



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Ненад Врга

Једно решење рендеровања аудио објекта на ДСП платформи

МАСТЕР РАД

Нови Сад, 2017



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:	
Идентификациони број, ИБР:	
Тип документације, ТД:	Монографска документација
Тип записа, ТЗ:	Текстуални штампани материјал
Врста рада, ВР:	Дипломски – мастер рад
Аутор, АУ:	Ненад Врга
Ментор, МН:	Др Јелена Ковачевић
Наслов рада, НР:	Једно решење рендеровања аудио објеката на ДСП платформи
Језик публикације, ЈП:	Српски / латиница
Језик извода, ЈИ:	Српски
Земља публикавања, ЗП:	Република Србија
Уже географско подручје, УГП:	Војводина
Година, ГО:	2017
Издавач, ИЗ:	Ауторски репринт
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога)	7/45/18/4/26/0/0
Научна област, НО:	Електротехника и рачунарство
Научна дисциплина, НД:	Рачунарска техника
Предметна одредница/Кључне речи, ПО:	ДСП, аудио објекти, рендеровање
УДК	
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У последњих десет година дошло је до великих промена како у количини тако и у начину коришћења аудио и видео садржаја. Једна од кључних карактеристика новијих технологија је тачна просторна репродукција звучног сигнала, независно од циљне поставке звучника. Приступ коришћењу аудио објеката је управо то и омогућио. Уз помоћ рендеровања објекти се динамички распоређују на звучнике присутне у систему. У оквиру овог рада описано је једно решење рендеровања аудио објеката ДСП (eng. Digital Signal Processor) платформи.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	Председник: Др Илија Башичевић
	Члан: Др Иван Мезеи
	Члан, ментор: Др Јелена Ковачевић
	Потпис ментора



KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	Monographic publication
Type of record, TR :	Textual printed material
Contents code, CC :	Master Thesis
Author, AU :	Nenad Vrga
Mentor, MN :	PhD Jelena Kovačević
Title, TI :	One solution of object audio rendering algorithm on DSP platform
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Republic of Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2017
Publisher, PB :	Author's reprint
Publication place, PP :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/45/18/4/26/0/0
Scientific field, SF :	Electrical Engineering
Scientific discipline, SD :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, S/KW :	DSP, audio object, rendering
UC	
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, N :	
Abstract, AB :	In last decade, there have been a major changes in both quality and quantity of audio and video content. One of the key features of newer technology is accurate spatial reproduction of the sound signal, regardless of the speaker configuration. Using concept of 3D object enabled it. With rendering objects are dynamically allocated to the speakers present in the system configuration. In this work is described one solution of object audio rendering algorithm on DSP platform.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	
Defended Board, DB :	President: PhD Ilija Bašičević
	Member: PhD Ivan Mezei
	Member, Mentor: PhD Jelena Kovačević
	Mentor's sign

Zahvalnost

Zahvaljujem se razvojno-istraživačkom institutu “RT-RK” iz Novog Sada, kao i mentoru i kolegama iz tima na stručnoj pomoći prilikom realizacije ovog rada.

SADRŽAJ

1. Uvod.....	1
2. Teorijske osnove	3
2.1 Audio sistemi zasnovani na više kanala	6
2.2 Audio sistemi zasnovani na objektima.....	6
2.3 Meta podaci	7
2.4 Ambisonik	8
2.5 Koordinatni sistemi	9
2.6 Prijemnik audio i video sadržaja	11
2.7 Soundbar.....	13
3. Koncept rešenja.....	15
3.1 Objektni sistem.....	15
3.2 Renderovanje objekata u prostoru.....	16
3.2.1 VBAP (Pomeranje amplitude vektora u 3D prostoru).....	17
4. Implementacija.....	20
4.1 Ciljna platforme.....	20
4.2 Razvoj sistema na DSP platformi.....	21
4.2.1 Radni okvir procesora CS49844	24
4.3 Moduli implementacije	26
4.3.1 Dekoder (DSPA jezgro).....	27
4.3.2 Modul za renderovanje objekata u prostoru (DSPB jezgro).....	29
4.3.2.1 Inicijalizacija	29
4.3.2.2 Renderovanje	32
4.3.3 PostProcessing (DSPC jezgro)	34

4.4	Protokol za prenos podataka između jezgara	34
5.	Ispitivanje i rezultati	37
6.	Zaključak	43
7.	Literatura.....	44

SPISAK SLIKA

Slika 2.1 Raspored zvučnika standardizovanog 5.1 sistema.....	4
Slika 2.2 Prikaz konfiguracija zvučnika 7.1 i 5.1 sistema	5
Slika 2.3 Sistem 22.2.....	5
Slika 2.4 Primer jedne putanje pokretnog objekta	7
Slika 2.5 Enkodovane funkcije ambisonika prvog reda.....	9
Slika 2.6 Enkodovane funkcije ambisonika drugog reda.....	9
Slika 2.7 Kartezijanski koordinatni	10
Slika 2.8 Sferni koordinatni sistem	10
Slika 2.9 Uprošćena blok šema sistema prijemnika audio i video sadržaja.....	12
Slika 2.10 Prikaz izgleda i blok dijagram prijemnika audio-video signala zasnovanog na CS49844 procesoru	13
Slika 2.11 Ilustracija soundbar sistema	14
Slika 2.12 Ilustracija dobijanja surround efekta.....	14
Slika 3.1 Blok šema protoka zvuka.....	15
Slika 3.2 Koncept renderovanja objekata.....	16
Slika 3.3 Definisanje vektora u stereo konfiguraciji	18
Slika 3.4 Konfiguracija trodimenzionalnog VBAP-a.	18
Slika 4.1 Blok dijagram CS49844 čipa	21
Slika 4.2 Blok dijagram sprege modula sa operativnim sistemom	26
Slika 4.3 Raspodela modula implementiranog sistema na jezgra ciljne platforme.....	27
Slika 4.4 Ping-Pong struktura memorijskog niza.....	28
Slika 4.5 AHB magistrala	35
Slika 5.1 Primer generisanog testnog slučaja za BBT alat (deo .tst datoteke).....	38

Slika 5.2 BBT aplikacija nakon pokretanja.....	39
Slika 5.3 Blok dijagram izvršavanja jednog BBT testa	39
Slika 5.4 Ispitno okruženje.....	40
Slika 5.5 Prikaz ishoda izvršavanja testova	41

SPISAK TABELA

Tabela 4.1 Količina raspoloživih resursa ciljne platforme.....	21
Tabela 4.2 Utrošak resursa DSPA jezgra.....	28
Tabela 4.3 Utrošak resursa DSPB jezgra	34
Tabela 4.4 Utrošak resursa DSPC jezgra	34

SKRAĆENICE

DSP	- Digitalni signal procesor, <i>Digital Signal Processor</i>
VBAP	- Amplitudsko pomeranje na bazi vektora, <i>Vector Base Amplitude Panning</i>
DAW	- Muzička stanica, <i>Digital Audio Workstation</i>
MAC	- Instrukcija za množenje i akumuliranje rezultata, <i>Multiply and Accumulate</i>
MIPS	- Milion instrukcija u sekundi, <i>Milion instructions per second</i>
SRS	- Jedinica sa aritmetičko pomeranje, zaokruživanje i saturaciju Crystal DSP familije procesora, <i>Shift Round Saturate</i>
DMA	- Direktan pristup memoriji, <i>Direct Memory Access</i>
SPDIF	- Sony Philips digitalni sprežni podsistem, <i>Sony Philips Digital InterFace</i>
HDMI	- Multimedijalni sprežni podsistem visoke definicije, <i>High Definition Multimedia Interface</i>
MIF	- Sprežni podsistem, <i>Module Interface</i>
MCT	- Tabela pozivanja modula, <i>Module Call Table</i>
MCV	- Tabela kontrolnih vektora modula, <i>Module Control Vector</i>
OS	- Operativni sistem, <i>Operating System</i>
FIFO	- Prvi ulaz, prvi izlaz, <i>First In First Out</i>
AVR	- Audio/Video prijemnik, <i>Audio Video Receiver</i>
BBT	- Ispitivanje metodom crne kutije, <i>Black Box Testing</i>

1. Uvod

U okviru ovog rada realizovano je renderovanje audio objekata na platformi sa ograničenim resursima, odnosno digitalnom signal procesoru DSP (*eng. Digital Signal Processor*). Renderovanje audio objekata namenjeno je radu u realnom vremenu, a kako zahtevi podrazumevaju rešenje na jednom procesoru, za implementaciju izabran je četvorojezgarni CS49844 procesor Crystal DSP familije kompanije *Cirrus Logic*. Da bi se zadovoljili resursi potrebni za implementaciju datog algoritma neophodno je iskoristiti tri od četiri jezgra procesora i samim tim prilagoditi algoritam višejezgarnom okruženju. Poslednje jezgro ostaje dostupno korisniku za dodatnu audio obradu.

DSP platforme sa tehnologijom zasnovana na audio objektima ugrađuju se u audio/video prijemnike AVR (*eng. Audio Video Receiver*), kao i u soundbar sisteme. Ubrzani napredak uređaja potrošačke elektronike poslednjih godina nije zaobišao ni oblast audio sistema. Razvoj brojnih uređaja, mobilnih telefona, 3D televizora i drugih uređaja nameće potrebu i za razvojem u načinu emitovanja audio sadržaja. Nove generacije audio tehnologije prate ove promene. Jedna od ključnih osobina novih tehnologija jeste mogućnost prenosa celokupne audio scene umesto njene prezentacije na određenom mestu u prostoru. To znači da audio sadržaj nije vezan za konkretnu postavku zvučnika, čime vernost reprodukcije studijskog sadržaja ne zavisi od konfiguracije zvučnika u sobi. Pristup korišćenja audio objekata za prenos audio sadržaja je ovo omogućio.

Rad je podeljen u sedam poglavlja. U drugom poglavlju rada date su teorijske osnove koje podrazumevaju kratak istorijski razvoj audio tehnologije, uvod u audio tehnologiju zasnovanu na kanalima kao i uvod u audio tehnologiju zasnovanu na 3D objektima, kratak opis realizovanog audio sistema kao i meta podataka, pregled koordinatnih sistema koje služe za opis pozicije u prostoru. Navedeni su i neki od uređaja potrošačke elektronike gde se ovakav sistem koristi. U trećem poglavlju izneto je jedno rešenje renderovanja audio objekata. U četvrtom poglavlju date

su tehničke specifikacije ciljne platforme, detalji implementacije audio sistema koji koristi opisani protokol. Peto poglavlje sadrži rezultate testiranja. U šestom poglavlju ukratko je sažeto sve što je u okviru ovog rada urađeno. Poslednje, sedmo poglavlje, sadrži spisak korišćene literature prilikom izrade ovog rada.

2. Teorijske osnove

Prostorna zvučna reprodukcija počinje sa stereo formatom, odnosno dvo-kanalnom reprodukcijom koju je uveo Blumlein 1931. godine [1]. U isto vreme Harvey Fletcher u Bellovim laboratorijama eksperimentisao je sa usmerenom reprodukcijom, koristeći veliki broj omnidirekcionih mikrofona, gde je svaki povezan sa jednim pojačavačem za odgovarajući zvučnik u slušnoj sobi. Tražeći optimalan broj zvučnika za dovoljno dobar kvalitet reprodukcije, došao je do četiri kanala, a Steinberg i Snow 1934. godine [2] pokazuju da se koristeći tri kanala takođe dobijaju dobri rezultati.

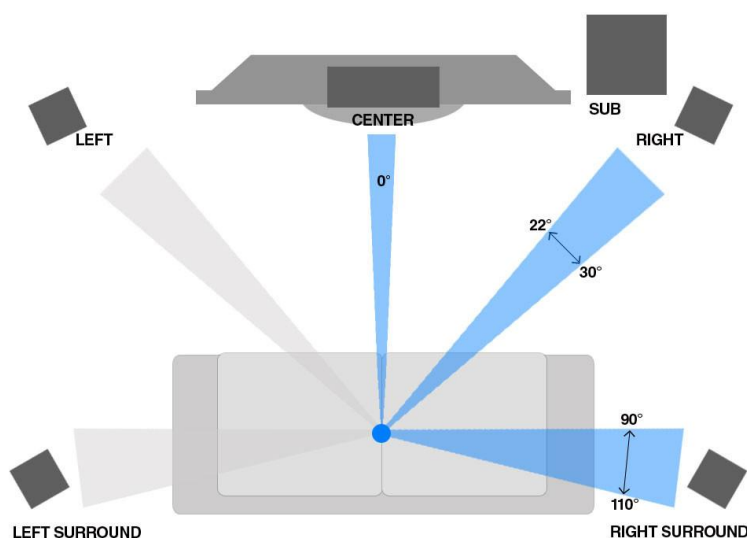
Dostupnost ovakvih sistema za kućnu upotrebu pojavila se 1968. godine kada je Peter Scheiber predstavio kvadrofonski (čtetvorokanalni) sistem. Ipak, ovakav sistem se nije popularizovao iz brojnih razloga: slaba standardizacija sistema, skupa oprema, potreba za dodatnom opremom, nedovoljno razvijena tehnologija.

Sledeći pomak u audio reprodukciji došao je sa Dolby stereo sistemom, 1975. godine. Uključivao je tri prednja (eng. *front*) kanala, odnosno stereo konfiguraciju sa dodatnim centralnim kanalom i jedan zadnji (eng. *rear*) kanal.

Takođe u to vreme, tačnije 1973. Michael Gerzon razvija Ambisonic [3], sistem koji je u potpunosti bio u mogućnosti da reprodukuje prostorni zvuk. Osnovna ideja je bila omogućiti snimanje sa specijalnim mikrofonom, koji je trebao da što kvalitetnije snimi potpunu zvučnu sliku pomoću 4 kanala (npr. sa *Soundfield* mikrofonom). Kao rezultat istraživanja, došlo se do bitnih otkrića: Ambisonic sistem je radio nezavisno od postavke zvučnika, na način da su se pozicije implementirale matrično u dekeru, što se pokazalo dosta praktično. Dalje, efekat prostornosti i lokalizacija su bili stabilni na jako širokom prostoru, tako da se i izvan mreže zvučnika mogla osetiti prostorna zvučna slika koja dolazi iz sistema. Ambisonics je ponudio nešto više od prethodnih tehnika - reprodukciju informacije o visini. Ipak doživeo je loš

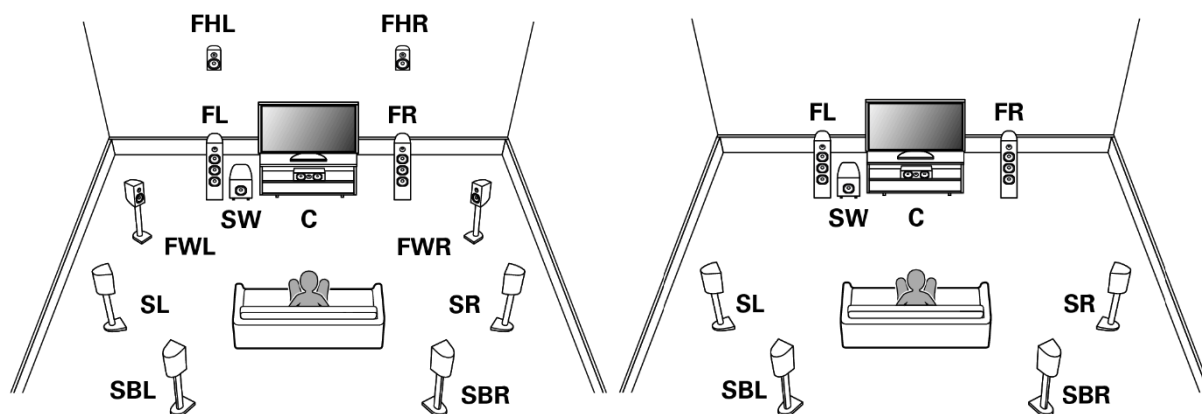
komercijalni uspeh zbog visokih troškova. Cela implementacija uključuje dodatni dekodер, i sam način snimanja i obrade ovakvog sistema je dosta složen.

