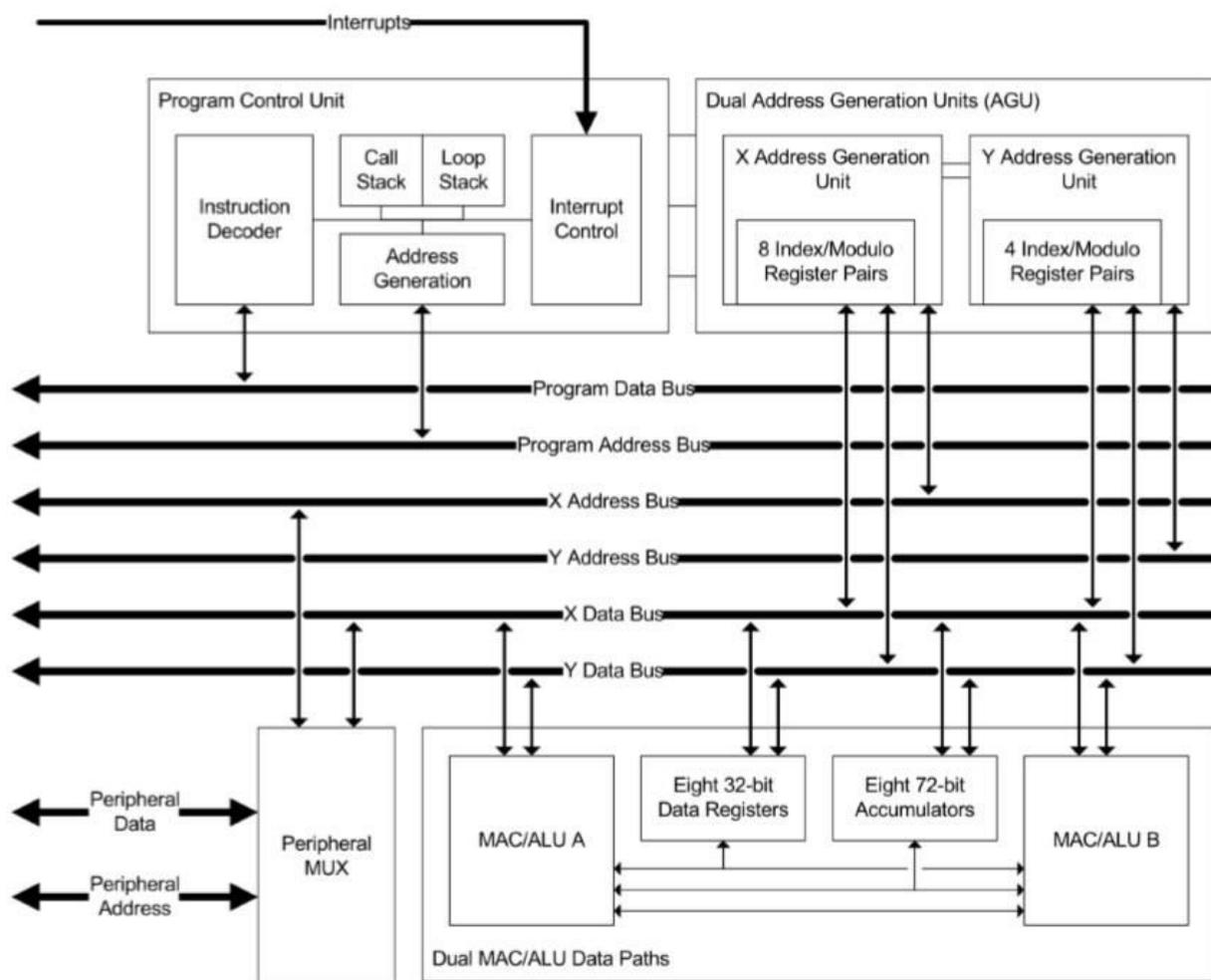
DSP procesori familije CS49xxx kompanije Cirrus Logic su namenjeni obradi audio signala u uređajima potrošačke elektronike. Prvenstveno se ugrađuju u audio/video prijemnike (eng. AVR – Audio/Video Receiver) i aktivne zvučnike, ali imaju i primenu u televizorima, automobilskim audio sistemima i prenosnim uređajima. Ovi DSP procesori imaju ulogu audio pod-procesora (eng. Coprocessor) uređaja. Tipičan sistem je prikazan na slici 2.5. Pripadnici CS49xxx familije DSP procesora su zasnovani na istoj arhitekturi DSP jezgra pod imenom Crystal 32, a razlike između modela unutar familije se odnose prvenstveno na broj jezgara, radni takt i količinu interne memorije.



Slika 2.6 CS49XXX procesor

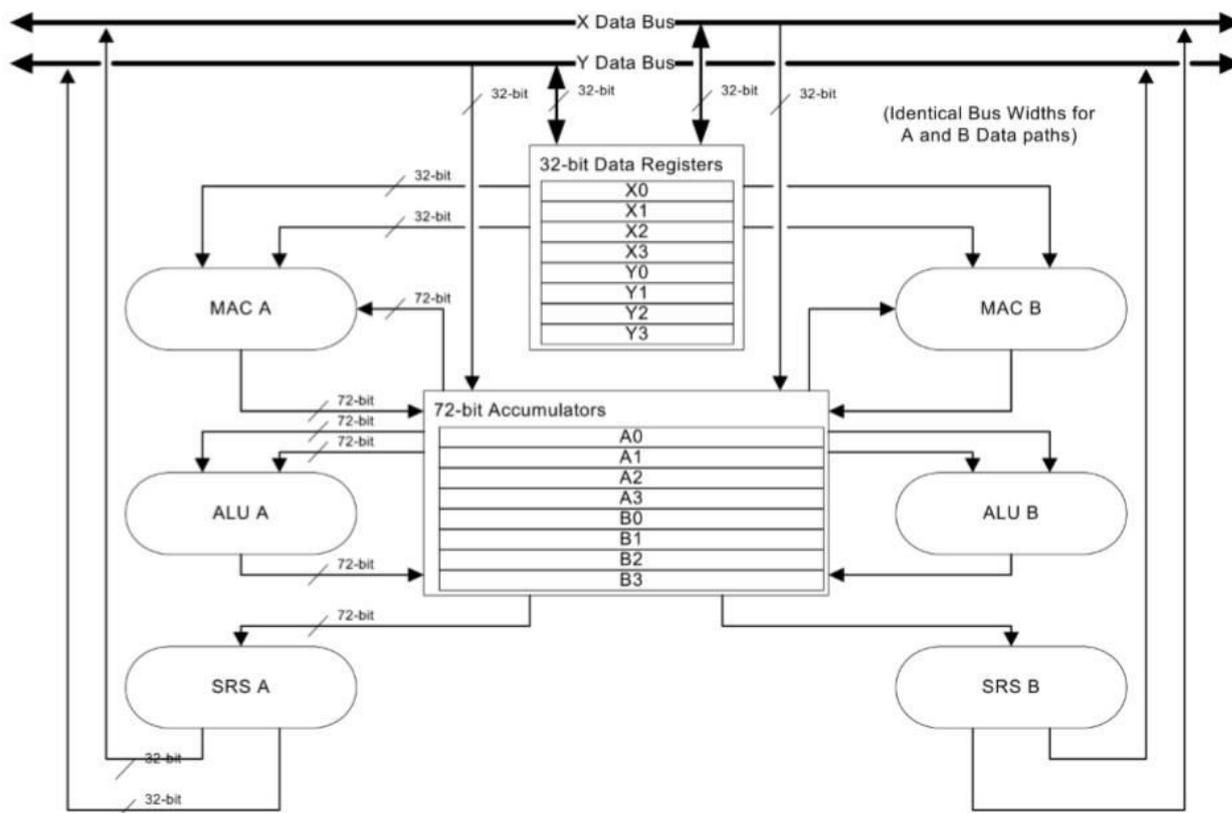
DSP CS49xxx je DSP procesor sa dva jezgra, od kojih svako jezgro ima dve odvojene memorije za podatke, X i Y, i programsku memoriju (slika 2.8 i tabela 2.1). Svako jezgro predstavlja tridesetdvobitni DSP koji radi sa aritmetikom u fiksnom zarezu i koji u sebi sadrži osam 72-bitnih akumulatora, četiri X i četiri Y registra za podatke i dvanaest indeks registara od kojih svaki ima odgovarajući modulo registar. Sprega između jezgara i spoljašnje memorije ostvarena je upotrebom kontrolera za direktan pristup memoriji (eng. Direct Memory Access, DMA).