Dolby stereo kao sistem sa najvećim komercijalnim uspehom se razvija i 1991. godine dovodi do standardizovanog 5.1 sistema [4] (broj pre tačke predstavlja broj zvučnika punog frekvencijskog opsega, dok broj nakon tačke predstavlja broj niskofrekvencijskih zvučnika), prikazanog na Slika 2.1. Sastoji se iz 5 diskretnih kanala, 3 prednja, 2 zadnja i jednog niskofrekventnog kanala.



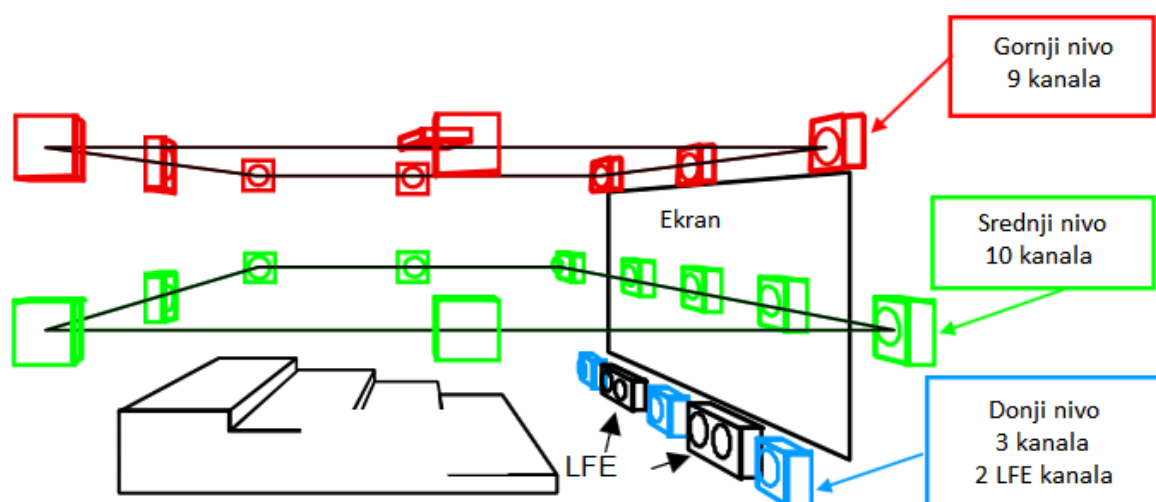
Slika 2.1 Raspored zvučnika standardizovanog 5.1 sistema

U cilju realnijeg zvuka koji dolazi do slušaoca, uvodi se i treća dimenzija dodavanjem podignutih ili spuštenih zvučnika sistemu. Thomlinson Holman 1999. godine razvija 10.2 sistem sa dodatna dva prednja kanala i samo jednim dodatnim zadnjim kanalom. To je zato što je ljudska prostorna percepcija zvuka mnogo preciznija u frontalnom delu, nego iza. Osim toga, on dodaje dubinu zvuku time što izdiže dva frontalna kanala za 45^0 u odnosu na ravan u kojoj se nalazi slušalac. Ovakav sistem je značajno povećao kvalitet reprodukcije, što dovodi do razvijanja i drugih sistema koji uvode treću dimenziju u postavci zvučnika. Neke od konfiguracije ovih 3D sistema zvučnika, 7.1 i 5.1 su prikazane na Slika 2.2.



Slika 2.2 Prikaz konfiguracija zvučnika 7.1 i 5.1 sistema

Na Slika 2.3 prikazan je 22.2 sistem, razvijen od strane Japanske radio-televizijske korporacije [5].



Slika 2.3 Sistem 22.2

I dok se 3D sistem zvučnika pokazao kao kvalitetniji u postizanju prostornog kvaliteta zvuka od 2D sistema, potrebno je obezbediti i interoperabilnost između proizvođača audio sadržaja, proizvođača uređaja i potrošača. Jedno od rešenja za univerzalno i efikasno kodovanje i pružanje kvalitetnog prostornog zvučnog sadržaja jeste 3D objektno orijentisano renderovanje. Osim što je u stanju da obezbedi visoki kvalitet prostornog zvuka, ima mogućnost i da objedini mnoštvo 3D audio formata prihvatajući sve ove formate na ulazu i nudeći visokokvalitetnu reprodukciju na izlazu za veliki broj audio formata od 22.2 pa do nižih, 5.1, stereo, zavisno od originalnog enkodovanog formata. Na ovaj način se postiže i kompatibilnost novih formata sa prethodnim uređajima.

Postoji mnogo različitih načina predstavljanja zvučne scene, sa različitim stepenom tačnosti u zavisnosti od tražene prostorne rezolucije. Na osnovu toga se koristi aproksimaciono predstavljanje ili matematički tačno predstavljanje.

2.1 Audio sistemi zasnovani na više kanala

Najčešći način predstavljanja prostiranja zvuka je kroz skup talasnih formi (*waveforms*), takozvanih kanala. Svaki kanal je dizajniran tako da se reprodukuje na tačno određenom zvučniku u sistemu u odnosu na slušaoca.

U filmu, zvuk koji publika čuje kombinacija je velikog broja pojedinačnih zvučnih elemenata, dijaloga snimljenih na setu, muzike snimljene u studiju itd. Svi ovi elementi distribuirani su na nekoliko audio zapisa i snimljeni u studiju koristeći ogromne miksete i muzičke stanice DAW (*eng. Digital Audio Workstation*) - elektronski sistem dizajniran za snimanje, uređivanje i reprodukciju digitalnog signala, u skladu sa standardizovanom konfiguracijom zvučnika. Krajnji rezultat ovog procesa jeste skup kanala, gde svaki od njih predstavlja sadržaj jednog od zvučnika [5].

Međutim takav signal pravilno će se reprodukovati na samo jednoj izlaznoj konfiguraciji zvučnika odnosno zvučnoj sceni. Naravno postoji mogućnost i reprodukcije takvog signala na drugim zvučnim scenama za koje nije predviđen, ali to zahteva još neke dodatne korake u obradi pa i sam zvuk može izgubiti na kvalitetu. Da bi reprodukovali signal koji sadrži kanale za npr. 5.1 zvučnu scenu na neku zvučnu scenu koja sadrži manje zvučnika (npr. 2.1 sistem) potrebno je izvršiti smanjivanje broja izlaznih kanala (*downmixing*) tog signala. Nasuprot tome ukoliko želimo da reprodukujemo signal pravljen za 5.1 zvučnu scenu na 7.1 potrebno je da izvršimo povećavanje broja izlaznih kanala (*upmixing*) signala. U oba slučaja može doći do gubitka kvaliteta tokom konverzije za drugu zvučnu scenu. Prilikom smanjivanje broja izlaznih kanala signala na manju zvučnu scenu tokom filtriranja može doći do pojave tako zvanih češalj efekata (*comb effect*) na koherentnom sadržaju signala. Tako da je potrebno koristiti pažljivo dizajnirane algoritme za spuštavanje/podizanje signala kako bi ostao visoko kvalitetni signal (*high-quality*) na željenim izlaznim zvučnim scenama [6]-[8].

Danas su audio sistemi bazirani na kanalima veoma rasprostranjeni i vrlo stabilni, iako nalazi svoje granice između raznih formata koji idu ka reprodukciji 3D zvuka.

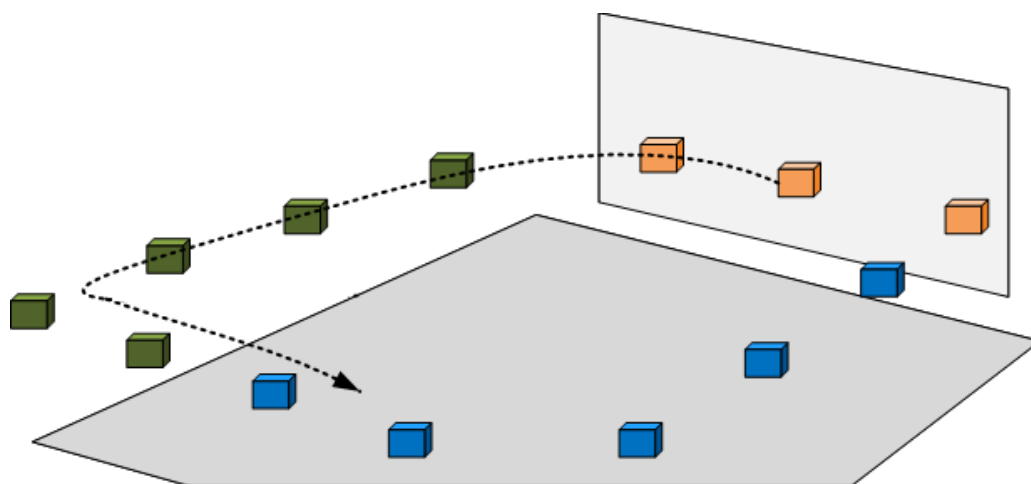
2.2 Audio sistemi zasnovani na objektima

Kod audio sistema zasnovanim na objektima za opisivanje prostiranja zvučne scene koristi se određen broj virtualnih izvora (objekata). Za razliku od kanala čije pozicije su tačno definisane, pozicija objekata u prostoru mogu biti:

- a) Potpuno nezavisna od lokacije prisutnih zvučnika u sistemu
- b) Mogu varirati tokom vremena za modelovanje pokretnih objekata kao sto je avion koji se kreće iznad glave slušaoca.

Obzirom da pozicije objekata ne moraju nužno da se podudaraju sa pozicijom zvučnika u sistemu, objektni signal je u osnovi potrebno renderovati na njegovu ciljanu poziciju. Renderovanje je postavljanj objekata na tačno određene pozicije u 3D prostoru. Gledajući podatke objekat se sastoji iz objektivne talasne forme i odgovarajućeg meta podataka u kojima se nalaze informacije o poziciji objekta, njegovom pojačanju, slabljenju itd. Radi lakšeg razumevanja u svrhu kodovanja kanali se mogu gledati kao objekti koji su postavljeni na fiksnu poziciju na kojoj se nalazi odgovarajući zvučnik. Treba obratiti pažnju da nezavisno od položaja zvučnika objekti se mogu nalaziti na bilo kom mestu u prostoru. U takvim slučajevima objekat će se reprodukovati iz više zvučnika sa različitim nivoima glasnoće. Velika količina audio signala koji sadrže objekte je prisutna danas u studiju i sadrži posebno snimljene muzičke instrumente, naratore, kao i razne zvučne efekte itd. čija pozicija je precizno izračunata uz pomoć raznih programskih alata. Prvi standardizovan audio sistem zasnovan na objektima je bio MPEG-4 [9] 1999 godine.

Ukratko, ideja audio sistema zasnovanim na objektima je da bez obzira na izlaznu konfiguraciju zvučnika (5.1, 7.1, 11.1 itd.) uvek reprodukuje objekat na odgovarajućoj poziciji u prostoru.



Slika 2.4 Primer jedne putanje pokretnog objekta

2.3 Meta podaci

Meta podaci za audio sisteme zasnovane na objektima mora tačno definisati kada, gde i koliko glasno treba objekti da se reprodukuju. Na osnovu informacija iz meta podataka sistem renderuje zvuk i reprodukuje ga na prisutnoj zvučnoj konfiguraciji.

Upotrebom meta podataka prilikom renderinga osiguravamo da će se na predefinisanoj lokaciji objekat pojaviti ili ukoliko se objekat kreće njegova pozicija će se menjati bez obzira na zvučnu konfiguraciju. Shodno tome audio sistemi zasnovanim na objektima omogućuju tačnu

reprodukciju snimljenog zvuka u prostoru, što je bila jedna od ključnih tema prethodnih godina u audio obradi [10]- [11]. Takođe omogućavaju interaktivnost korisnika, zbog toga što se audio objekti obrađuju ponaosob pa korisnik i prioritet samog objekta mogu da utiču na njegovo renderovanje. Meta podaci kontrolišu i nivo interaktivnosti odnosno kategorije na koje korisnik može da utiče na objekte i koliko može da utiče. Na primer ograničenja mogu da budu vezana za promenu položaja objekata do 30 stepeni u svim pravcima ili maksimalnu jačinu objekta. Takođe može i da onemogući korisniku interaktivnost za određeni objekat.

Meta podaci mogu biti statički ili dinamički. Statički meta podaci mogu da se pojave samo na početku ulaznog komprimovanog sadržaja i nigde više i mogu da budu aktivni duž celog ulaznog test vektora. U statičkim meta podacima spadaju npr. jezik na kom se govori u ulaznom komprimovanom sadržaju i da li je njegovo stanje podrazumevano ili ne. Dinamički meta podaci sadrže informacije koje se menjaju tokom vremena kao i samu kontrolu procesa renderovanja. U njih spadaju pozicije objekata, pojačanje objekata, virtualni izvori korišćeni za renderovanje objekata itd.

2.4 Ambisonik

Ambisonic sistem prvog reda razvio je Gerzon [12] u sedamdesetim godinama prošlog veka i nazvao ga B-format. Zvučna slika snima se specijalnim mikrofonom (*Soundfield* mikrofonom) ili sa jednim omnidirekcionim mikrofonom i tri gradijentna mikrofona, od kojih je svaki postavljen u jedan od tri zamišljena koordinatna sistema (X, Y, Z). Princip je sledeći: od 4 kanala koja se snimaju, jedan je mono (W), dobijen preko omnidirekcionalnog mikrofona (ili superpozicijom svih kapsula *Soundfield* mikrofona), a ostala tri su snimljena kao razlike pritisaka na odgovarajućim osama, na primer kanal X je razlika u gradijentu „ispred“, odnosno „iza“ referentne tačke na x-osi. Analogno je i za kanale Y i Z. Daljim dekodovanjem snimka moguće je za odgovarajuću postavku zvučnika dobiti reprodukciju koja odgovara zamišljenoj: očuvane su lokalizacija i karakteristike snimka i prostorije.

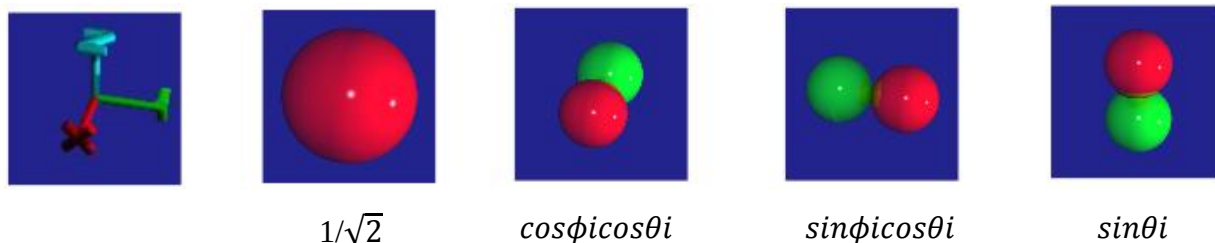
$$W = \frac{1}{k} \sum_{i=1}^n Si \left[\frac{1}{\sqrt{(2)}} \right] \quad \text{omnidirekciona informacija} \quad (10)$$

$$X = \frac{1}{k} \sum_{i=1}^n Si [\cos\phi_i \cos\theta_i] \quad \text{informacija x-pravca} \quad (11)$$

$$Y = \frac{1}{k} \sum_{i=1}^n Si [\sin\phi_i \cos\theta_i] \quad \text{informacija y-pravca} \quad (12)$$

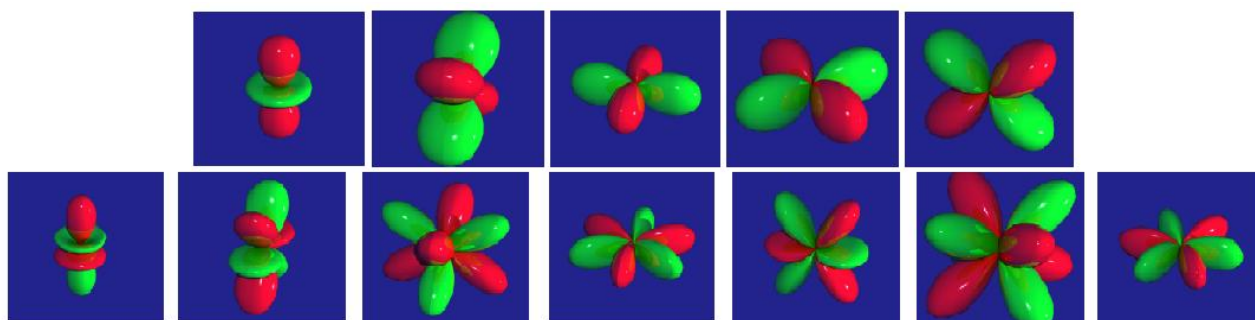
$$Z = \frac{1}{k} \sum_{i=1}^n S_i [\sin\theta_i] \quad \text{informacija z-pravca} \quad (13)$$

Fizički posmatrano, ovo predstavlja dekompoziciju zvučnog polja po sfernim koordinatama. Na Sliku 2.5 prikazani su sferni harmonici nultog i prvog reda koje koriste ambisonici prvog reda.



Slika 2.5 Enkodovane funkcije ambisonika prvog reda

Princip ovih sistema proširio se i na više redove, povećavajući pritom prostor u kojem se zvučno polje precizno reprodukuje kao i ukupni kvalitet lokalizacije. Međutim, sve većim redom ambisonik snimci rastu a sa njima i broj potrebnih kanala nužnih za reprodukciju. Tako je za drugi red potrebno 5 dodatnih kanala, a za treći red još dodatnih 7 kanala [12].



Slika 2.6 Enkodovane funkcije ambisonika drugog reda

Na Sliku 2.6 vidimo sferne harmonike drugog i trećeg reda. Svakim novim redom harmonika, broj kanala potreban za reprodukciju raste po kvadratnoj zavisnosti:

$$N = (M + 1)^2 \quad (14)$$

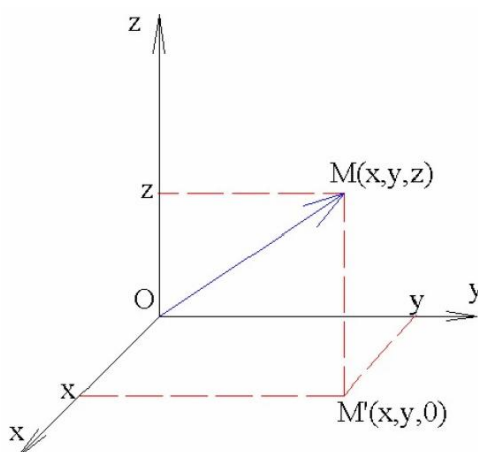
gde je N broj kanala i M red harmonika.

2.5 Koordinatni sistemi

Koordinate objekata koje se nalaze u meta podacima predstavljeni su u jednom od dva koordinatna sistema: sferni koordinatni sistem ili kartezijev (Dekartov) koordinatni sistem.

Prilikom renderovanja ukoliko je potrebno vrši se transformacija iz jednog koordinatnog sistema u drugi.

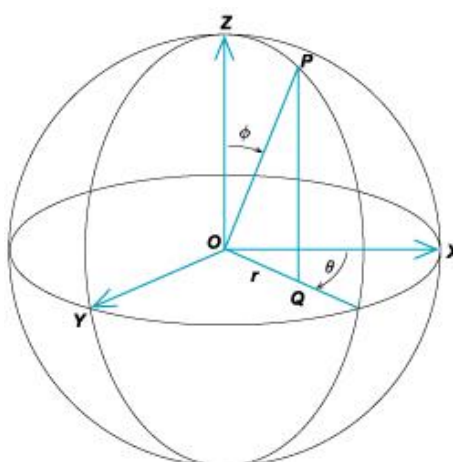
Kartezijev koordinatni sistem sastoji se iz 3 koordinatne ose (x-osa, y-osa, z-osa) i 3 koordinatne ravni (xy ravan, xz ravan i yz ravan) što je prikazano na Slika 2.7 Kartezijanski koordinatni. Preseci dveju koordinatnih ravni su koordinatne ose. Zajednička tačka za sve koordinatne ravni kao i za sve tri koordinatne ose je koordinatni početak (O). Koordinatne ose su orijentisane i imaju svoj pozitivan i negativan smer.



Slika 2.7 Kartezijanski koordinatni

U prostoru se može uočiti bilo koja tačka M. Tački M odgovara samo jedna uređena trojka brojeva (x, y, z) koja predstavlja koordinate te tačke. Koordinate tačaka na x-osi su oblika $(x, 0, 0)$, na y-osi tačke imaju koordinate $(0, y, 0)$, a na z-osi $(0, 0, z)$.

Sferni koordinatni sistem (prikazan na slici Slika 2.8) je trodimenzionalan koordinatni sistem kojim se opisuje položaj tačke u prostoru. Koristi tri koordinate (r, ϕ, θ) : r – udaljenost tačke od koordinatnog početka, ϕ (zenit, u daljem tekstu *elevation*) –, ugao koji prava koja spaja tačku sa koordinatnim početkom zaklapa s pozitivnim delom z – ose, θ (azimut) – ugao iste prave sa pozitivnim delom x-ose.



Slika 2.8 Sferni koordinatni sistem

Transformacije iz jednog koordinatnog sistema u drugi se računaju po sledećim formulama:

- iz sfernog u kartezijev $x = r \sin\theta \cos\phi$ $y = r \sin\theta \sin\phi$ $z = r \cos\theta$

- iz kartezijevog u sferni $r = \sqrt{(x^2 + y^2 + z^2)}$, $\theta = \tan^{-1}(\frac{z}{\sqrt{(x^2 + y^2)}})$, $\phi = \tan^{-1}(\frac{y}{x})$

2.6 Prijemnik audio i video sadržaja

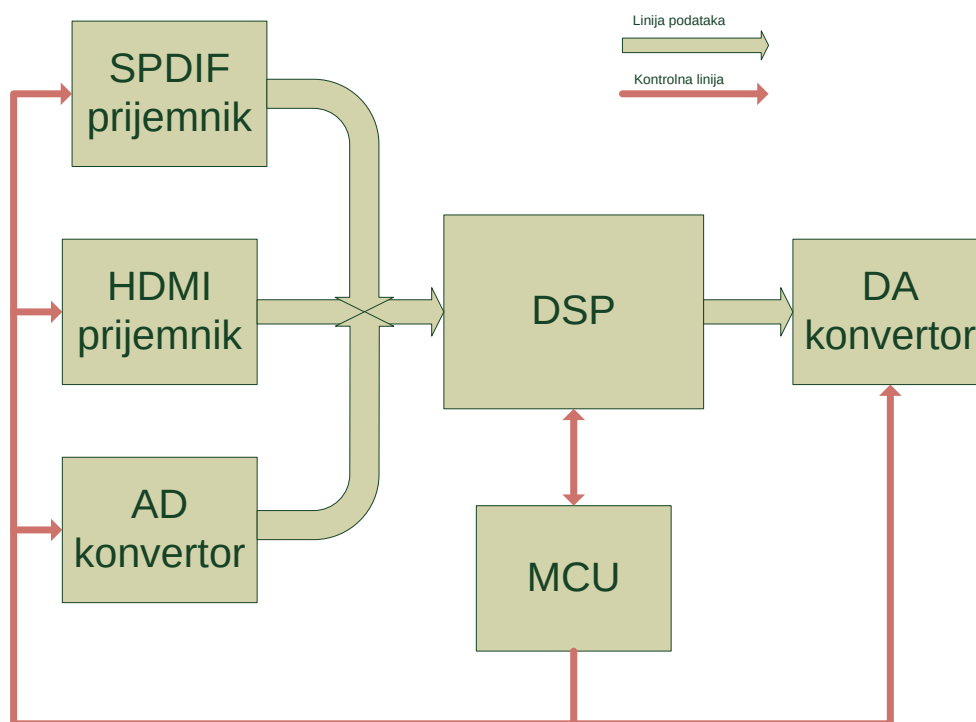
Razvojem potrošačke elektronike i informacionih tehnologija, a sa njima i filmske i muzičke industrije, došlo je do pojave rešenja tzv. kućnih bioskopa. Sa njihovim razvojem i željom da se funkcija više različitih uređaja objedini u jedan, nastao je prijemnik audio i video sadržaja u daljem tekstu AVR (*Audio-Video Receiver*).

Glavna uloga AVR-a je prijem audio i video signala sa raznih izvora i obrada tog signala kako bi se mogli proslediti na zvučnike i displej (TV, projektor). Izvor signala mogu biti razni uređaji, poput satelitskog prijemnika, TV, radio prijemnika, DVD plejera, Blue-Ray plejera, video rekordera i konzola za video igre.

U audio terminologiji, pojam prijemnika podrazumeva uređaj koji objedinjuje audio pojačavač sa radio prijemnikom. AVR proširuje mogućnosti klasičnog prijemnika sa mogućnostima prijema i obrade različitih vrsta digitalnih i analognih audio signala, dok uporedo sa tim omogućava propuštanje video sadržaja na odgovarajući izlaz. U poslednjih par godina postoji tendencija da AVR-ovi takođe obrađuju i video signal.

Svaki AVR uređaj poseduje mogućnost dekodovanja jednog ili više tipova audio formata koji se nalaze trenutno na tržištu. Tendencija je proizvođača da u ponudi imaju uređaje koji mogu dekodovati sve postojeće tehnologije, dok se te tehnologije stalno unapređuju, nudeći veći broj izlaznih kanala, verniju reprodukciju zvuka, itd., što iziskuje sve moćnije sisteme za obradu audio signala.

Prijemnik audio i video sadržaja prima analogni ili digitalni signal na ulazu, obradi ga (po potrebi dekoduje), pa tako obrađen signal da pretvori u analognu veličinu, koja se dalje vodi na audio pojačavač, na čijem se izlazu dobija signal za pobudu zvučnika. Blok šema sistema je data na Slika 2.9.



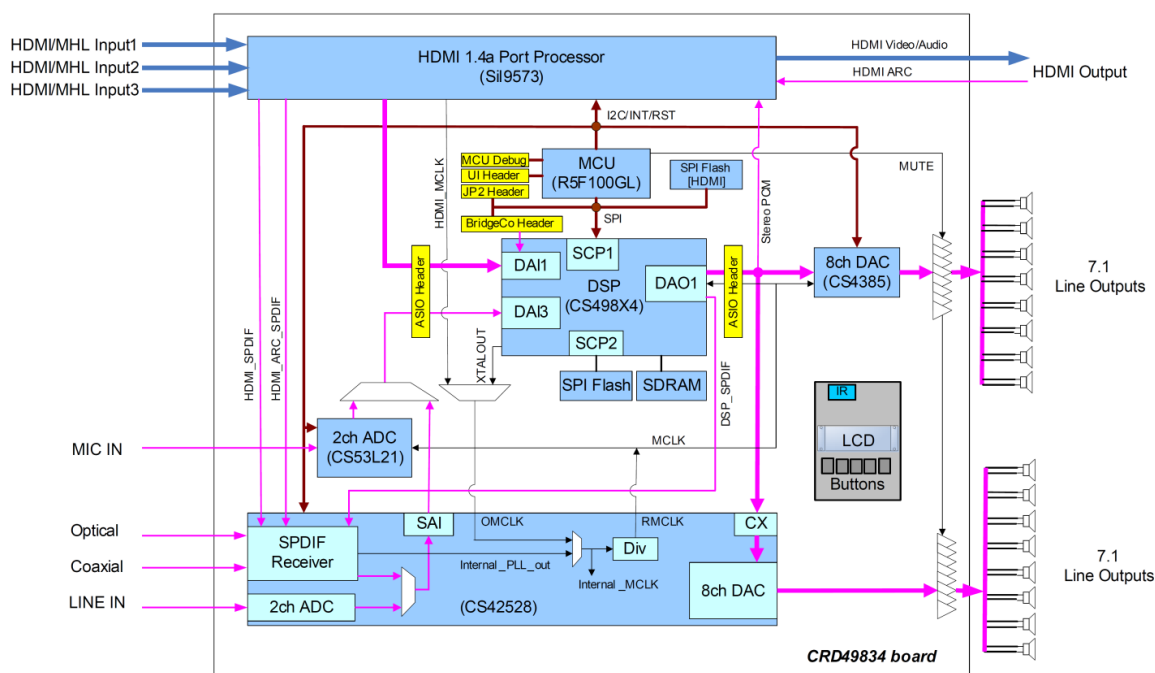
Slika 2.9 Uprošćena blok šema sistema prijemnika audio i video sadržaja

Razvojem tehnologije, analogna obrada signala je u potpunosti potisnuta digitalnom obradom signala, zbog mnogobrojnih prednosti digitalne nad analognom obradom. Jedina analogna obrada signala koja se vrši u našem sistemu je pojačavanje na izlaznom stepenu (audio pojačavač), kako bi se korisni signal mogao sprovesti na zvučnike.

Centralna jedinica obrade audio signala celog sistema predstavlja DSP, sa implementiranim opisanim audio podsistemom, koga nadgleda zaseban sistemski mikroračunar (eng. *host*). Kako je DSP sistem sa ograničenim resursima, zbog efikasnijeg rada, upravljanje njegovim radom, učitavanje programa sa perifernih jedinica, podešavanje sistema i sl. obavlja mikrokontroler.

Dakle, prijemnik audio i video sadržaja ima ulogu da primi analogni ili digitalni signal na ulazu, obradi ga (po potrebi dekoduje), pa tako obrađen signal da pretvori u analognu veličinu, koja se dalje vodi na audio pojačavač, na čijem se izlazu dobija signal za pobudu zvučnika. Blok dijagram sistema, kao i izgled samog uređaja dati su na Slika 2.10.