Jezgro Crystal procesora, zasnovano na aritmetici u nepokretnom zarezu, predstavlja programabilan procesor čije su visoke performanse ostvarene kroz visok stepen paralelizma. Koristi predstavu razlomljenih brojeva u drugom komplementu i sadrži magistrale za dva odvojena memorijska i jedan programski memorijski prostor. Blok dijagram arhitekture prikazan je na slici 2.7.



Slika 2.7 Blok dijagram Crystal 32 DSP arhitekture

Blok dijagram jezgra je prikazan na slici 2.8. Jezgro sadrži jedinicu za kontrolu toka programa, paralelne jedinice za generisanje adresa (AGU) i paralelne tokove podataka. Jedinice za generisanje adresa sadrže dvanaest 16-bitnih indeksnih registara za čuvanje adresa i dvanaest modulo 16-bitnih registara koji rade u sprezi sa indeksnim registrima u cilju obezbeđivanja različitih režima adresiranja. Svaka putanja podataka ima četiri 32-bitna registra opšte namene i četiri 72-bitna akumulatorska registra. Akumulatori se sastoje od 3 podregistra, pri čemu se svakom delu može zasebno pristupiti. Svaka putanja podataka takođe ima jednu MAC, SRS i ALU jedinicu. ALU jedinica obavlja sve logičke operacije nad akumulatorskim registrima.



Slika 2.8 Blok dijagram Crystal 32 jezgra

Tip memorije	DSP A	DSP B
X – data	16k SRAM, 16k ROM	10k SRAM, 8k ROM
Y – data	24k SRAM, 32k ROM	16k SRAM, 16k ROM
P – code	8k SRAM, 32k ROM	8k SRAM, 24k ROM

Tabela 2.1 Raspoloživa memorija Athena CS497004 procesora

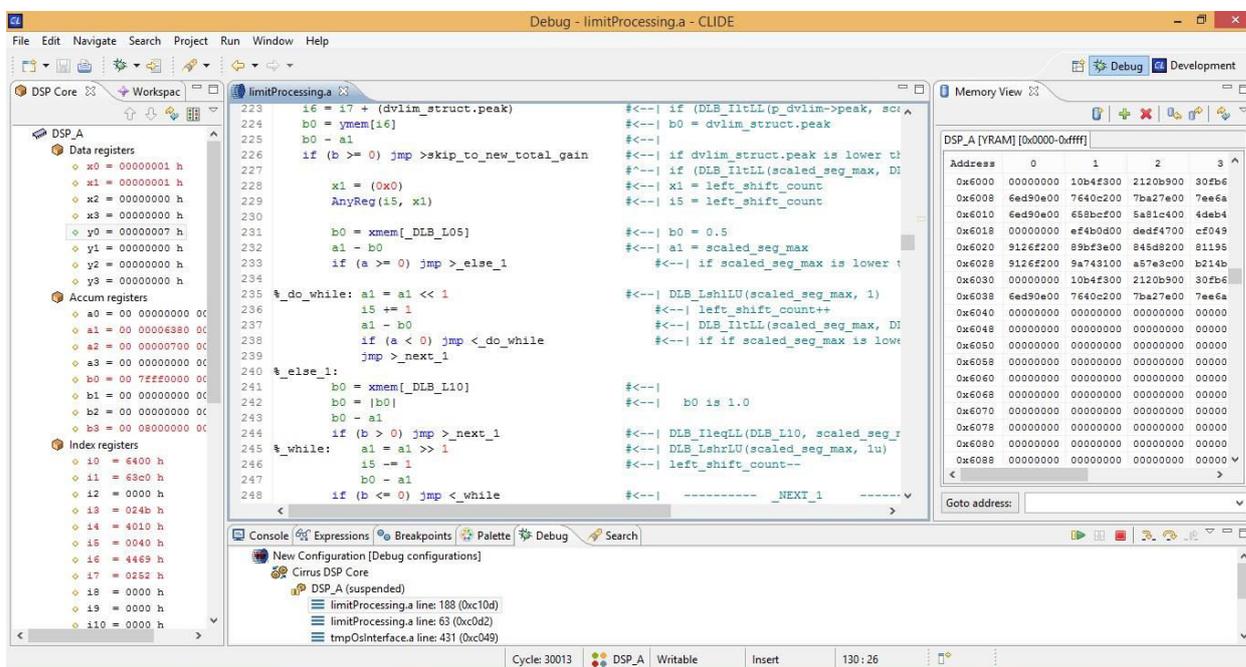
## 2.4 Razvojno okruženje CLIDE

U toku razvoja programske podrške za DSP platforme izuzetno je važno efikasno otkrivanje i uklanjanje grešaka. Da bi se ovaj zadatak olakšao koristi se simulator. Simulator omogućava simulaciju izvršenja programa namenjenog DSP platformi pomoću PC računara. Ovim se izbegava korišćenje potencijalno skupih razvojnih ploča. Takođe je na simulatoru lakše pratiti kontrolisano izvršavanje, samim tim i otkrivanje grešaka u odnosu na razvojne ploče.

Razlika između kontrolisanog izvršavanja na razvojnim pločama i simulatoru je u načinu izvršavanja. Na simulatoru je moguće kontrolisano izvršavanje programa svakog učitano bloka, jer se ulazni tok podataka može zaustaviti. Na razvojnim pločama nije moguće zaustaviti ulazni tok podataka, pa je kontrolisano izvršavanje moguće nad samo jednim blokom podataka, nakon čega nije moguće preći na sledeći, već se mora ponovo pokrenuti program.

Za otkrivanje grešaka u toku razvoja programske podrške za Crystal familiju digitalnih procesora koristi se razvojno okruženje *CLIDE*. Ovaj rad se koncentriše na upotrebu simulatora Crystal jezgra.

Na slici 2.9 prikazan je izgled jednog prozora razvojnog okruženja *CLIDE*.