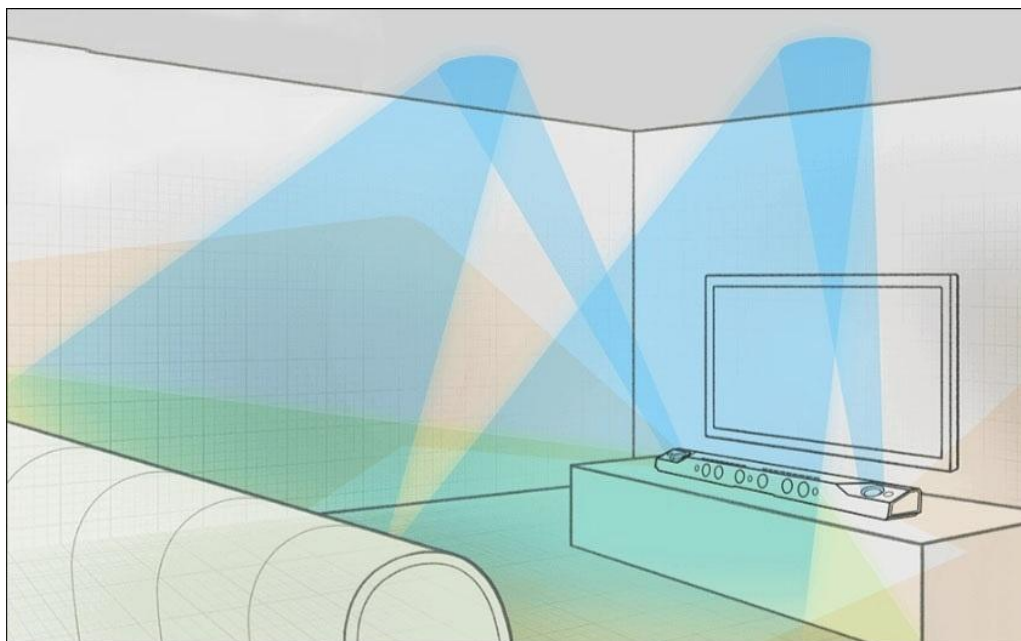
Slika 2.10 Prikaz izgleda i blok dijagram prijemnika audio-video signala zasnovanog na CS49844 procesoru

2.7 Soundbar

Ukoliko se gradi prostorija namenjena za kućni bioskop, tada postoji veća sloboda u izboru željenih zvučnika, kako po pitanju veličine ili estetike, tako i po pitanju 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. Dizajn ovakvog zvučnika omogućava lako postavljanje ispod ili iznad ekrana računara, TV uređaja ili kućnog bioskopa.

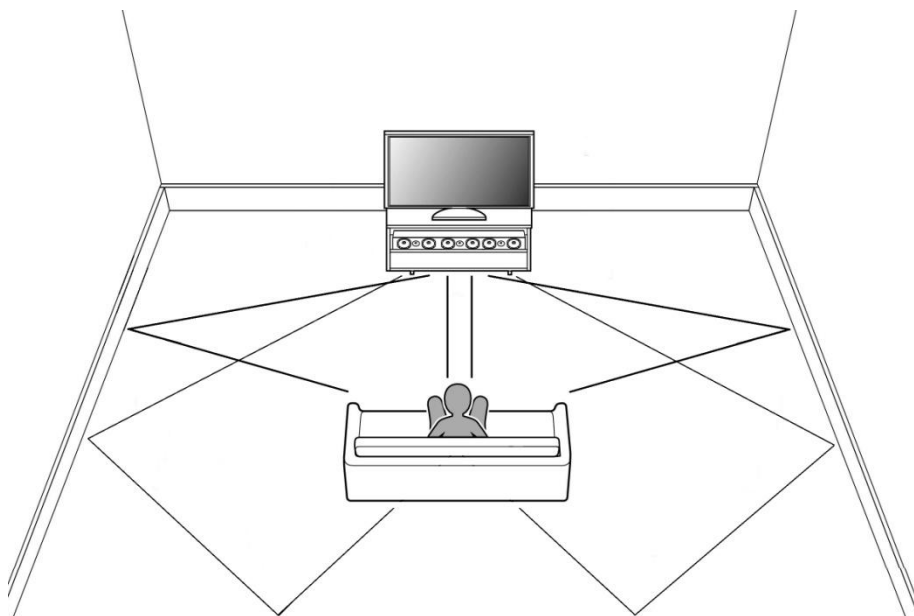
Soundbar je specijalna vrsta kućišta zvučnika koja od jedne tačke izvora stvara *surround* efekat. Unutar kućišta nalaze se zvučnici punog opsega, koji su pozicionirani na određeni način, tako da emitovanjem audio signala i njegovim odbijanjem o zid prostorije, zvuk dolazi do slušaoca stvarajući surround efekat. Ilustracija principa rada jednog soundbar uređaja prikazana je na Slika 2.11



Slika 2.11 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 soundbar uređaji (Slika 2.12) koji surround efekat stvaraju reflektovanjem audio signala o zid prostorije, postavljanjem zvučnika punog opsega pod različitim uglovima unutar kućišta.



Slika 2.12 Ilustracija dobijanja surround efekta

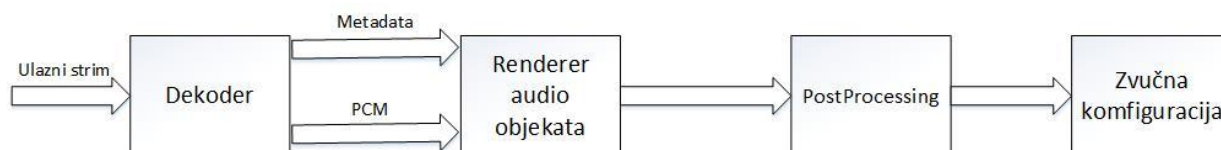
3. Koncept rešenja

Upotreba objekata omogućava tačnu prostornu reprodukciju zvučnog signala nezavisno od konfiguracije zvučnika. Prednost objekata je da ne zavise od konfiguracije zvučnika samog sistema. Uz pomoć renderovanja objekata oni će biti postavljeni na približno iste pozicije na najbolji mogući način bez obzira na samu konfiguraciju zvučnika. Na samom tržištu koncept sa objektima je izuzetno dobro prihvaćen. Jedan od razloga je i činjenica da nije potrebno menjati zvučnu konfiguraciju da bi sve zvučalo kako je predviđeno.

U okviru ovog poglavlja dat je kratak koncept celog sistema sa audio objektima koji je implementiran na platformi sa ograničenim resursima kao i koncept renderovanja objekata.

3.1 Objektni sistem

Na Sliku 3.1 predstavljen je blok šema sistema za obradu zvuka zasnovanog na



Slika 3.1 Blok šema protoka zvuka

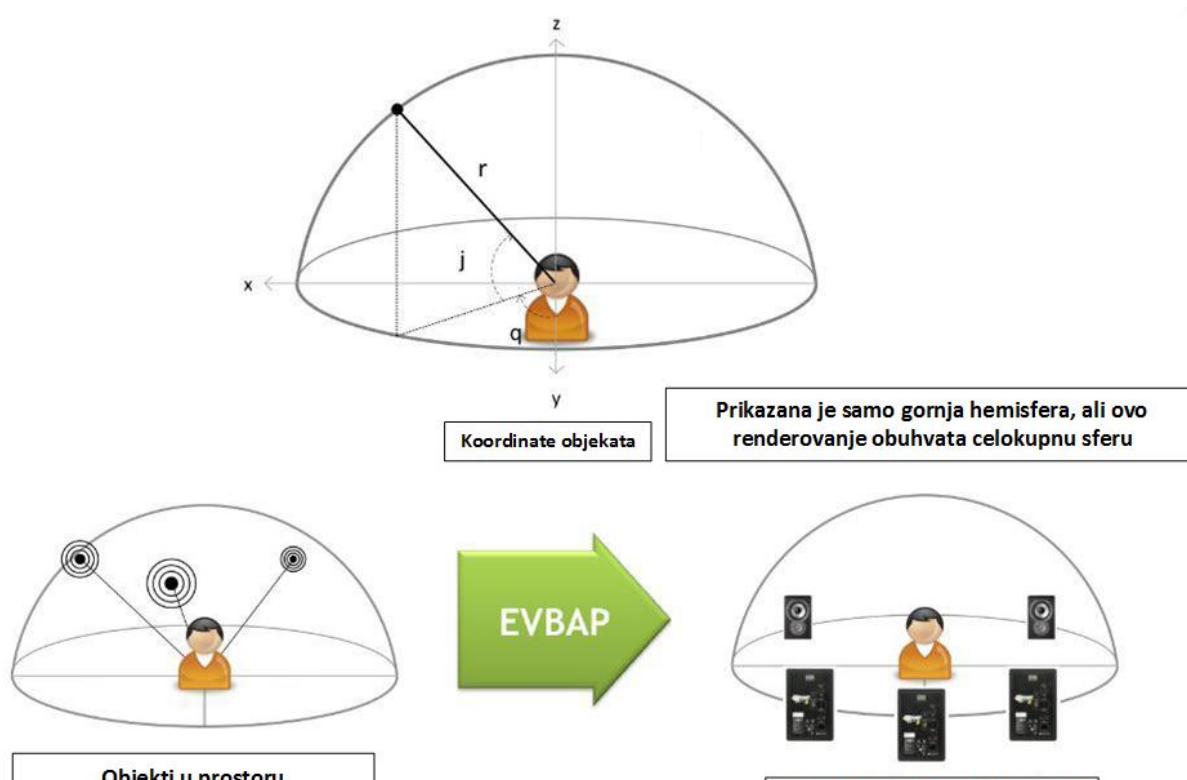
objektima.

Na ulazu u sistem dovodi se ulazni test vektor koji sadrži enkodovane kanale, objekte, meta podatke itd. Ulazni test vector ulazi u dekodler koji vrši njegovo dekodovanje i parsiranje meta podataka. Dekoder obrađene PCM odbirke stavlja u izlazne nizove (*I/O buffer*) koji se uz pomoć operativnog sistema prenose u Renderer audio objekata. U drugom nizu (*buffer*) nalaze se meta podaci isparsirani iz ulaznog test vektora. PCM niz se sastoji iz dva dela. Prvi deo sadrži do 13

kanala, dok u drugi deo može da se smesti najviše 32 objekata. Renderer audio objekata biće detaljnije objašnjen u sledećem poglavlju. Iz njega se preko 13-to kanalnog niza šalju umiksani kanali i objekti u PostProcessing modul. Miksovanje kanala i objekata se radi u cilju postizanja kompatibilnosti novih snimaka sa starijim uređajima. PostProcessing modul zadužen je za povećanje broja izlaznih kanala, smanjivanje broja izlaznih kanala u zavisnosti od izlazne zvučne konfiguracije i samog sadržaja ulaznog test vektora. Ovde se vrše i razna filtriranja signala u cilju postizanja što jasnijeg i boljeg zvuka. Izlazne zvučne konfiguracije mogu biti sistemi 2.0, 2.1, 5.1, 7.1, 11.1, 11.2 ili soundbar.

3.2 Renderovanje objekata u prostoru

Renderer audio objekata je modul zadužen za raspodelu objekata u 3D prostoru. Glavni posao ovog modula je da na tačno definisanu poziciju u 3D prostoru doda odnosno ukloni objekat. Uklanjanje objekata vrši se pretežno zbog kompatibilnosti sa starijim uređajima. Svaki objekat dolazi sa meta podacima koji diktiraju njegovu preciznu poziciju u prostoru. Meta podaci su azimut, zenit, udaljenost objekta, pojačanje, signalna informacija koja fiksira objekat za najbliži zvučnik itd. Renderer audio objekata podržava objekte koji su fiksirani u jednoj poziciji kao i one koji se kreću u zavisnosti od tipa njenih meta podataka. Modul za renderovanje objekata koristi sfernu reprezentaciju sistema. Ukoliko je predviđeno od strane enkodera da se 3D prostor gleda kao kocka odnosno da su kordinati objekata pretstavljeni u kartezijskom sistemu vršiće se



Slika 3.2 Koncept renderovanja objekata

konverzija u sferni sistem i 3D prostor će se gledati kao sfera. Algoritam za renderovanje objekata koji je implementiran u okviru ovog rada je pomeranje amplitude vektora.

3.2.1 VBAP (Pomeranje amplitude vektora u 3D prostoru)

VBAP (*Vector Base Amplitude Panning*) je metoda za pozicioniranje izvora zvuka u proizvoljnim pravcima u dvodimenzionalnim i trodimenzionalnim sistemima sa proizvoljnim brojem zvučnika. Nezavisna je od položaja zvučnika u sistemu i pruža maksimalnu moguću preciznost za datu konfiguraciju zvučnika.

U dvodimenzionalnim konfiguracijama zvučnika, odnosno sistemima sa dva ili više zvučnika smeštenih u istoj, najčešće horizontalnoj ravni sa slušaocem, biraju se po dva zvučnika u paru na kojima će se primeniti audio signal u određenom trenutku i nazivaju se aktivni luk. Svaki od zvučnika može biti deo dva aktivna luka. Ako su jedinični vektori l_1 i l_2 usmereni od pozicije slušaoca do pozicije zvučnika (Slika 3.3) [13], vektor zvučnog izvora p može se opisati kao njihova linearna kombinacija:

$$p = g_1 l_1 + g_2 l_2 \quad (1)$$

gde su g_1 i g_2 faktori pojačanja datih zvučnika. Ako ove vektore prikažemo u matičnom obliku

$$g = [g_1 \ g_2]^T, \quad L_{12} = [l_1 \ l_2], \quad (2)$$

tada se pojačanja mogu izraziti na sledeći način:

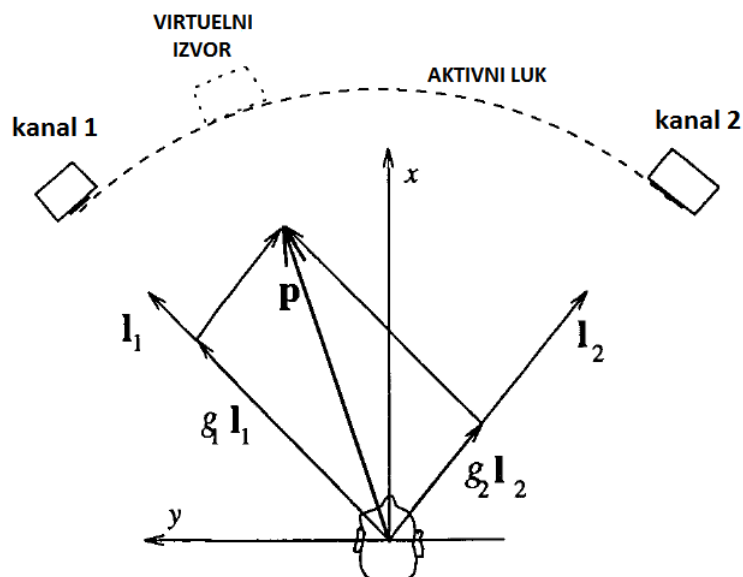
$$g = p^T L_{12}^{-1}. \quad (3)$$

Snaga zvuka je konstantna vrednost, što se može zapisati u sledećem obliku:

$$g_1^2 + g_2^2 = C. \quad (4)$$

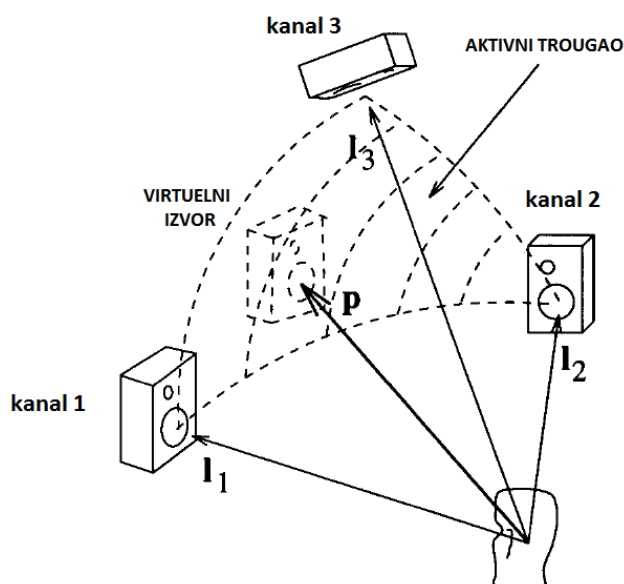
Konstanta C je pozitivna vrednost i može se posmatrati kao kontrola jačine zvuka koji se reprodukuje.

Dakle ova pojačanja nakon normalizacije jesu pojačanja signala koja će se primeniti na odgovarajući zvučnik.



Slika 3.3 Definisanje vektora u stereo konfiguraciji

Trodimenzijsni VBAP koristi se za trodimenzionalne sisteme, odnosno sisteme gde svi zvučnici ne moraju da se nalaze u istoj horizontalnoj ravni, već i iznad i ispod nje. Na ovaj način omogućena je lokalizacija zvučnog signala i po vertikali. U ovom slučaju biraju se površine određene sa po tri zvučnika na kojima se izvor zvuka može nalaziti - dva iz iste horizontalne ravni, dok je treći zvučnik van nje. Ovakva trojka naziva se aktivni trougao. Svaki od zvučnika može pripadati većem broju ovakvih trojki. Aktivni trouglovi ne smeju da se seku međusobno i treba ih izabrati tako da se postigne maksimalna moguća tačnost lokalizacije u svakom pravcu. Jedan ovakav sistem prikazan je na Slika 3.4.



Slika 3.4 Konfiguracija trodimenzionalnog VBAP-a.

Zvučnici formiraju trougao u kojem virtuelni izvor može biti pozicioniran

Vektor izvora zvuka se definiše kao linearna kombinacija vektora tri zvučnika l_1 , l_2 i l_3

$$p = g_1 l_1 + g_2 l_2 + g_3 l_3. \quad (5)$$

Faktori pojačanja izvode se iz prethodne jednačine:

$$g = p^T L_{123}^{-1}, \quad (6)$$

gde su

$$g = [g_1 \ g_2 \ g_3]^T, \ L_{123} = [l_1 \ l_2 \ l_3]. \quad (7)$$

Kako je snaga zvuka konstanta, analogno dvodimenzionalnom sistemu može se zapisati:

$$g_1^2 + g_2^2 + g_3^2 = C. \quad (8)$$

Nakon normalizacije:

$$g_{normalizovano} = \frac{\sqrt{C}}{\sqrt{(g_1)^2 + (g_2)^2 + (g_3)^2}} \quad (9)$$

ova pojačanja koriste se kao umnošci signala prilikom njihove primene na odgovarajuće zvučnike.

VBAP ima tri važna svojstva:

1. Ako je izvor zvuka lociran u istom pravcu kao i neki od zvučnika, signal ostaje reprodukovan samo na tom zvučniku, što obezbeđuje maksimalnu oštrinu zvuka.
2. Ako se izvor zvuka nalazi na liniji koja povezuje dva zvučnika, zvuk se primenjuje samo na taj par, prateći tangentin zakon. Faktor pojačanja trećeg zvučnika je nula.
3. Ako se izvor nalazi u centru aktivnog trougla, faktori pojačanja za svaki od zvučnika su jednaki.

Ove osobine ukazuju na to da VBAP reprodukuje izvore zvuka što oštrije i preciznije je moguće sa datom konfiguracijom zvučnika.

4. Implementacija

Ovo poglavlje sadrži detalje implementacije datog rešenja na jednom DSP procesoru pa je za ciljnu platformu odabran četvorोजezgarni CS49844 procesor Crystal DSP familije kompanije *Cirrus Logic* zbog svoje arhitekture i resursa. Dato je i objašnjenje nekoliko bitnijih funkcija, iskorišćenost resursa i tehničke osobine same DSP platforme na kojoj je algoritam implementiran.

4.1 Ciljna platforme

Ciljna platforma ove implementacije je 32-bitni audio DSP procesor CS49844 [14] kompanije Cirrus Logic (Slika 4.1) unapređene Harvard arhitekture [15] sa dve memorijske zone za podatke i jednom memorijskom zonom za smeštanje programskog koda. Ovakva organizacija omogućava istovremeni pristup instrukcijama i podacima iz memorije, čime se znatno ubrzava rad procesora i postiže velika iskorišćenost resursa datog procesora i do čak šest instrukcija u jednom taktu – dve pomnoži i saberi MAC (*eng. Multiply and Accumulate*) instrukcije, dva čitanja ili upisa iz/u memoriju i dva povećanja/umanjenja adresnih registara.

Procesor poseduje četiri jezgra, pri čemu svaki od njih obezbeđuje 300 miliona instrukcija u sekundi (MIPS) procesne moći, 60 kilo reči memorije u svakoj od memorijskih zona, gde je reč dužine 32 bita. Svako jezgro radi na taktu od 300 MHz. U Tabela 4.1 dat je pregled ovih resursa.

Takođe, svako jezgro sadrži osam 32-bitnih registara opšte namene (x0-x3 i y0-y3), osam 72-bitnih registara (tzv. akumulatora, a0-a3 i b0-b3), namenjenih za smeštanje međurezultata, dvostruku preciznost i dr. Svako jezgro takođe sadrži dvanaest tzv. indeksnih registara (i0-i11), veličine 16 bita, namenjenih za adresiranje kod čitanja i upisa iz/u memoriju, kao i za druge namene, kao što su brojači iteracija petlji i slično. Svakom od indeksnih registara odgovara i tzv. moduo registar, koji omogućava da odgovarajući registar radi u moduo režimu (korisno kod

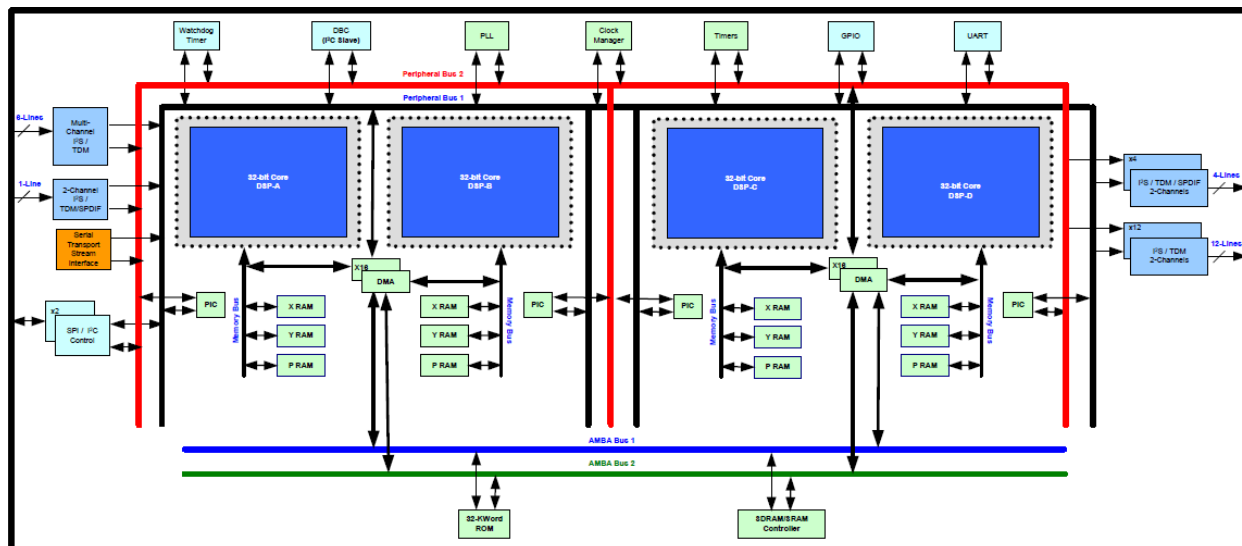
kružnih bafera) kao i bit-reverznom režimu (vrlo korisno kod algoritama koji zahtevaju izuzetno brzo izvršenje, kao što su brze furijeove transformacije i dr.).

Svako od jezgara sadrži tzv. SRS jedinicu (*eng. Shift Round Saturate*) koja upravlja aritmetičkim pomeranjem, zaokruživanjem i saturacijom između akumulatora i registara, kako joj i samo ime kaže. Platforma sadrži i jedinicu za direktni pristup memoriji, DMA (*eng. Direct Memory Access*), koji služi kako za prenos podataka između jezgara, tako i za prenos podataka između spoljne memorije i svakog od četiri jezgra.

Povezivanje sa periferijama omogućeno je preko SPDIF (*eng. Sony Philips Digital InterFace*), HDMI (*eng. High Definition Multimedia Interface*) kao i analognih ulaza-izlaza.

	DSP A	DSP B	DSP C	DSP D
Procesorsko vreme	300 MIPS	300 MIPS	300 MIPS	300 MIPS
Memorija za podatke (X memorija)	60 kilo reči	60 kilo reči	60 kilo reči	60 kilo reči
Memorija za podatke (Y memorija)	60 kilo reči	60 kilo reči	60 kilo reči	60 kilo reči
Programska memorija (P memorija)	60 kilo reči	60 kilo reči	60 kilo reči	60 kilo reči

Tabela 4.1 Količina raspoloživih resursa ciljne platforme



Slika 4.1 Blok dijagram CS49844 čipa

4.2 Razvoj sistema na DSP platformi

Polazna osnova implementacije na ciljnoj platformi bio je komercijalni algoritam implementiran u programskom jeziku višeg nivoa C. Prvi korak je upoznavanje i analiza datog

referentnog koda, zatim procena potrebnih resursa, kako u pogledu memorije, tako i u pogledu procesorskog vremena.

Treba imati u vidu da se prilikom pisanja referentnog koda najviše obraća pažnja na ispravnost samog algoritma, dok je za pisanje koda u izvršnom programskom jeziku assembleru neophodno voditi računa o ograničenjima nametnutim odabirom ciljne platforme.

Prvi problem su, dakle, ograničenja memorijskih resursa i procesorskog vremena, te se već tokom samog početka pisanja koda uvode odgovarajuće modifikacije i algoritamske optimizacije referentnog koda koje omogućavaju efikasniju implementaciju na DSP platformi.

Prvi korak jeste funkcionalna optimizacija koda, koja podrazumeva organizaciju podataka, prilagođavanje pristupa podacima, optimizaciju programskih petlji.

Smeštanje promenljivih u registre je ograničeno brojem registara koji je na DSP procesoru mali (dve grupe od po osam registara), dok je njihovo smeštanje na stek ograničeno ne samo veličinom steka, već i povećanim brojem instrukcija obrade u odnosu na neposredno smeštanje u memoriju podataka (jer je potrebno pristupiti steku i premestiti promenljive u memoriju ili registre pre obrade podataka). Zato je dobra praksa korišćenje globalnih ili lokalnih promenljivih. Primer za to jeste prosleđivanje argumenata kod funkcije obrade. U referentnom kodu se često koriste velike strukture u kojima se opisuje trenutno stanje aplikacije ili pojedinačnih procesnih modula. Ponekad se takve strukture prosleđuju kao parametri funkcija, a češće se prosleđuje pokazivač na strukturu. Ovakvo prosleđivanje rezultuje generisanjem neefikasnog koda i samim tim ga treba izbegavati. Poželjno je uočiti takve strukture i napraviti njihove globalne instance. Kada se funkciji prosleđuje pokazivač na strukturu, potrebno je obratiti pažnju da li je u kodu realizovana jedna ili više struktura. Samo ukoliko se koristi ista struktura, moguće joj je neposredno pristupiti u telu funkcije. U asemblerskoj implementaciji argumenti mogu biti prosleđeni na nekoliko načina: preko registara, preko strukture čiji su elementi argumenti funkcije, korišćenjem globalnih i lokalnih promenljivih. Ukoliko se argumenti funkciji prosleđuju preko registara potrebno je popuniti određene registre pre poziva funkcije. Ukoliko postoji dosta argumenata koje je potrebno proslediti najčešće se zauzme struktura sa svim potrebnim argumentima tako da se u samoj funkciji pristupa određenom polju strukture kada je to potrebno. Pre poziva funkcije potrebno je popuniti strukturu odgovarajućim podacima. Ukoliko je moguće poželjno je koristiti lokalne i globalne promenljive koje se jednom popune i koriste za svaki poziv funkcije. Korisno je koristiti konstante gde god je to moguće.

Jedno od hardverskih proširenja digitalnih signal procesora je i jedinica za generisanje adresa podataka. Adresni generator je u stanju da obavi izvestan broj aritmetičkih operacija i da generiše adresu na kojoj se nalazi podatak. Po pravilu podržava kružno ili moduo adresiranje, kao i režim bit-obrnuto pristupa koji se koristi kod FFT-a. Iz ovog razloga operacije pristupa

podacima potrebno je prilagoditi radu adresnog generatora. Ovo prilagođenje podrazumeva zamenu posrednog pristupa elementima niza preko indeksa, pristupom preko pokazivača na element.

Najveći deo procesorskih ciklusa tokom izvršavanja DSP aplikacija troši se na izvršavanje programskih petlji. Ušteda jednog instrukcionog ciklusa unutar tela petlje dovodi do uštede N instrukcijskih ciklusa, gde N predstavlja broj iteracija petlje po jedinici obrade. Deo koda unutar programskih petlji ponekad je nezavisan od trenutne iteracije ili čitave petlje, pa nema potrebe da se izvršava više puta. Ovaj kod naziva se nezavisni, odnosno invarijantni kod. Primer ovakvog koda jeste dodeljivanje konstantne vrednosti nekom resursu. Umesto da se ta dodela obavlja u svakoj iteraciji petlje, može se izvršiti samo jednom, pre ulaska u petlju. Ovaj pristup produžava životni vek promenljivih i dodatno opterećuje resurse, ali ubrzava izvršenje celokupnog koda jer skraćuje telo petlji.

Kako C radi u aritmetici pokretnog zareza, a DSP platforma u aritmetici nepokretnog zareza, ograničena na preciznost od 32 bita, drugi korak predstavlja prilagođenje koda aritmetici DSP-a.

Jedan od problema razlike u aritmetici javlja se kod koeficijenata čija je vrednost van zadanog opsega $[-1, 1)$. Konstante koje su van ovog opsega je potrebno skalirati, da bi se omogućile aritmetičke operacije nad ispravnim vrednostima. Konstante koje se nalaze u intervalu od $[1, 2)$ treba da pomerimo za 1 mesto u desno (ekvivalentno deljenju sa 2), konstante koje se nalaze u intervalu od $[2, 3)$ treba pomeriti za 2 mesta u desno (ekvivalentno deljenju sa 4) i sl.

Osim ovoga, može da dođe do prekoračenja maksimalne vrednosti koja može da se smesti u akumulator. Kako bi se ovo izbeglo, moguće je uvesti bezbednosni opseg (*headroom*), odnosno, skalirati vrednosti signala na ulazu u blok obrade (na opseg manji od maksimalne veličine tipova za podatke), a na izlazu iz bloka izvršiti inverzno skaliranje.