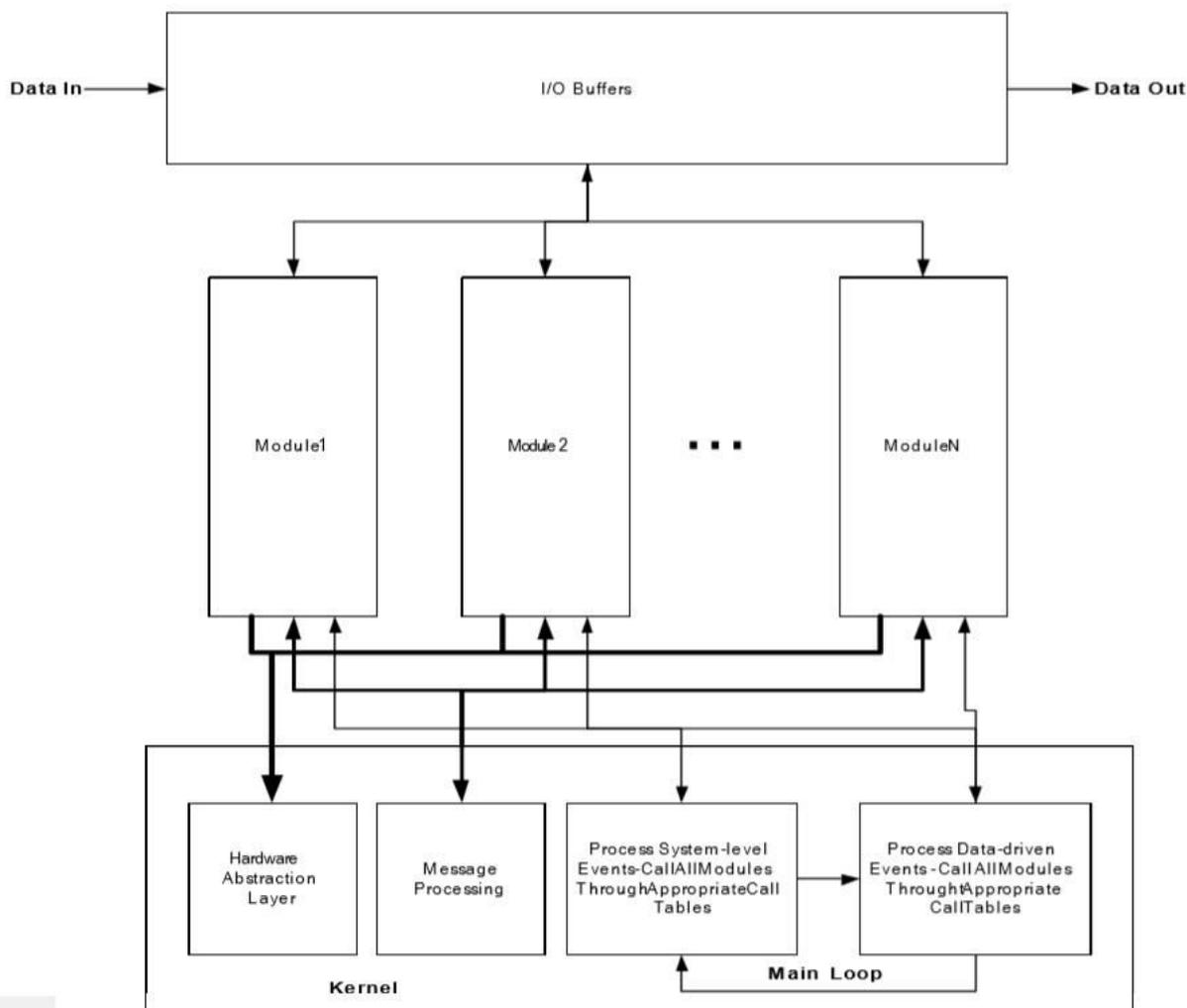
Slika 2.9 Izgled razvojnog okruženja CLIDE

Na simulatoru, kao i na hardveru, omogućen je uvid u vrednosti svih registara i memorije u svakom trenutku, što je pogodno pri razvijanju aplikacije. Takođe, postoji i mogućnost direktne promene vrednosti registara u toku kontrolisanog izvršavanja, što pomaže u analizi problema koji može nastati pri implementaciji.

## 2.5 Programsko okruženje (framework)

Cirrus Logic programsko okruženje – framework slika 2.10 predstavlja sistemsku programsku podršku procesora koja skraćuje vreme i uloženi rad za razvoj aplikacije uvodeći neke od ideja i metodologija iz objektno orijentisanog programiranja u svet asemblerskog koda.

Jezgro programskog okruženja se sastoji od jednostavnog operativnog sistema. OS u stvari predstavlja monitorsku petlju koja poziva rutine odgovarajućih modula po unapred definisanom redosledu.



Slika 2.10 Cirrus Logic programsko okruženje

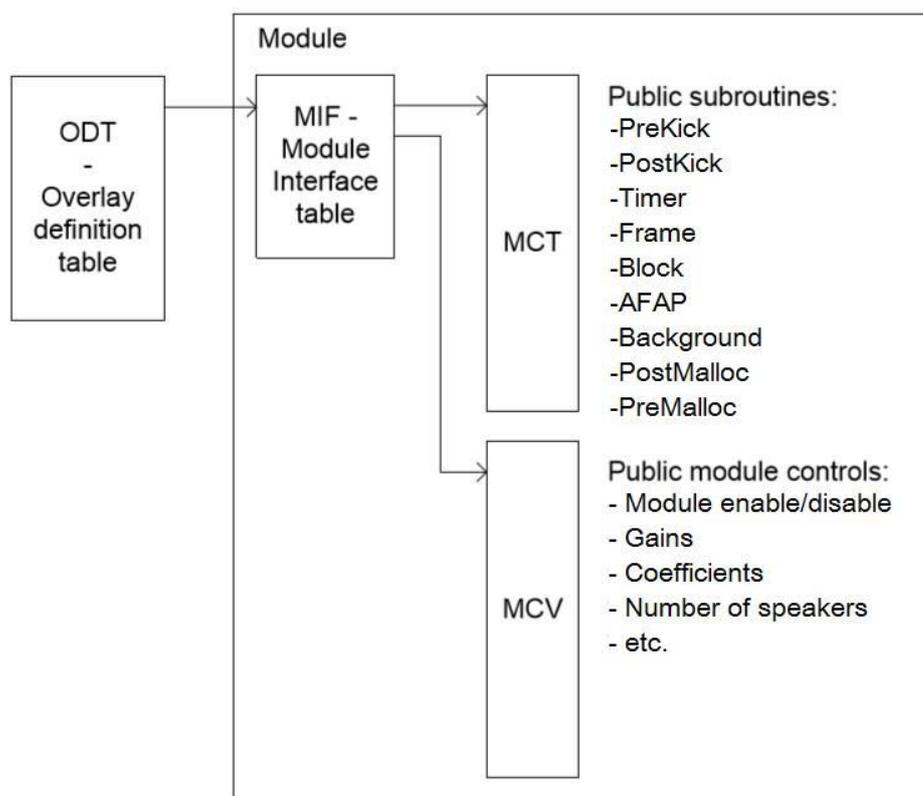
Pored dela za rukovanje događajima, važan deo programskog okruženja čini sistemski ulazno/izlazni memorijski niz koji služi za smeštanje audio podataka koji ulaze u sistem, nad kojima moduli vrše obradu i koji potom izlaze iz sistema (eng. Input-Output Buffers).

Osnovnu komponentu CS49xxx sistemske programske podrške čine moduli. Moduli su definisani kao objekti sastavljeni od rutina i podataka, u skladu sa programskim okruženjem.

Svaki modul ima svoj jedinstveni sprežni podsistem MIF kojim je modul povezan sa OS-om. Njega čini MIF tabela koja sadrži pokazivače na tabele sa ostalim sprežnim informacijama. Dve najvažnije tabele su MCT tabela i MCV tabela. Sa OS strane, sprega ka modulima se sastoji od ODT tabele koja sadrži pokazivače na MIF tabele svih učitanih modula

MCV tabela predstavlja niz javno dostupnih konfiguracionih parametara datog modula, i ona omogućava konfigurisanje modula od strane glavnog kontrolera uređaja.

MCT tabela je niz od devet elemenata – pokazivača na osnovne javne rutine. Redosled elemenata u tabeli je unapred definisan, a ukoliko neka od rutina nije definisana za dati modul, na mestu njenog pokazivača se nalazi nula. Ove rutine OS poziva kao odgovor na pojavu odgovarajućih događaja u sistemu.



Slika 2.11 Blok dijagram sprege modula sa operativnim sistemom

**Pre-kickstart** rutinu OS poziva samo nakon prijema inicijalizacione poruke (reset) i pre uspostavljanja komunikacije sa sistemskim kontrolerom. Ona omogućava inicijalizaciju modula, prvenstveno elemenata MCV tabele na njihove podrazumevane vrednosti.

**Post-kickstart** rutina se poziva nakon što je uspostavljena komunikacija sa kontrolerom. Drugim rečima ona omogućava obradu konfiguracionih podataka prosleđenih modulu od strane kontrolera.

Nakon što su izvršene Post-Kickstart rutine svih modula završena je inicijalizaciona faza i OS prelazi na izvršavanje normalnog ciklusa izvršavanja modula.