Zaseban problem jeste pojava da je vrednost konstante u nepokretnom zarezu isuviše mala da bi bila predstavljena sa datim tipom podataka. Preciznost tipa u nepokretnom zarezu može se predstaviti rastojanjem Δ između bilo koja dva susedna broja predstavljena ovim tipom. Ako je bilo koji tip u nepokretnom zarezu predstavljen u formatu $\langle a.b \rangle$, gde je a broj bita kojim je predstavljen celobrojni deo broja, a b broj bita kojim je predstavljen razlomljeni deo broja, tada je $\Delta = 2^{-b}$. Konstanta ima isuviše malu vrednost da bi se predstavila određenim tipom ukoliko je njena vrednost manja od Δ za taj tip. U nekim slučajevima je dovoljno ovakav broj zaokružiti na najbližu vrednost. Međutim, u određenim slučajevima ovakav pristup dovodi do gubitka preciznosti i ispravnosti rezultata. Ovaj problem se često javlja u slučajevima kada je potrebno rezultat neke obrade pomnožiti sa koeficijentom pojačanja. Ukoliko je vrednost tog koeficijenta

manja od Δ , i zaokružena na 0 tada je konačan rezultat obrade uvek 0. Jedan od načina da se ovaj problem reši jeste primena koeficijenta pojačanja po međufazama obrade. Neophodno je podeliti obradu na određene faze, potom razložiti konačni koeficijent tako da rezultat množenja novim koeficijentima po fazama daje rezultat ekvivalentan proizvodu ukupnog koeficijenta i izlazne vrednosti obrade, i zatim izvršiti množenje sa odgovarajućim novim koeficijentom nakon svake faze obrade. [16]

Problem sa većom bitskom razlikom javio se u implementaciji koda u programskom jeziku assembler. Nastao je zbog razlike u aritmetici u odnosu na referentni kod koji je napisan u programskom jeziku C čija je aritmetika sa pokretnim zarezom (*float point*), dok je kod assemblera aritmetika sa fiksnim zarezom (*fixed point*). Na pojedinim mestima u toku obrade bilo je potrebno da se uvede dvostruko precizno računanje. Ukoliko se množe dva registra koji su 32-bitni njihov rezultat se dobija u 72-bitnom akumulatoru. Za ovo je zgodna i L memorija (64-bitna memorija DSP procesora) gde se rezultat množenja može čuvati. U jednom ciklusu moguće je upisati značajnih 64 bita u memoriju. Kada je potrebno pomnožiti akumulator i registar sa dvostrukom preciznosti množi se prvo registar sa donja 32 bita akumulatora potom se taj isti registar množi sa gornjih 32 bita akumulatora. Rezultat prvog množenja se premešta iz gornjih 32 bita akumulatora u donja 32 bita i 8 bita iz zaštitnog (*guardian*) dela registra se premeste u gornjih 32 bita akumulatora. Potom se sabere akumulator sa pomerenim rezultatom i akumulator sa rezultatom drugog množenja registra i akumulatora. Rezultat je dvostruko precizno množenje registra i akumulatora. Radi lakšeg razumevanja deo koda će biti prikazan ispod.

```
y1 = xmem[i2]
a0 = 0; x0 = a1h
b0 = (unsigned)x0 * (unsigned)y1; x0 = a1l
AnyReg(a0h,b0g), (a0l,b0h)
a0+= x0 * (unsigned)y1
```

Funkcionalne celine je potrebno integrisati sa sistemskim softverom digitalnog signal procesora, odnosno, potrebno je sinhronizovati funkcije obrade sa događajima u realnom vremenu (kao što je prijem podataka). Ovaj sistemski softver se često zove i radno okruženje ili radni okvir (*eng. Framework*) digitalnih signal procesora.

4.2.1 Radni okvir procesora CS49844

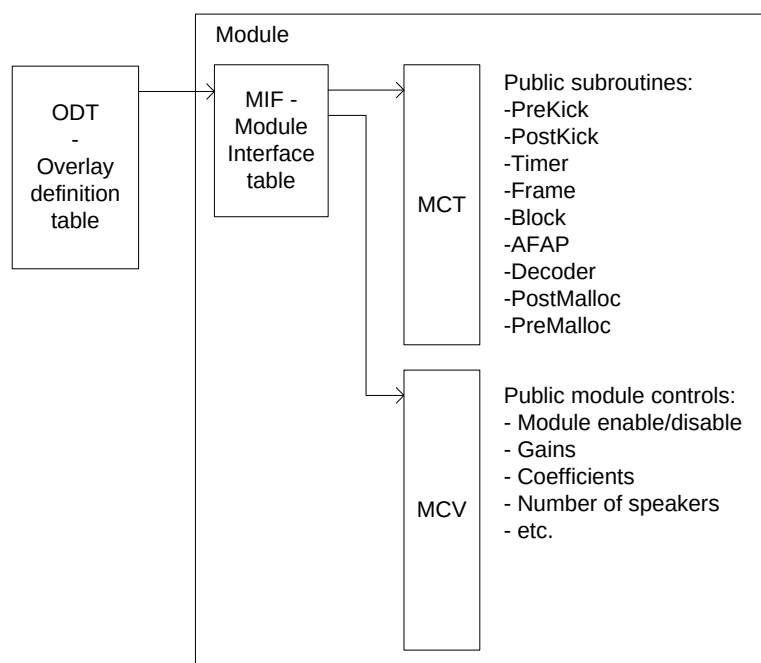
Radni okvir digitalnog signal procesora obezbeđuje raspoređivanje zadataka (funkcija obrade), rukovanje vremenski kritičnim zadacima, upravljanje tokom podataka između različitih

zadataka, komunikaciju između programskih celina unutar samog procesora, kao i komunikaciju sa spoljnim uređajima.

Jezgro radnog okvira se sastoji od jednostavnog operativnog sistema (OS), čija je glavna uloga da bude raspoređivač za određen broj procesnih entiteta (modula). Uslovno rečeno, OS predstavlja monitorsku petlju koja poziva rutine odgovarajućih modula po unapred definisanom redosledu.

Pored operativnog sistema, osnovne komponente radnog okvira čine moduli. Moduli su definisani kao celine sastavljene od rutina i podataka, u skladu sa radnim okruženjem. Svaki modul ima svoj jedinstveni sprežni podsistem - MIF (*eng. Module Interface*) 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 (*eng. Module Call Table*) i MCV tabela (*eng. Module Control Vector*). MCV tabela predstavlja niz javno dostupnih konfiguracionih parametara datog modula, i ona omogućava konfigurisanje modula od strane glavnog kontrolera uređaja (*eng. host*). Struktura ove tabele nema neku unapred definisanu formu i programeru je prepušteno da formira njen sadržaj i strukturu. MCT tabela je niz od devet elemenata – pokazivača na osnovne javne (*eng. public*) rutine. Svaka rutina ima specifičnu ulogu i pozivaju se od strane OS-a kao odgovor na pojavu odgovarajućih događaja u sistemu (*eng. event handlers*). On pristupa modulima preko ODT tabele (*eng. Overlay Definition Table*) koja sadrži pokazivače na MIF tabele svih učitanih modula.

Prva rutina, *Pre-Kickstart* rutina, poziva se nakon prijema inicijalizacione poruke (*eng. reset*). Nakon izvršene inicijalizacije svih modula, OS čeka *kickstart* poruku od kontrolera i poziva se *Post-Kickstart* rutina koja omogućava obradu konfiguracionih podataka prosleđenih modulima od strane kontrolera. Ukoliko je neki od modula zatražio dinamičko zauzimanje memorije ono će se izvršiti u *Pre-Malloc* rutini, a potom je moguće i inicializovati zauzetu memoriju u *Post-Malloc* rutini. *Timer* rutina poziva se periodično kao odgovor na prekid (*eng. interrupt*) generisan od strane brojača realnog vremena. Izvršavanje *Block* i *Frame* rutine je upravljano ulaznim tokom podataka pri čemu se *block* rutina izvršava pri prijemu svakih 16 PCM odbiraka, dok se *frame* rutina izvršava na svakih N blokova. Ovaj broj blokova zavisi od dekoderskog modula u sistemu, odnosno od njegove jedinice dekodovanja. *AFAP* je skraćenica od “*As Fast As Possible*”. Ova rutina se poziva kada god se desi neki događaj u sistemu uz uslov da ne prekida druge rutine istog prioriteta. *AFAP*, *Timer*, *Frame* i *Block* rutine čine takozvanu *Foreground thread* (nit). *Background* rutina se izvršava periodično u pozadinskoj niti (*Background thread*), koja ima niži prioritet od *Foreground* niti i samim tim može biti prekinuta bilo kojom rutinom višeg prioriteta.

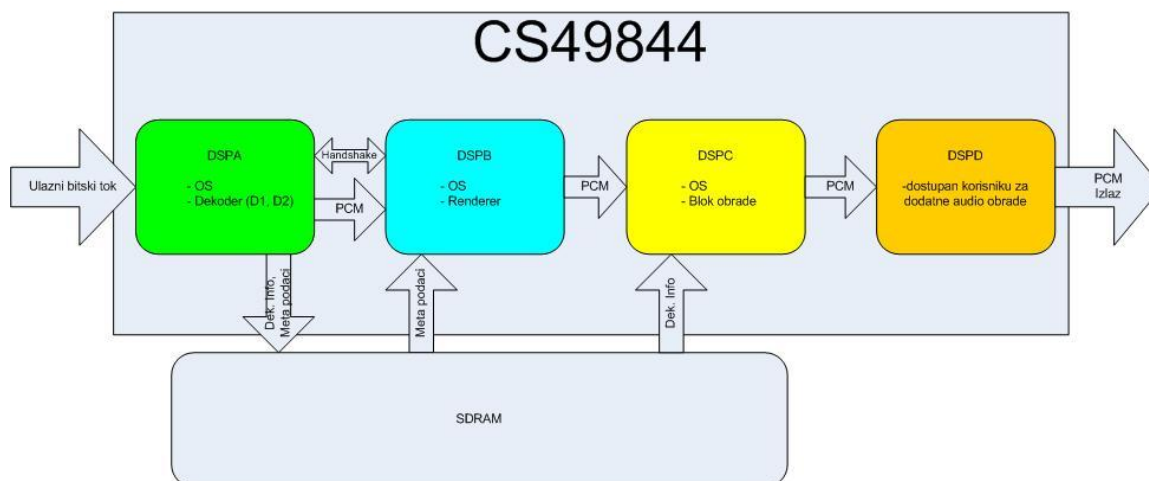


Slika 4.2 Blok dijagram sprege modula sa operativnim sistemom

Pored dela za rukovanje događajima, važan deo radnog okvira 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*). Na Slika 4.2 dat je prikaz sprege modula i operativnog sistema.

4.3 Moduli implementacije

Celokupni implementirani sistem sastoji se iz tri modula: dekodera, renderera i bloka obrade. Sistem je složen te je i pored optimizovanog koda i pažljivog trošenja resursa vrlo zahtevan, kako memorijski tako i u pogledu procesorskog vremena. Iz tog razloga smešten je na tri od četiri jezgra CS48944 procesora. Na Slika 4.3 prikazana je raspodela modula po jezgrima. Dekoder kao prvi nivo obrade smešten je na DSPA jezgru koje prima ulazni komprimovani tok podataka. Na DSPB jezgru smešten je blok renderer – uklanjanje i dodavanje objekata. Na DSPC jezgru izvršavaju se različite obrade signala u cilju prilagođenja audio sadržaja ciljnoj konfiguraciji zvučnika, kao i prilagođenje nivoa signala zahtevima korisnika. Pored datih modula, za svako od jezgara postoji i zaseban operativni sistem.



Slika 4.3 Raspodela modula implementiranog sistema na jezgra ciljne platforme

4.3.1 Dekoder (DSPA jezgro)

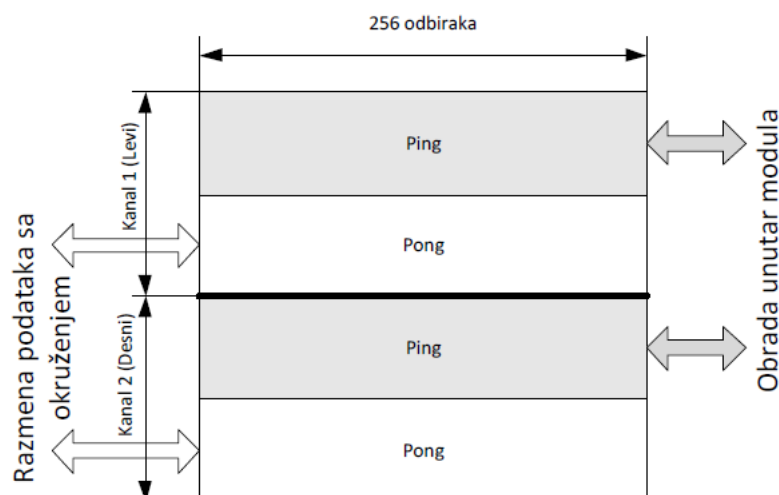
Jezgro DSPA prima ulazni bitski tok koji nosi enkodovani sadržaj sa HDMI ili SPDIF serijske sprege. On se smešta u internu Y memoriju ovog jezgra - FIFO memoriju, veličine 16k reči i dodatnih 24k reči eksterne memorije. Kako ulazni bitski tok neprekidno pristiže, u slučaju da je interna FIFO memorija puna, ulazni podaci se smeštaju u eksternu memoriju, zatim se prebacuju u internu kada je ona oslobođena - za ovo se brine Operativni sistem DSP procesora.

Prvi nivo obrade ulaznog signala jeste dekodovanje koje obavlja dekodera. Implementirana su dva tipa dekodera, D1 i D2. Svaki dekoduje odgovarajući tip ulaznog signala. Dakle zavisno od ulaznog bitskog toka samo jedan od dva dekodera će se učitati na DSPA jezgru. Za smeštanje i prenos PCM podataka odgovoran je sistemski ulazno/izlazni niz (*eng. I/O buffer*) alociran od strane operativnog sistema u Y memoriji, izdijeljen na 32 segmenta, po 128 lokacija, za svaki od podržanih kanala. Svi moduli mogu pristupiti ovom nizu, putem pokazivača na svaki od kanala koje OS redovno ažurira.

Međutim, dekodovani podaci se od strane dekodera prvo smeštaju u interni PCM memorijski niz, odakle se kopiraju u ulazno/izlazni niz. Razlog za ovo je fiksna veličina ulazno-izlaznog niza, koja nije jednaka jedinici obrade dekodera - dekodera u jednom ciklusu obrade izdekoduje više podataka nego što se može smestiti u ulazno-izlazni niz.

Izlazni PCM odbirci iz dekodera predstavljaju ulazne odbirke narednog nivoa obrade, što znači da je nakon završenog ciklusa obrade od strane dekodera, potrebno čekati da se odbirci iz internog PCM niza kopiraju u ulazno/izlazni niz, da bi moglo da se pristupi njihovoj obradi od strane drugih modula i kada svi moduli završe svoju obradu i krajnji izlazni odbirci se nađu u ulazno/izlaznom nizu, dekodera može da otpočne novi ciklus obrade. Sve ovo se dešava na DSP procesoru u realnom vremenu i ovakvo čekanje narušava dinamiku sistema, pa je interni PCM memorijski niz implementiran sa posebnom strukturom. Veličine je dvostruko veće od jedinice

obrade dekodera, po kanalu. Ovakva struktura naziva se *Ping-Pong* memorijski niz, prikazan na Slika 4.4. Svaki kanal sastoji se iz dva dela - ping i pong. Dok dekodek dekoduje i upisuje sadržaj u Ping deo kanala, Pong deo kanala se prosleđuje na ulazno-izlazni memorijski niz na dalju obradu drugim modulima. Kada dekodek završi i popuni Ping delove kanala, Ping i Pong menjaju uloge i sada se dekodovani sadržaj upisuje u Pong deo, dok se upisani PCM podaci iz Ping dela upisuju u ulazno/izlazni memorijski niz i na taj način prosleđuju drugim modulima.



Slika 4.4 Ping-Pong struktura memorijskog niza

U Tabela 4.2 dat je pregled utrošenih resursa ovog jezgra. Ovaj modul zauzima oko 80% resursa ovog jezgra.

Utrošak procesorskog vremena	240 MIPS
Utrošak memorije za podatke (X memorija)	40448 reči
Utrošak memorije za podatke (Y memorija)	35626 reči
Utrošak programske memorije	25797 reči

Tabela 4.2 Utrošak resursa DSPA jezgra

Osim generisanja PCM odbiraka, dekodek raspakuje meta podatke iz ulaznih komprimovanih podataka i prosleđuje ih odgovarajućim modulima. Ovo slanje se obavlja preko spoljne SDRAM memorije i biće detaljnije objašnjeno u narednom poglavlju.