**Pre-malloc** se prvi put poziva nakon završetka inicijalizacionih rutina ukoliko je bilo koji modul u sistemu tokom inicijalizacije od OS-a zatražio inicijalizaciju dinamički dodeljene memorije. Takođe, može biti inicirana slanjem zahteva za reinicijalizaciju OS-a. Po prijemu zahteva, OS poziva Pre-Malloc rutine aktivnih modula, unutar kojih moduli prosleđuju zahteve za dodelu memorije OS-u. Pri tome se navodi memorijska zona i veličina zahtevane memorije kao i pokazivač koji treba da se inicijalizuje

**Post-malloc** se poziva nakon završetka alokacije dinamičke memorije. Ova rutina omogućava inicijalizaciju dinamički dodeljene memorije modula. Pre/Post-Malloc rutine se po potrebi izvršavaju pre prvog poziva Block i Frame rutina.

**Timer** rutina se poziva periodično na svakih N milisekundi (podrazumevano je 1ms) kao odgovor na prekid generisan od strane brojača realnog vremena. Može se koristiti za proveru ulaznih konfiguracionih podataka modula.

**Block i Frame** rutine. Njihovo izvršavanje je upravljano ulaznim tokom podataka pri čemu se *Block* rutina izvršava pri prijemu svakih 16 PCM odbiraka u U/I nizu, dok se *Frame* rutina izvršava na svakih N blokova. Ovaj broj blokova zavisi od dekoderskog modula u sistemu, odnosno od njegove jedinice dekodovanja. Na primer za AC3 dekodera to je 1536 odbiraka, DTS dekodera to može biti 512 i tako dalje. U slučaju ne-komprimiranog ulaznog toka perioda *Frame* rutine je usvojena da bude 256 odbiraka.

**AFAP** rutina, “As Fast As Possible” – “najbrže moguće”. Ova rutina se poziva kada god se desi neki događaj u sistemu, naravno uz uslov da ne prekida druge rutine istog prioriteta. *AFAP*, *Timer*, *Frame*, i *Block* rutine čine takozvanu Foreground nit (thread).

**Background** rutina se izvršava u pozadinskoj niti (Background thread) koja ima niži prioritet od Foreground niti.

U slučaju dekodera, unutar ove rutine se preuzimaju podaci iz ulaznog komprimiranog toka, obavlja se dekodovanje pri čemu se dekodovani PCM odbirci privremeno smeštaju u

---

lokalne memorijske nizove, odakle se kasnije korišćenjem *AFAP* rutine kopiraju u sistemski ulazno/izlazni memorijski niz.

Kod algoritama završne PCM obrade u *Block* rutini se kopira novi, neobrađeni, blok od 16 PCM odbiraka iz ulazno-izlazne sprežne memorije u lokalni memorijski niz nad kojim se potom u *Background* rutini vrši obrada. U *Block* rutini se istovremeno sa čuvanjem ulaznih odbiraka vrši kopiranje prethodno obrađenih odbiraka u ulazno-izlaznu sprežnu memoriju.

## 3. Koncept rešenja

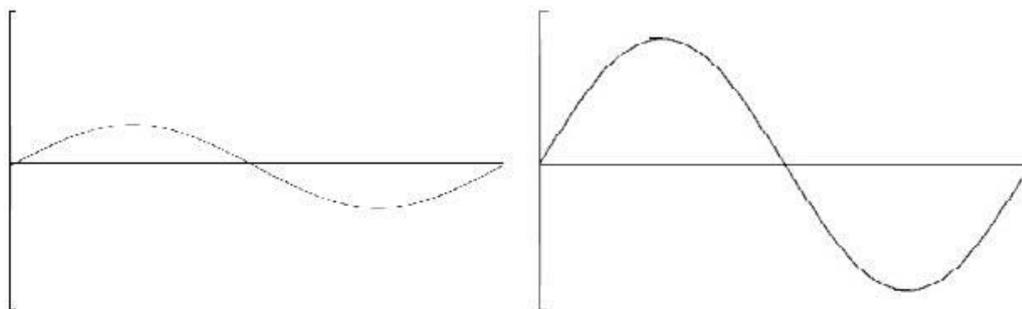
Unutar ovog poglavlja dat je kratak opis algoritma kao i tok implementacije datog algoritma za ciljnu platformu.

### 3.1 Kratak opis algoritma



Slika 3.1 Blok dijagram algoritma

Modul funkcioniše kao višekanalni ograničavač (eng. Peak limiter), tako što ograničava signale visoke vrednosti na izlazu minimizirajući tako odsecanje istog, odnosno distorzije prouzrokovane čistim odsecanjem signala. Još jedan dodatak modula je pojačavanje signala veoma malih amplituda. Ukoliko se na ulazu svih kanala pojavi signal male amplitude, algoritam će se postarati da vrednost signala, odnosno amplituda bude pojačana kako bi se na izlazu dobio signal u određenim granicama.

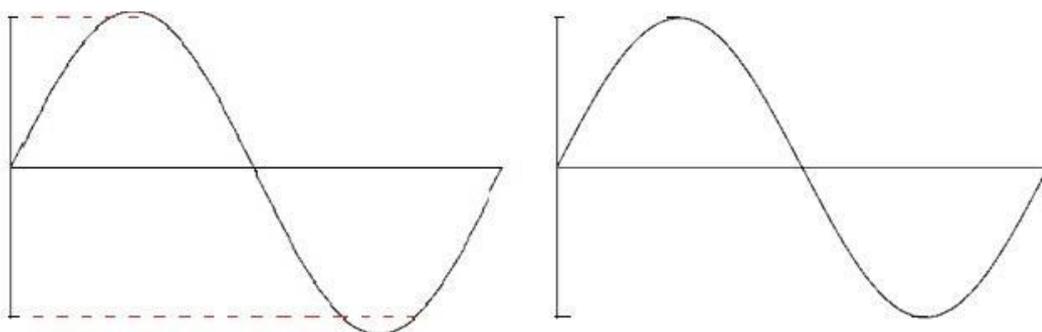


Slika 3.2 Pojačanje signala male amplitude

Unutar modula postoji opcija *Hard* i *Soft Limit*. *Hard Limit* je jednostavno odsecanje signala, dok *Soft Limit* predstavlja kompleksniju analizu ulaznog signala i njegovo modifikovanje tako da ne dođe do odsecanja, pri čemu se vrednost signala menja u vrlo maloj meri.

Ovo se postiže analizom ulaznog signala i skaliranjem celog bloka sa koeficijentima koji zavise od vrednosti samog signala, pri čemu se vodi računa i o istoriji, odnosno prethodnom bloku učitanih vrednosti.

Ukoliko trenutna vrednost signala prelazi određenu, unapred zadatu granicu, tada se ona skalira koeficijentom takvim da rezultujuća vrednost signala bude u dozvoljenim granicama. Opseg dozvoljenog izlaznog signala je podeljen u više nivoa. Razlog podele je da bi se izlazni signal što manje razlikovao od ulaznog, kao i da bi odbacivanje visoke vrednosti signala izgledalo kao da nema odsecanja.



Slika 3.3 Signal pre i posle limitera

## 3.2 Tok razvoja implementiranog rešenja

Proces razvoja softverskih aplikacija za DSP procesore sadrži određene specifičnosti u odnosu na procese razvoja opštenamenskih aplikacija. Razvoj aplikacija za DSP procesore u

praksi se vrlo često svodi na implementaciju već postojećih algoritama obrade signala na određenoj fizičkoj arhitekturi. Prilikom razvoja aplikacije kritične zahteve predstavljaju osobine odabranog DSP procesora kao što su memorijska ograničenja, ograničenja nametnuta korišćenim procesorom u vidu dužine programske memorije i vreme izvršenja.

Jedan od procesa implementacije podrazumeva pisanje asemblerskog koda za zadatak platformu na osnovu zadanog referentnog algoritma. Zatim se dati kod prilagođava da bi se uklopio u postojeće programsko okruženje. U ovakvom pristupu najviše vremena se troši u prvoj fazi koja obuhvata pisanje celokupnog asemblerskog koda na osnovu referentnog algoritma, zatim na testiranje izvršenja i otklanjanje grešaka. Dolazi se do zaključka da bi podela i strukturiranje ove faze mogli značajno doprineti brzini i pojednostavljenju razvoja kvalitetnih aplikacija.

### **3.2.1 Realizacija referentnog modula**

Početnu fazu u toku razvoja ovog DSP modula predstavlja referentni kod. Jedini zahtev koji se stavlja pred algoritam jeste ispravnost. Pošto se referentni kodovi najčešće razvijaju na računarima opšte namene, nije bitno voditi računa o efikasnom iskorišćenju memorije i procesorskog vremena, niti o aritmetici sa fiksnim zarezom kao i drugim ograničenjima ciljnih platformi. Referentni kod, model 0, je najčešće pisan u programskom jeziku C odnosno C++ ili u programskom paketu Matlab.

### **3.2.2 Implementacija modula za ciljnu platformu**

Referentni kod DSP aplikacije može da sadrži obradu podataka koristeći aritmetiku sa pokretnim ili nepokretnim zarezom u zavisnosti od toga za koju fizičku arhitekturu je pisan.

Ova faza predstavlja model 1 i obuhvata implementaciju algoritma u asemblerskom jeziku ciljne platforme. Radi se u simulatorskom režimu. Da bi se napisao optimalan kod potrebno je sagledati koji resursi ciljne platforme stoje na raspolaganju. Dobrom organizacijom resursa se postiže optimalni kod, kako u pogledu zauzetosti memorije tako i u pogledu iskorišćenosti procesorskog vremena, koja se najčešće izražava jedinicom MIPS (eng Million instruction per second).

Prvi korak u ovoj fazi jeste ispravnost algoritma. U najvećem broju slučajeva kod generisan u ovoj fazi je izvršiv na ciljnoj platformi. Međutim, neretko prevazilazi ograničenja ciljne platforme.

---

Potrebno je napomenuti da nakon prelaska sa aritmetike u pokretnom zarezu na aritmetiku u nepokretnom zarezu vrlo često dolazi do razlike preciznosti, te je velika verovatnoća da rezultati obrade referentnog i modela pisanog u assembleru ne budu identični na nivou bita. Ono što je zadatak jeste da razlike budu svedene na minimum.

U toku pisanja koda, potrebno je maksimalno iskoristiti mogućnosti koje nudi ciljna platforma. Potrebno je uočiti koji se delovi mogu optimizovati u vidu paralelnog izvršavanja više operacija procesora u okviru jedne instrukcione reči.

### **3.2.3 Optimizacija implementiranog modula**

Poslednji korak, odnosno model 2, predstavlja optimizaciju asemblerskog koda. Cilj ovog procesa je maksimalno smanjenje programske memorije i potrošnje procesorskog vremena. Tehnika optimizacije koja se koristi u ovoj fazi je promena u implementaciji algoritma, tako da se funkcionalnost, kao ni algoritam, ne menja. Krajnji rezultat je optimizovan asemblerski kod, prilagođen izvršavanju na ciljnoj platformi.

## 4. Implementacija algoritma za ciljnu platformu

Unutar ovog poglavlja objašnjeni su detalji implementacije za svaku od navedenih faza u prethodnom poglavlju. Takođe je dat prikaz iskorišćenosti resursa ciljne platforme.

### 4.1 Model 0

Model 0 predstavlja referentni C kod koji je polazna osnova za implementaciju algoritma na ciljnoj platformi. Referentni kod može biti podeljen u dve celine. Prva obuhvata rad sa ulaznom i izlaznom datotekom. Implementirani su mehanizmi za pravilno otvaranje ulazne i izlazne datoteke, iščitavanje ulaznih odbiraka, kao i njihov upis u izlaznu datoteku

Druga celina predstavlja obradu odbiraka. Obrada učitanih odbiraka je realizovana kao što je to opisano u prethodnom poglavlju (3.1). Moduli za datu obradu su preuzeti iz komercijalnog projekta.

### 4.2 Model 1

Model 1 podrazumeva promenu radnog okruženja i programskog jezika. Algoritam referentnog koda, napisan u programskom jeziku C, u *Microsoft Visual Studio 2010* radnom okruženju se implementira u asemblerskom jeziku ciljne platforme, u *CLIDE* radnom okruženju.

Razvoj algoritma je u simulatorskom režimu, gde se kao ciljna platforma koristi Cirrus DSP Core koja predstavlja simulator Crystal jezgra.

Modul se sastoji iz sledećih celina:

- limiterOsInterface
- limiterDataVars
- limiterProcessing
- limiterDataStruct
- limiterGlobalConst

#### 4.2.1 LimiterOsInterface

Predstavlja spregu modula sa OS-om. Sprega se ostvaruje pozivanjem rutina iz MCT tabele.

Kako je u poglavlju 2.5 opisana svaka od rutina, pomenuto je da se Block rutina poziva na svakih 16 učitanih PCM odbiraka, dok je veličina bloka nad kojim se vrši obrada 64 odbiraka. Zbog ovoga napravljen je PingPong, dvostruki bafer, koji je duplo veći od veličine bloka obrade. Ovakav bafer služi da, dok se jedna strana bafera puni novim odbircima, nad drugoj strani se vrši obrada. Kada je obrada završena i učitano novih 64 odbiraka, tada se strane menjaju, pa se obrada vrši nad novim odbircima, dok se obrađeni odbirci smeštaju u izlaz, a na njihovo mesto dolaze novi.

U Post-Kickstart rutini inicijalizuju se vrednosti promenljivih koje se koriste kao kontrolne promenljive, kao i inicijalizacija strukture koja se koristi u obradi signala.

U Pre-Malloc rutini je zatražena memorija za PingPong bafer, dok se u Post-Malloc rutini inicijalizuje dinamički zauzeta memorija.

U okviru Block rutine realizovano je kopiranje sadržaja u i iz PingPong bafera nad kojim se vrši obrada.

Pošto se radi u simulatoru, neophodno je konfigurisati simulator pre pozivanja rutina iz MCT tabele. Konfiguracija simulatora se vrši pozivom I\_S\_SimulatorInit funkcije, u kojoj se konfiguriše brzina odabiranja (eng. Sample Rate), broj ulaznih i izlaznih kanala, kao i da li se radi o PCM odbircima, odnosno da li je nekomprimovan ili komprimovan ulazni tok.

#### 4.2.2 LimiterDataStruct

U ovoj datoteci definisana je struktura limitera. Struktura sadrži promenljive i bafere koji se koriste u izračunavanju koeficijenta kojima se ulazni signal skalira da bi se dobio izlaz u dozvoljenim granicama.

### 4.2.3 LimiterProcessing

U okviru ove datoteke realizovane su sledeće funkcije:

- X\_S\_dap\_limitProcess
- I\_S\_dap\_new\_gain\_for\_block
- I\_S\_dap\_hard\_limit
- I\_S\_dap\_dvlim\_blk\_limit\_and\_boost
- I\_S\_dap\_find\_abs\_max
- I\_S\_dap\_dvlim\_blk\_ch\_limit\_and\_boost

Unutar funkcije X\_S\_dap\_limitProcess se određuje koja vrsta limitiranja se vrši, na osnovu čega se pozivaju odgovarajuće funkcije.

Ukoliko se vrši Hard Limit, poziva se funkcija I\_S\_dap\_hard\_limit koja odseca signal na unapred zadatu vrednost.

Ukoliko se vrši Soft Limit pozivaju se sledeće funkcije:

I\_S\_dap\_find\_abs\_max u kojoj se pronalazi maksimalna apsolutna vrednost među učitanim odbircima. Potom se poziva I\_S\_dap\_new\_gain\_for\_block funkcija. Unutar ove funkcije realizovan je algoritam koji omogućuje izračunavanje koeficijenta sa kojim se signal množi, tako da izlazni signal ostane u dozvoljenim granicama.