4.3.2 Modul za renderovanje objekata u prostoru (DSPB jezgro)

Modul za renderovanje objekata zbog svoje kompleksnosti možemo podeliti u 2 dela. Inicijalizaciju i Renderovanje. Svaki od njih će biti detaljno objašnjen u nastavku.

4.3.2.1 Inicijalizacija

Da bi se algoritam za renderovanje izvršavao ispravno potrebno je izvršiti inicijalizaciju algoritma. Pod inicijalizacijom algoritma podrazumeva se računanje određenih tabela. Nije moguće izvršavati ispravno renderovanje ukoliko inicijalizacija nije završena. Tek nakon uspešne inicijalizacije modula za renderovanje objekata modul obaveštava ostatak sistema (dekoderski modul prevashodno) o njegovoj spremnosti za početak rada. Iz ovih razloga, dekodler neće početi sa slanjem podataka ka modulu za renderovanje sve dok ne primi poruku o uspešno završenoj inicijalizaciji. Inicijalizacija se vrši u *Background* rutini i zavisi od izlazne kanalne maske. Izlazna maska je 32-bitna i predstavlja zvučnu konfiguraciju sistema odnosno prisutne zvučnike u sistemu. Svaki bit u maski predstavlja jedan zvučnik. Na osnovu prisutnih zvučnika u kanalnoj maski kreiraju se tabele koje su potrebne za ispravno renderovanje objekata. Jednom proračunate tabele nije potrebno ponovo računati tako da se inicijalizacija samo jednom izvršava.

Dekoder preko OS-a upisuje u odgovarajuće MCV polje ovog modula izlaznu zvučnu masku u koji je potrebno izvršiti renderovanje objekata. Na osnovu te maske se vrši inicijalizacija modula. Poziva se funkcija za prebrojavanje broja kanala enkodovanih u ulaznom test vektoru ovo se vrši iz same maske. Potom se u petlji po kanalima proverava da li je kanal punog spektra (eng, *Fullrange*) ili je u pitanju niskofrekventni kanal (*low-frequency effects LFE*). Kad naiđe na LFE preskače na sledeći kanal. Ukoliko je kanal punog spektra poziva se funkcija za iščitavanje sfernih koordinata tog kanala. Potom se vrši provera da li se kanal nalazi u gornjoj hemisferi koja je prikazana na Slika 3.2. Ukoliko je ugao zenita veći od 30 stepeni taj kanal se nalazi u gornjoj hemisferi. Uvećava se brojač kanala gornje hemisfere. U slučaju kada je kanal manji od 30 stepeni ulaz u funkciju za proveru donje hemisfere. Da bi kanal bio u donjoj hemisferi potrebno je da je vrednost njegovog zenita manja od -30 stepeni. Ukoliko je taj uslov ispunjen kanal se stvarno nalazi u donjoj hemisferi i brojač kanala donje hemisfere se uvećava za jedan. Tek ako se kanal ne nalazi ni u gornjoj ni u donjoj hemisferi odnosno da je ugao njegovog zenita između -30 i 30 stepeni kažemo da je on na ekvatoru. Ovde se pored povećanja brojača kanala na ekvatoru povećava i broj tačkastih izvora, konvertuju se njihove koordinate iz sfernog koordinatnog sistema u kartezijanski. Na osnovu izlazne konfiguracije zvučnika vrši se računanje da li je potrebno uvoditi neki od virtualnih izvora. Za virtualne zvučnike se takođe

određuje kojoj hemisferi pripadaju i računaju se njihove koordinate. Vršiti se sortiranje koordinata u odnosu na njihov azimut tako da se kreće od najmanjeg do najvećeg ugla.

Računanje tripleta se vrši u tri ugnježdene petlje (i) gde prva kreće od nule do broja tačkastih izvora, početak druge petlje (j) koja se nalazi unutar prve petlje kreće od 1 isto do broja tačkastih izvora treća petlja (k) počinje od broja 2 do broja tačkastih izvora. Kreira se matrica koju čine koordinate kanala odnosno tačkastih izvori. Za prvi vektor matrice uzimaju se koordinate zvučnika sa najmanjim azimutom, drugi vektor je odmah sledeći tako i treći. Radi lakšeg razumevanja deo koda je prikazan u polju ispod.

```
uint32_t i;
for( i = 0; i < point_count; i++ )
{
    uint32_t j;
    for( j = i + 1; j < point_count; j++ )
    {
        uint32_t k;
        for( k = j + 1; k < point_count; k++ )
        {
            base_matrix3_f32_t m; /* spanning matrix */
            float determinant;    /* determinant */
            bool_t is_invertible; /* is invertible flag */

            m.v[0] = p_point_array[i];
            m.v[1] = p_point_array[j];
            m.v[2] = p_point_array[k];
        }
    }
}
```

Računa se determinanta matrice i proverava se da li postoji inverzna matrica. Ukoliko inverzna matrica ne postoji prelazi se na sledeći vektor i novu matricu. Ako inverzna matrica postoji izračunava se. U novoj petlji uzima se redom jedan po jedan vektor iz *point_array* niza i množi se prethodno izračunatom inverznom matricom. X, Y i Z vrednosti dobijenog vektora se sabiraju. Ukoliko njihov zbir nije veći od 1 uzima se sledeći vektor iz niza. Ako jeste broj tripleta se povećava i čuva se inverzna matrica kao i m matrica u za to predviđene nizove.

Kada je računanje tripleta završeno i njihov broj je poznat može se zvati *hull_pan* funkcija. Ova funkcija se poziva dva puta u okviru inicijalizacije hull-a. Prvi put se računaju pojačanja u prednjoj, a drugi put u zadnjoj hemisferi.

```
hull_f32_t_pan ( hull_f32_t * p_hull, bool_t * p_out_panned, float *
p_out_point_gain_array, float in_azimuth_degrees, float in_elevation_degrees,
float in_width_degrees, float in_height_degrees, float in_rotation_degrees )
```

Na samom ulasku u funkciju proveravaju se granični slučajevi. Da li se *width* nalazi između 179 i 180 stepeni ukoliko da dobija vrednost 179 stepeni, zatim da li se nalazi između 180 i 181 stepen ako da njegova vrednost se postavlja na 181 stepen. Isto ovo važi i za *height* ugao.

Proveravaju se i kombinacije ukoliko je *width* veći od 180 stepeni a *height* se nalazi između 0 i 180 stepeni u tom slučaju menja se vrednost *width* na 179 stepeni. Isto važi i ako je *height* veći od 180 stepeni a *width* se nalazi između 0 i 180 stepeni onda se vrednost *height* menja na 179 stepeni. Na osnovu ovih parametara imamo sledećih 6 slučajeva:

- 1) $width = 0 \ \& \ height = 0$
- 2) $(width > 0 \ \& \ height = 0) \ || \ (width = 0 \ \& \ height > 0)$
- 3) $width < 180 \ \& \ height < 180$
- 4) $(width > 180 \ \& \ height < 180) \ || \ (width < 180 \ \& \ height > 180)$
- 5) $width < 360 \ \& \ height < 360$
- 6) sve ostale kombinacije

Ovde će biti objašnjen samo prvi slučaj jer se on pretežno najviše koristi u samom renderovanju.

Na samom početku se vrši konverzija iz sfernog u kartezijev koordinatni sistem koristeći sledeće formule $x = r \sin\theta \cos\phi$, $y = r \sin\theta \sin\phi$, $z = r \cos\theta$. Potom se ulazi u petlju koja se izvršava po broju tripleta. Iščitavaju se matrice sačuvane prilikom računanja broja tripleta i množe sa prethodno izračunatim vektorom. Proveravaju se X, Y i Z koordinate vektora da li su veće od dozvoljene aritmetičke greške u daljem tekstu epsilon (-0,000001). Ukoliko su sve tri koordinate vektora veće od epsilon njihove vrednosti se smeštaju u privremen niz pojačanja koji je inicijalno postavljen na nulu. Smeštanje se vrši iščitavanjem određenog elementa iz niza i dodavanjem X, Y, Z vrednosti na njega kao i potom vraćanjem novonastale vrednosti u privremeni niz pojačanja. Ovom prilikom se povećava i brojač renderovanih tripleta. Po završetku petlje koja se izvršava po broju tripleta proverava se da li je broj renderovanih tripleta veći od 1 i ukoliko jeste svaki elemenat privremenog niza pojačanja se deli sa brojem renderovanih tripleta i tako čuva u izlaznom nizu. Po izlasku iz *hull_pan* funkcije radi se normalizacija pojačanja.

Proverava se da li je potrebno praviti virtualne zvučnike u gornjoj i donjoj hemisferi. Ukoliko jeste potrebno je proračunati pojačanje koje odgovara njihovim tačnim pozicijama. To se radi na sledeći način. Prva petlja ide po svim tačkama na ekvatoru unutar koje se za svaki potrebnii virtualni izvor iščitava azimut. On se konvertuje u opseg od -180° do 180° . *Elevation* se postavlja na 0 i pomoću *hull_pan* funkcije dobija se odgovarajuće pojačanje. Nešto drugačije se postavljaju i proračunavaju virtualni izvori koji se nalaze na ekvatoru. Pronađe se minimalni i maksimalni azimut fizičkih zvučnika koji postoje u sistemu. Na osnovu sledeće formule vrši se remapiranje azimuta virtualnog zvučnika na fizičke zvučnike.

$$diff = max_azimuth - min_azimuth$$

$$Remap_azimuth = max_asimut - \frac{virtual_azimuth - max_azimuth}{360^\circ - diff} * diff$$

Dobijeni `remap_azimuth` se prosleđuje kao parametar funkciji `hull_pan` koja proračunava potrebna pojačanja.

Inicijalizacija traje približno 180 000 000 instrukcija, odnosno oko 0,6 sekundi. Ovim je završena inicijalizacija i sve potrebne tabele su popunjene i spremne za korišćenje tokom renderovanja.

4.3.2.2 Renderovanje

Završetkom inicijalizacije dekodera počinje da šalje meta podatke modulu za renderovanje objekata u prostoru. Proverava se da li je došlo do promene meta podataka i da li je ovo prvi put da ih je dekodera poslao u toku ulaznog test vektora. Ukoliko jeste potrebno je isparsirati ih i postaviti u određene strukture. Informacije koje nose meta podaci su izlazna maska po kojoj treba izvršiti renderovanje, broj objekata koji se nalaze u ulaznom test vektoru kao i njihove koordinate u prostoru. Meta podaci se šalju na početku svakog bloka jedinica obrade u daljem tekstu frejma. Broj odbiraka u jednom frejmu modula za renderovanje objekata u prostoru je 256. Podržane frekvencije su 1fs: 32kHz, 44,1kHz, 48 kHz i 2fs: 64kHz, 88,2kHz i 96kHz.

Nakon postavljanja validnih meta podataka u za to predviđene strukture i tabele nastavlja se sa proračunom pojačanja. Postavljanje objekta na tačno određenu lokaciju u prostoru dobija se množenjem pojačanja i PCM odbiraka objekta. Na samom početku funkcije vrši se prebrojavanje kanala na osnovu izlazne maske, kao i prebrojavanje kanala punog spektra. Petlja u kojoj se računa pojačanje izvršava se po objektima. Iz meta podataka poslanih od strane dekodera iščitavaju se koordinate u sfernom koordinatnom sistemu za trenutni objekat. Ukoliko izlazna maska sadrži samo jedan kanal odnosno u pitanju je mono izlaz onda se ne proračunava pojačanje uz pomoć VBAP nego se njegova vrednost postavlja na 1. Za ostale izlazne maske poziva se `vector_base_panner_pan` funkcija. Kao parametar ovoj funkciji prosleđuje se azimut i *elevation* od trenutnog objekta. Na samom početku se poziva `hull_pan` funkcija sa azimutom i *elevation*. Ona je detaljnije objašnjena u poglavlju 4.2.1. Nakon `hull_pan` funkcije potrebno je proći kroz virtualne kanale i rasporediti ih na određene fizičke kanale. Prolazi se kroz gornju, donju hemisferu i ekvator. Kod gornje hemisfere petlja se izvršava po svim tačkama na ekvatoru, iščitava se pojačanje izračunato u `hull_pan` funkciji za virtualni kanal gornje hemisfere i ukoliko je njegova vrednost nula prelazi se na sledeću tačku. U slučajevima kada je vrednost pojačanja

virtualnog kanala iz gornje hemisfere različita od nule radi se sledeće. Iščitava se vrednost pojačanja trenutne tačke na ekvatoru i pomoću sledeće formule računa se novo pojačanje.

$$hull_equator_gain_new = \sqrt{(hull_equator_gain_previus^2 + hull_upper_ring_gain^2)}$$

Novo tek izračunato pojačanje upisuje se na istu poziciju sa koje je iščitano prethodno pojačanje za tu tačku ekvatora. Po završetku petlje sa svim tačkama na ekvatoru prelazi se na virtualne tačke u donjoj hemisferi. Postupak izračunavanja pojačanja za ove tačke je sličan kao i za virtualne tačke u gornjoj hemisferi osim što se umesto pojačanja gornje hemisfere iščitava pojačanje donje hemisfere i formula izgleda ovako.

$$hull_equator_gain_new = \sqrt{(hull_equator_gain_previus^2 + hull_lower_ring_gain^2)}$$

Kod virtualnih kanala koji se nalaze na ekvatoru proces je identičan kao i za gornju i donju hemisferu samo što se uzimaju pojačanja koja odgovaraju virtualnim kanalima na ekvatoru.

$$hull_equatorPhysical_gain_new = \sqrt{(hull_equatorPhysical_gain_previus^2 + hull_lowerVirtual_ring_gain^2)}$$

Nakon izračunavanja pojačanja prelazi se na njihovu primenu. Prva petlja ide po broju objekata, a druga petlja koja se nalazi unutar nje po broju prisutnih kanala u izlaznoj konfiguraciji. Treća petlja ide po broju odbiraka u frejmu. Unutar nje se množi pojačanje za taj objekat sa PCM odbircima objekta. Na ovaj način se dobijaju objekti raspoređeni na kanale na izlazu.

U Tabela 4.3 dat je pregled utrošenih resursa ciljne platforme bloka za renderovanje. Ovi rezultati zadovoljavaju ograničenja ciljne platforme. Utrošak procesorskog vremena je oko 40 miliona instrukcija u sekundi tj. MIPS-a (*Milion instructions per second*) prilikom renderovanja objekata za njihovo uklanjanje kao i prilikom renderovanja objekata za njihovo uključivanje u audio izlaz, od čega je 25 MIPS-a utrošeno u toku preračunavanja faktora pojačanja kojim će se primeniti objekat na pojedine izlazne zvučnike, ili oduzeti iz osnovnog zvučnog zapisa, dok je 15 MIPS-a potrebno za samu primenu. Kada su kanalni objekti u pitanju potrošnja procesorskog vremena je oko 15 MIPS-a. Ovo je očekivano obzirom da se za ovu vrstu renderovanja ne vrše nikakva preračunavanja faktora pojačanja već se oni dobijaju direktno iz meta podataka.

Data potrošnja u tabeli odnosi se za frekvencije od 1Fs (32 kHz, 44,1 kHz i 48 kHz), dok je u slučaju dvostruko veće frekvencije i utrošak procesorskog vremena veći. Predračuni faktora pojačanja su nezavisni od frekvencije, pa je utrošak MIPS-a isti, dok se potrebni MIPS-i za samo uklanjanje i dodavanje objekata udvostručuje. Ovo je takođe očekivano ponašanje obzirom da je veličina ulaznih PCM odbiraka duplo veća u slučaju 2Fs (64 kHz, 88,2 kHz i 96 kHz) frekvencija. Sumirano, DSPB troši oko 155 MIPS-a na frekvencijama od 88,2kHz, 64kHz i 96kHz.