I\_S\_dap\_dvlim\_blk\_limit\_and\_boost za svaki kanal priprema podatke i poziva funkciju I\_S\_dap\_dvlim\_blk\_limit\_and\_boost u kojoj se izračunavaju koeficijenti i skalira ulazni signal.

### 4.2.4 LimiterDataVars i limiterGlobalConst

U okviru ovih datoteka statički je zauzeta memorija za sve promenljive (pokazivače, tabele, bafere) koje se koriste u modulu. U datoteci limiterGlobalConst definisane su sve konstante. U datoteci limiterDataVars definisane su ostale globalne promenljive i tabele koje se koriste u modulu.

## 4.3 Model 2

Model 2 predstavlja poslednji korak u razvoju ovog modula. Vršiti se izmena prvobitne implementacije algoritma u cilju smanjenja utrošenih resursa, kako programske memorije tako i procesorskog vremena.

Izmenama u implementaciji algoritma postignuta je ušteda procesorskog vremena do 6 MIPS-a. Ovo je postignuto uočavanjem delova koda koji su nezavisni i čija izmena u redosledu

izvršavanja dovodi do manjeg broja instrukcija, pa samim tim i većom uštedom procesorskog vremena.

#### 4.4 Iskorišćenost resursa ciljne platforme

Optimalno iskorišćenje resursa podrazumeva minimalno iskorišćenje memorije za maksimalnu iskorišćenost procesorskog vremena. Jedinica za iskorišćenost procesorskog vremena je MIPS – milion instrukcija u sekundi.

$$MIPS = \frac{broj\_ciklusa * \frac{Fs}{BLOCK\_SIZE}}{1000000}$$

Parametar  $F_s$  predstavlja frekvenciju odabiranja, tj. broj odbiraka ulaznog signala u jednoj sekundi.  $BLOCK\_SIZE$  predstavlja veličinu bloka obrade. U ovom slučaju  $F_s$  je 48kHz, a  $BLOCK\_SIZE$  je 64.

Da bi se utvrdio broj ciklusa koji je potreban za izvršavanje funkcije, potrebno je koristiti kontrolisano izvršavanje programa u simulatorskom režimu. Ovim načinom je moguće tačno utvrditi broj ciklusa potreban za izvršenje funkcije u kojoj se vrši obrada prethodno učitanih odbiraka. Uopšteno, kada se radi profilisanje koda u vidu računanja utrošenosti procesorskog vremena, meri se minimalna i maksimalna vrednost, kako bi se dobila srednja vrednost potrošnje procesorskog vremena, koja je i najbitnija.

U tabeli 4.1 dat je pregled iskorišćenosti resursa ciljne platforme.

X memorija	Y memorija	P memorija	MIPS
117	1106	364	14,984

Tabela 4.1 Iskorišćenost resursa ciljne platforme

U tabeli 4.2 dat je detaljniji pregled iskorišćenosti procesorskog vremena po podfunkcijama algoritma.

Naziv funkcije	MIPS
I_S_dap_find_abs_max	2,340
I_S_dap_new_gain_for_block	0,116
I_S_dap_dvlim_blk_ch_limit_and_boost	1,558

Tabela 4.2 Potrošnja procesorskog vremena po podfunkcijama algoritma

U tabeli 4.1 data je maksimalna vrednost iskorišćenosti procesorskog vremena, odnosno koliko se ovakvom implementacijom limitera troši MIPS-a za osmokanalni signal, dok su u tabeli 4.2 date vrednosti koje predstavljaju utrošenost procesorskog vremena na izvršavanje podfunkcija algoritma. Vrednost MIPS-a direktno zavisi od broja kanala ulaznog signala, odnosno, što je više kanala prisutno, potrebno je više procesorskog vremena za izvršenje algoritma.

U tabeli 4.3 data je vrednost iskorišćenosti procesorskog vremena u zavisnosti od broja kanala ulaznog signala.

Ulazni testni vektor	MIPS
Dvo-kanalni testni vektor	5,572
Šesto-kanalni testni vektor	11,804
Osmo-kanalni testni vektor	14,984

Tabela 4.3 Potrošnja procesorskog vremena u zavisnosti od broja kanala ulaznog signala

## 5. Ispitivanje

Ispitivanje i verifikacija modula predstavljaju krajnji korak u implementaciji. Svrha ovog procesa je da se dokaže da se izlazi generisani implementiranim modulom slažu sa referentnim izlazima koji se dobijaju uz projekat ili se generišu pomoću referentne aplikacije. Postoji više načina ispitivanja rezultata:

1. Slušni testovi – provera čujnih artefakta u odnosu na referentni izlaz
2. Testovi identičnosti u bit – poređenje referentnih i ispitnih izlaza na nivou bit-a
3. Spektralni testovi – poređenje u spektralnom domenu.
4. Testovi koji se obavljaju korišćenjem namenskim verifikovanih programa koji se dobijaju uz referentnu aplikaciju.

Definisana ispitivanja se mogu izvršavati ručno. Ovakav pristup je sporiji i pri tome se unosi mogućnost ljudske greške. Zbog toga se za testiranje koriste skript jezici kojima se testovi automatizuju, čime se verifikacija ubrzava i otklanja se mogućnost ljudske greške. Najčešće se skripte pišu korišćenjem batch ili python programskim jezicima.

Za testiranje implementiranog modula, sastavljeni su višekanalni testni signali koji u potpunosti proveravaju ispravnost algoritma.

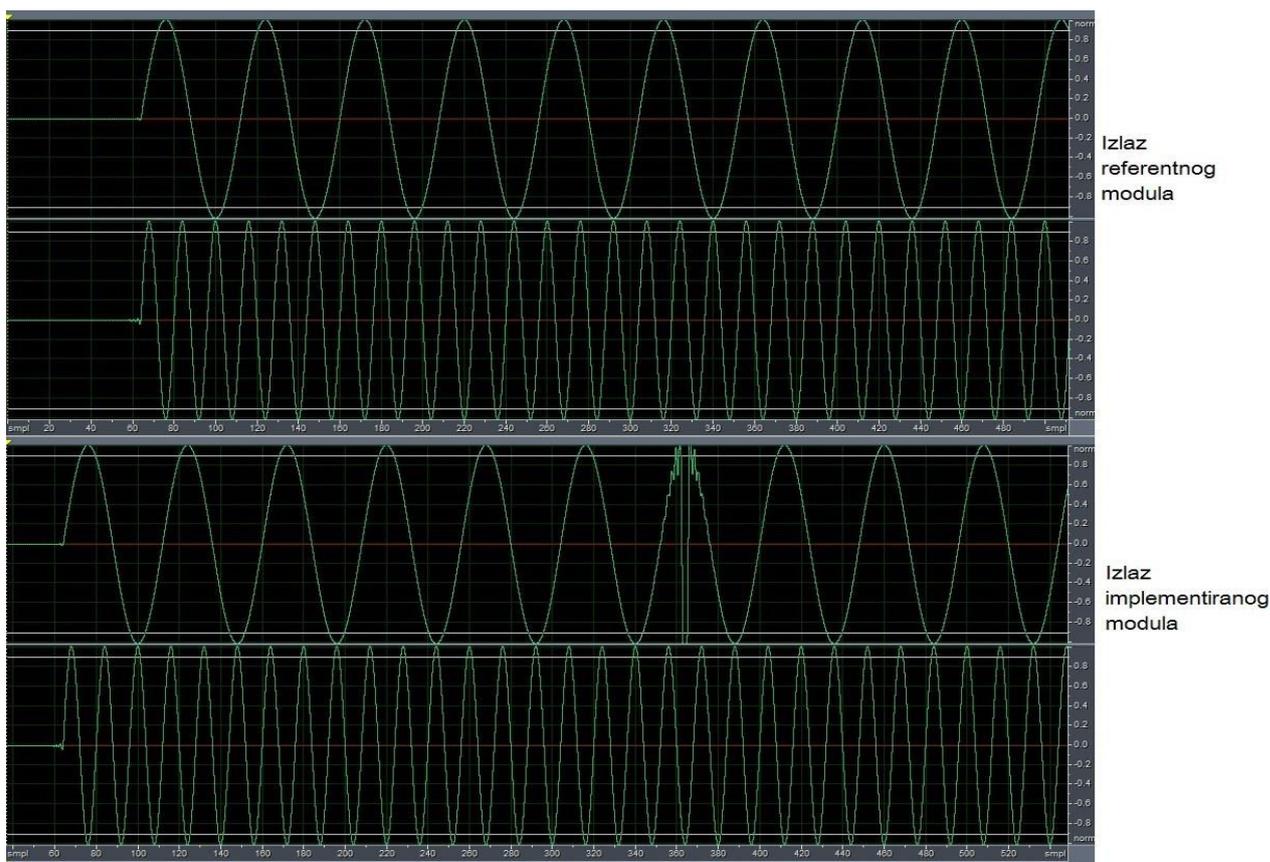
### 5.1 Slušni testovi

Jedan pristup provere ispravnosti rezultata jesu slušni testovi. Ovaj način testiranja spada u kategoriju subjektivnih metoda ispitivanja. Slušni testovi u formalnom smislu podrazumevaju

puštanje ispitnih vektora određenom broju ispitanika koji su prethodno obučeni na koji način treba da procene kvalitet zvuka.

## 5.2 Analiza signala u vremenskom domenu

Poređenje izlaza u vremenskom domenu predstavlja najbrže uočavanje grešaka u implementaciji algoritma.



Slika 5.1 Jasan prikazi greške generisane pri implementaciji

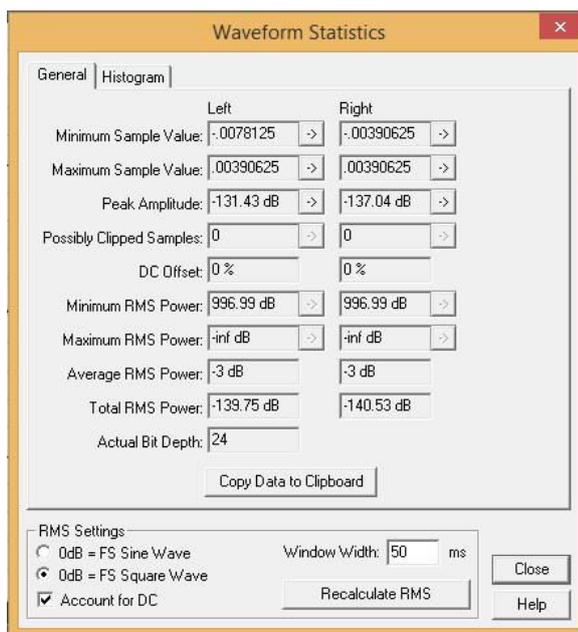
Na slici 5.1 se može uočiti razlika između izlaza referentnog modela i modela pisanog za ciljnu platformu u početnoj fazi razvoja. Ova razlika predstavlja grešku u implementaciji i pomaže u njenom otklanjanju, jer se analizom signala može pretpostaviti šta je prouzrokovalo dato ponašanje.

Analizom statistike izlaznog signala može se videti tačnost implementiranog algoritma. To se vrši oduzimanjem izlaza generisanog u simulatorskom režimu od izlaza generisanog referentnog modula. Na slici 5.2 je prikazan jedan prozor ovakvog poređenja, na kom se jasno vidi razlika među signalima izražena u [dB]. Iz formule za odnos signal/šum:

$dB(SNR) = 10 \log_{10} SNR$ , gde 1 bit tačnosti povećava odnos signal/šum za 6dB. Može se izračunati da je maksimalna razlika između poređenih signala 2 bita.

Do ovog računa se dolazi tako što se od referentnog oduzme signal generisan implementiranim modulom. Kao rezultat dobija se signal koji predstavlja razliku ova dva signala. Maksimalna vrednost dobijenog signala (eng. Peak Amplitude) predstavlja najveću razliku među poređenim signalima. Da bi se dobila vrednost razlike u bitima, potrebno je dati rezultat podeliti sa 6, jer kao što je prethodno rečeno, svakim bitom preciznosti povećava se i odnos signal-šum (SNR) za 6dB.

Da bi se jasnije predstavilo šta tačno znači bit razlike i koliko je on bitan dat je sledeći primer: za 24-bitni signal, ukoliko svi biti predstavljaju koristan signal, odnos signal-šum je 144dB. Ukoliko u 24-bitnom signalu postoji šum koji uzima 8 bita korisnog signala, tada je vrednost SNR-a 96dB. Potrebno je napomenuti da je 96dB donja granica SNR-a, odnosno manji odnos signal-šum nije dozvoljen.

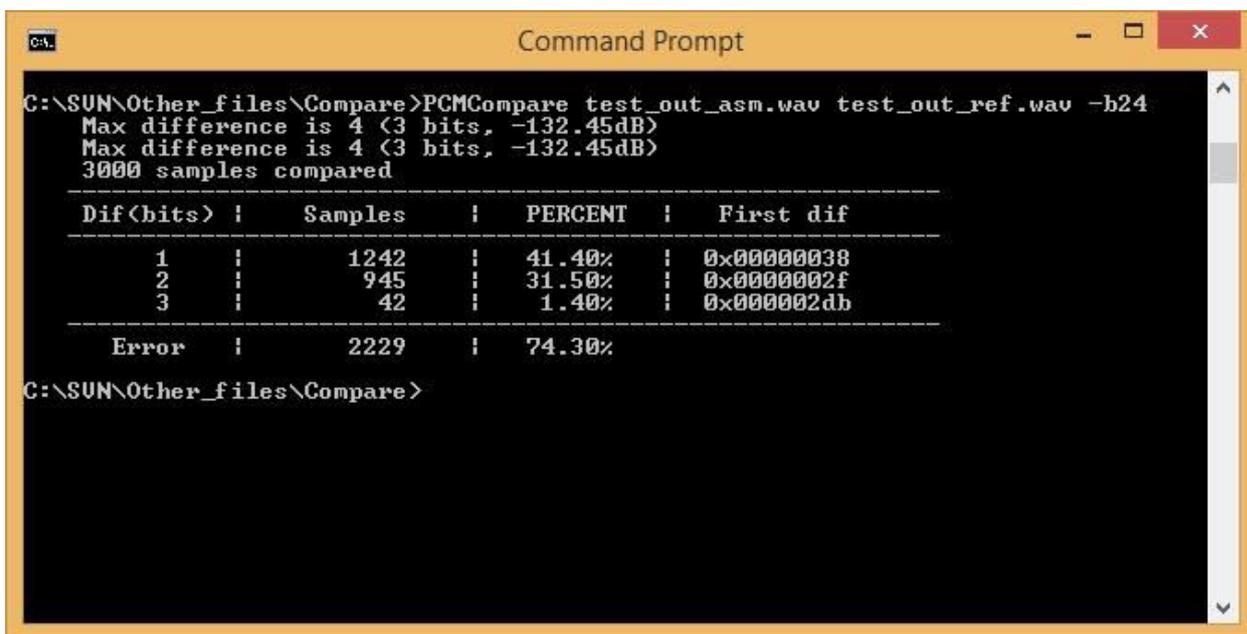


Slika 5.2 Statistika signala dobijena oduzimanjem izlaza

Ovaj način testiranja služi za brzu proveru tačnosti, odnosno greške u toku implementacije. Da bi se dobio precizniji rezultat koji pokazuje koliko se neka dva signala zaista razlikuju, vrše se Bit-identični testovi.

### 5.3 Bit identični testovi

Ovaj vid testiranja podrazumeva poređenje izlaza generisanih referentnim programom, sa izlazima generisanim u ostalim fazama na nivou bita. Za potrebe ovakvog testiranja korišćeni su alat PCMCompare.exe i Total Commander. Kao što je i očekivano, izlazi referentnog programa se razlikuju od izlaza generisanih u drugoj i trećoj fazi. Osnovni razlog postojanja ovih razlika jeste prelazak sa aritmetike u pokretnom zarezu na aritmetiku sa nepokretnim zarezom. Primer ovog vida testiranja dat je na sledećoj slici.



```
C:\SUN\Other_files\Compare>PCMCompare test_out_asm.wav test_out_ref.wav -b24
Max difference is 4 (3 bits, -132.45dB)
Max difference is 4 (3 bits, -132.45dB)
30000 samples compared

-----
Dif(bits) | Samples | PERCENT | First dif
-----
1 | 1242 | 41.40% | 0x00000038
2 | 945 | 31.50% | 0x0000002f
3 | 42 | 1.40% | 0x000002db
-----
Error | 2229 | 74.30%

C:\SUN\Other_files\Compare>
```

Slika 5.3 Poređenje izlaza iz referentnog modela i modela implementiranog u asembleru

Na ovoj slici prikazano je poređenje referentnih izlaza sa izlazima iz druge faze, odnosno rane faze implementacije simulatorskog projekta.

```

C:\SUN\test_stream_os_sim>PCMCompare 8ch_48k_asm.wav 8ch_48k_ref.wav -b24
Max difference is 2 (2 bits, -138.47dB)
Max difference is 2 (2 bits, -138.47dB)
119680 samples compared
-----
Dif(bits) | Samples | PERCENT | First dif
-----
1 | 21419 | 17.90% | 0x00000ad6
2 | 190 | 0.16% | 0x0002dbde
-----
Error | 21609 | 18.06%
C:\SUN\test_stream_os_sim>

```

Slika 5.4 Rezultati poređenja izlaza generisanih za maksimalan broj kanala

Na slici 5.4 prikazani su rezultati poređenja izlaza generisanih za maksimalan broj kanala posle optimizacije. Može se primetiti da je najveća greška 2 bita, koja iznosi svega 0,16% (190 odbiraka) od ukupnog broja odbiraka (119680), dok se najveći broj odbiraka (21419 odbiraka, 17,90%) razlikuje u jednom bitu.

Generisanje cele sekvence izlaza u simulatorskom režimu dugo traje. Da bi se izbeglo čekanje, generiše se samo deo izlaznog vektora.

### 5.3.1 Automatski bit identični testovi

Ručno poređenje izlaza je sporo i sklono greškama, pa se zbog toga pišu skripte batch ili python programskim jezicima, koje omogućuju automatsko generisanje i poređenje izlaza, pri čemu se rezultat poređenja smešta u tekstualnu datoteku radi kasnije analize.

Za potrebe ovog rada napravljeni su specifični test-vektori: dvo-kanalni, šesto-kanalni i osmo-kanalni testni signali, čime se teži pokrivanje svih slučajeva. Poređenjem generisanih izlaza referentnog i implementiranog modula, potvrđuje se ispravnost algoritma razvijenog za ciljnu platformu.

Najveća razlika od 2 bita se javlja kod osmo-kanalnog signala, dok je razlika u odnosu na referentni kod šesto-kanalnog i dvo-kanalnog signala samo 1 bit.

Razlika kod osmo-kanalnog signala od 2 bita se javlja, jer se ovaj testni vektor sastoji od osam signala veoma male amplitude, pa se prilikom računanja koeficijenta u određenim, kritičnim delovima, pri pojačanju, javlja razlika u odnosu na izlaz generisan referentnim modulom, zbog razlike u aritmetici pokretnog i nepokretnog zarez.

Razlog pravljenja takvog osmo-kanalnog testnog vektora je da bi se potvrdilo da implementirani modul pojačava ulazni signal, ukoliko je to potrebno, kao što je to zadato algoritmom.

Šesto-kanalni signal se sastoji iz dva kanala velike amplitude, jednog kanala veoma male amplitude i jednog kanala u granicama normale. Dvo-kanalni testni vektor sadrži samo dva signala velike amplitude.

Rezultat generisan jednim ovakvim testiranjem dat je na slici 5.5.

```
1 Max difference is 1 (1 bits, -144.49dB)
2 96672 samples compared
3 -----
4 Dif(bits) | Samples | PERCENT | First dif
5 -----
6 1 | 19713 | 20.39% | 0x0000093e
7 -----
8 Error | 19713 | 20.39%
9 -----
```

Slika 5.5 Rezultat jednog od testiranja

## 6. Zaključak

U okviru ovog rada implementiran je DSP modul za ograničavanje nivoa izlaznog signala na procesoru familije CS49XXX firme Cirrus Logic.

Implementacija je podeljena u tri modela. Model 0 je analiza i razumevanje referentnog koda, odnosno algoritma koji je potrebno implementirati. Koristi programski jezik *C* i *Visual Studio 2010* razvojno okruženje.

Sledeći model predstavlja prelazak na asemblerski jezik ciljne platforme i razvojno okruženje *CLIDE*. Bitno je napomenuti da se u ovom koraku prelazi sa aritmetike u pokretnom zarezu na aritmetiku u nepokretnom zarezu, što samo po sebi uvodi razliku u preciznosti. Glavni cilj je postizanje iste funkcionalnosti zadate referentnim modulom, dok je iskorišćenost resursa u drugom planu. Kada je ovaj cilj postignut, i ispravnost proverena analizom generisanih izlaza u vremenskom domenu i bit-identičnim testovima, prelazi se na sledeći korak što je optimizacija.

Optimizacija koja se u modelu 2 sprovodi je poboljšanje iskorišćenosti procesorskog vremena. Ovo se vrši tako što se redosled izvršavanja algoritma menja tako da funkcionalnost ostane identična, ali se paralelizuju delovi algoritma koje je moguće istovremeno obavljati. Takođe se redosled nekih operacija unutar funkcija menja kako bi se postigla što veća brzina izvršavanja, odnosno smanjio broj MIPS-a.

Nakon optimizacije smanjena je potrošnja procesorskog vremena za 6 MIPS-a, i iznosi 14,984 MIPS-a. Ovim postupkom je takođe uštedena i programska memorija i svedena na 364 reči, dok je za podatke zauzeto 1223 reči u X i Y memoriji.

Na samom kraju izvršeno je ispitivanje i verifikacija implementiranog modula proverom bit-identičnosti između generisanih izlaza u referentnom i implementiranom modelu. Testiranjem je pokazana maksimalna razlika od dva bita, koja je za module završne obrade (eng.

---

Post-Processing Module) zadovoljavajuća. Međutim, najveći broj odbiraka se razlikuje u jednom bitu, što se zajedno sa razlikama od dva bita može pripisati promeni aritmetike.

Dalja unapređenja ovog modula imala bi za cilj još veći stepen optimizacije, odnosno manju potrošnju procesorskog vremena i memorije.

## 7. Literatura

- [1] V. Kovačević, M. Popović, M. Temerinac, N. Teslić: *Arhitekture i algoritmi digitalnih signal procesora 1*, FTN izdavaštvo, Novi Sad, 2005
- [2] *Arhitekture i algoritmi DSP II – praktikum za laboratorijske vežbe*, Novi Sad, 2013
- [3] M. Temerinac, S. Berber, Ž. Lukač: *Osnovi algoritama i struktura DSP 1*, Novi Sad, 2013
- [4] M. Temerinac, Ž. Lukač, I. Kaštelan: *Osnovi algoritama i struktura DSP 2*, FTN izdavaštvo, Novi Sad, 2016
- [5] M. Popović: *Digitalna obrada signala*, Akademska misao, 2003
- [6] S. K. Mitra: *Digital signal processing*, McGraw-Hill Global Education, 1998/2005
- [7] John G. Proakis, D. Manolakis, *Digital Singnal Processing: Principles Algorithms and Applications*, Prentice Hall, 1995
- [8] M. Hajduković: *Operativni sistemi*, FTN izdavaštvo, Novi Sad, 2004
- [9] M. Đukić, N. Četić, R. Obradović, M. Popović: *An Approach to Instruction Set Compiled Simulator Development Based on a Target Processor C Compiler Back-End Design*, ECBS-EERC, Engineering of Computer Based Systems, 2009