Utrošak procesorskog vremena	125 MIPS
Utrošak memorije za podatke (X memorija)	47975 reči
Utrošak memorije za podatke (Y memorija)	41612 reči
Utrošak programske memorije	9375 reči

Tabela 4.3 Utrošak resursa DSPB jezgra

4.3.3 PostProcessing (DSPC jezgro)

Na DSPC jezgru nalazi se obrada signala kao što je spuštanje ili povećanje izlaznog broja kanala u skladu sa ciljnom konfiguracijom zvučnika, prilagođavanje zvučnog sadržaja različitom prostornom rasporedu zvučnika, normalizovanje nivoa signala. Ovaj modul takođe zahteva određene informacije od strane dekodera kao što su izlazna konfiguracija zvučnika, frekvencija signala, zatim opseg izlaznog signala, i slično. Prenos ovih informacija obavlja se na isti način kao i prenos podataka između DSPA i DSPB jezgra, preko SDRAM memorije. Utrošeni resursi prikazani su u Tabela 4.4.

Utrošak procesorskog vremena	250 MIPS
Utrošak memorije za podatke (X memorija)	51652 reči
Utrošak memorije za podatke (Y memorija)	52609 reči
Utrošak programske memorije	19365 reči

Tabela 4.4 Utrošak resursa DSPC jezgra

4.4 Protokol za prenos podataka između jezgara

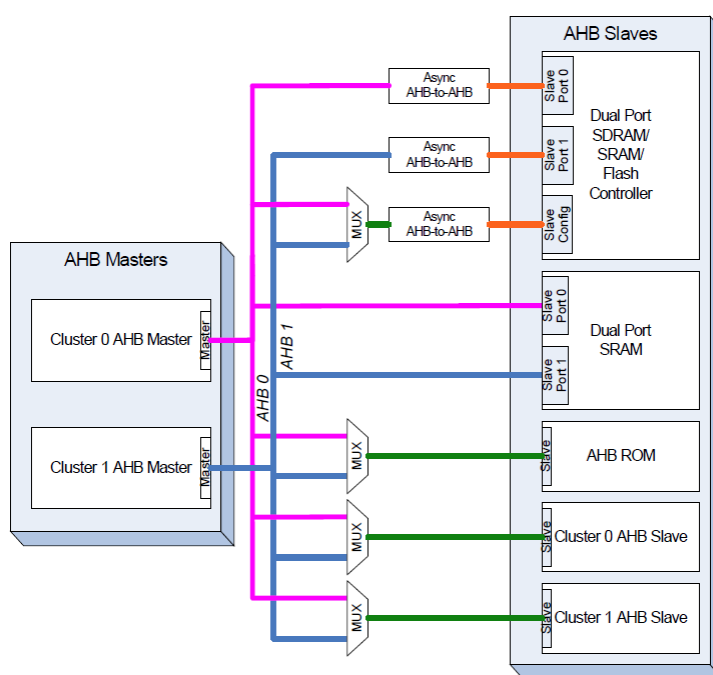
Opisani meta podaci neophodni za obradu u sklopu bloka renderer prenose se od strane dekodera raspakovani iz kompresovanog ulaznog bitskog toka. Ovo uvećava količinu podataka koji se prenose u odnosu na slanje komprimovanog sadržaja, međutim otpakivanje enkodovanih podataka na strani 3D renderera bi dodatno usložilo modul koji je već dosta procesorski opterećen mnogobrojnim matematičkim proračunima. Takođe, pojedine informacije potrebne su

i samom dekozeru, pa je implementacija algoritamski jednostavnija u slučaju da se kontinualan ulazni komprimovani sadržaj celokupan raspakuje na strani dekozera, odnosno jezgra koji i prima ove podatke u svoju internu memoriju.

Podaci između jezgara prenose se na početku svakog ciklusa obrade dekozera koji je dužine od 512, 1024 ili 2048 odbiraka, u zavisnosti od prirode ulaznog signala. Ovaj ciklus obrade dekozera naziva se *Frame*. Meta podaci su sinhronizovani sa audio sadržajem preko jedinstvenog broja Frame-a čije podatke nosi.

Ovako raspakovani podaci zauzimaju 5 reči meta podataka zajedničkih za sve prisutne objekte, zatim 522 reči meta podataka za svaki od postojećih objekata, kao i 17 reči koje nose informacije o ciljnoj konfiguraciji zvučnika, parametara zadatih od strane korisnika i sl. Najveći prenos obavlja se u slučaju maksimalnog broja objekata, što je 14 objekata, pa je tako maksimalan broj reči za prenos jednak 7330 reči.

Na Sliku 4.5 prikazan je AHB magistrala i Master/slave entiteti koji koriste tu magistralu za prenos podataka.



Slika 4.5 AHB magistrala

Najjednostavnija razmena podataka između jezgara DSP-ja jeste preko deljene SRAM memorije, koju vide sva jezgra i veličine je 8k. Kako je ova vrednost približna vrednosti veličine podataka za slanje, rizično je da se podaci šalju na ovaj način. Iz tog razloga podaci se šalju preko SDRAM memorije veličine 64k reči. Sva komunikacija obavlja se preko DMA AHB protokola. Postoje dve AHB magistrale, AHB0 i AHB1, gde jednoj pristupa DSPA i DSPB

jezgro - *Cluster 0*, a drugoj DSPC i DSPD – *Cluster 1*. DSPA kao nadređeni - *master* pristupa AHB0 magistrali nakon što dobije potvrdu od SDRAM-a da može da smesti podatke. DSPB potom pristupa istoj magistrali i čita podatke od podređenog - *slave* uređaja, SDRAM-a. SDRAM radi na taktu od 133 MHz sa bitskim protokom od 64 Mb u jednom transferu.

5. Ispitivanje i rezultati

Testiranje i verifikacija sistema predstavljaju krajnji korak u implementaciji. Svrha ovog procesa je da se pokaže ispravnost implementacije algoritma na ciljnoj platformi, tako što se utvrdi da se generisani izlazi iz implementiranog sistema slažu sa izlazima generisanim izvršavanjem referentnog koda - referentnim izlazima. U toku ovog procesa osim detektovanja grešaka u softveru proverava se i kvalitet softvera.

U okviru ovog rada urađeno je funkcionalno ispitivanje metodom crne kutije – BBT [17] (*eng. Black Box Testing*). BBT alat je razvijen na Odseku za računarsku tehniku i računarske komunikacije Fakulteta tehničkih nauka u Novom Sadu i predstavlja alat za ispitivanje i verifikaciju. Generalno rečeno, BBT alat je automatizovana i daleko složenija verzija batch skripti. Prednosti BBT alata nad batch skriptama su brojne, od relativno jednostavne sintakse za pisanje ispitnih slučajeva, samim tim i bržeg razvoja istih, do lakog korišćenja, prenosivosti, mnoštva ugrađenih alata koji pojednostavljaju proces ispitivanja, sve do preglednih izveštaja o rezultatima. Ovaj alat dakle omogućava brz razvoj ispitnih slučajeva, a ugrađeni alati omogućavaju izvršavanje širokog spektra operacija. Jednom razvijeni ispitni slučajevi mogu se ponoviti neograničeni broj puta, kao i na proizvoljnom računaru (što je ponekad problem kod običnih batch skripti).

Od proizvođača tehnologije koja je bila predmet ispitivanja dobijena je dokumentacija koja predstavlja ispitnu proceduru. Ova dokumentacija sadrži specifikacije testnih slučajeva, ulazne podatke i ostale dokumente potrebne za verifikaciju implementacije testirane audio tehnologije. Ova dokumentacija je dostupna samo softverskim kompanijama koje žele da verifikuju svoju implementaciju i da dobiju sertifikat od kompanije koja je licencirala tu tehnologiju.

Početni korak u fazi razvoja testova podrazumeva organizaciju direktorijuma radnog okruženja. Pošto i testni okvir i BBT alat imaju određene specifičnosti i unose određene zahteve, radno okruženje mora biti organizovano tako da odgovori svim zahtevima.

Uz svaki testni slučaj date su i tekstualne datoteke u kojima se nalaze sve informacije po kojima je određen testni slučaj karakterističan. U ovim datotekama navedeni su ulazni testni vektori, vrednosti konfiguracionih parametara i drugo. Kako bi se sve ove informacije, za sve testne slučajeve, nalazile sumirane na jednom mestu, napisana je Python skripta koja parsira prethodno pomenute datoteke i kreira Excel tabelu. U ovoj tabeli, u prvoj koloni izlistani su svi testni slučajevi, dok su u narednim kolonama upisane vrednosti svih parametara neophodnih za odgovarajući testni slučaj. Dalje se ova Excel tabela prosleđuje skripti u kojoj je implementirana logika pomoću koje se generišu BBT testovi. Uzima se jedna po jedna linija, odnosno željeni testni slučaj sa svim njegovim parametrima i generišu se testni slučajevi za BBT alat (.tst datoteke) čiji je deo prikazan na Slika 5.1, kao i konfiguracioni fajlovi (.cfg datoteke).

```

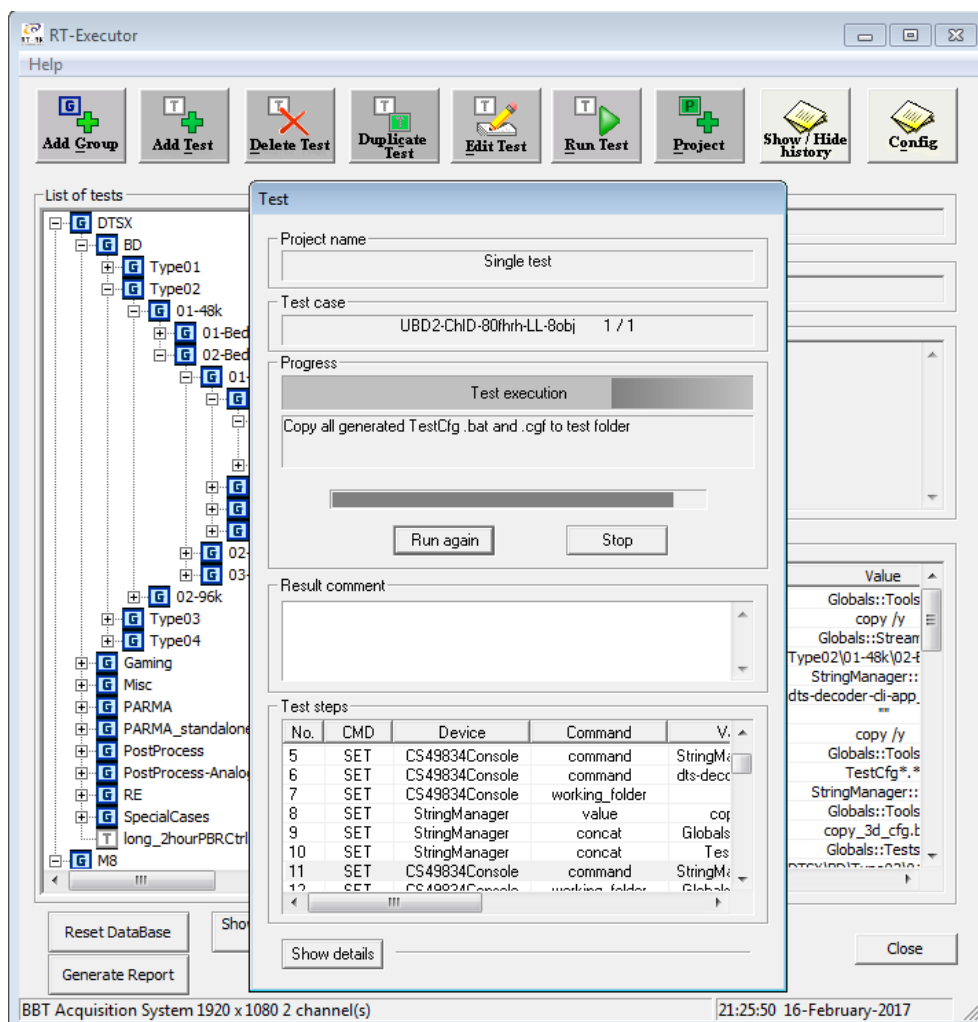
1  [test]
2  ID = 121
3  name = UBD2-ChID-80fhrh-LL-8obj
4  description = UBD2-ChID-80fhrh-LL-8obj test file
5  log_file = UBD2-ChID-80fhrh-LL-8obj.log
6
7  //////////////////////////////////////
8  //
9  //   Copy input stream to tools folder
10 //
11 //////////////////////////////////////
12
13 [step]
14 description = go to the Tools folder
15 device = CS49834Console
16 command = working_folder
17 value = Globals::ToolsDir
18 delay = 0
19 option = [SET]
20
21 [step]
22 description = Copy DTSUHD input streams to Tools folder
23 device = StringManager
24 command = value
25 value = copy /y
26 delay = 0
27 option = [SET]
28
29 [step]
30 description = Setup path to the input stream
31 device = StringManager
32 command = concat
33 value = Globals::StreamsBD
34 delay = 0
35 option = [SET]
36
37 [step]
38 description =
39 device = StringManager
40 command = concat
41 value = Type02\01-48k\02-BednObjects\01-ChID\01-8x\UBD2-ChID-4824LL-80fhrh-LL-8obj.dtsd
42 delay = 0
43 option = [SET]

```

Slika 5.1 Primer generisanog testnog slučaja za BBT alat (deo .tst datoteke)

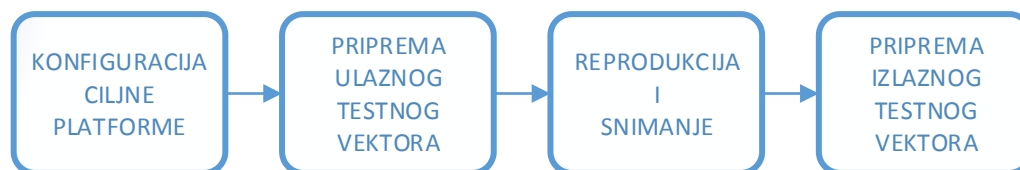
Ukupno je korišćen 3621 test, od kojih je sa 1418 testova testiran dekodir D1, dok se sa 2203 testa verifikovala funkcionalnost sistema sa dekodir D2. Testovi su podeljeni u 8 grupa, gde svaka grupa odgovara određenoj funkcionalnosti sistema.

Nakon generisanja testova, pokretanjem BBT aplikacije, Slika 5.2, počinje i samo izvršavanje testova.



Slika 5.2 BBT aplikacija nakon pokretanja

Na sledećoj slici je prikazan blok dijagram koji ilustruje osnovne korake jednog BBT testnog slučaja, koji redom prate korake date u .tst datotekama.



Slika 5.3 Blok dijagram izvršavanja jednog BBT testa

Prvi korak tiče se konfiguracije ciljne platforme, odnosno prebacivanja svih izvršnih datoteka koje su potrebne (*firmware* svih komponenti koje se paralelno izvršavaju) na DSP

procesoru. U ovom koraku se takođe vrši i pozivanje konfiguracionih datoteka, odnosno komandi koje su u njima navedene, a pomoću kojih se uključuju odnosno isključuju pojedine funkcionalnosti, šalju vrednosti pojedinih parametara, kao što su razna pojačanja, vrednosti graničnih frekvencija i tako dalje.

Na Slika 5.4 dat je primer testnog okruženja. Kao ciljni procesor koristi se već pomenuti CS48944, četvororejzarni digitalni signal procesor iz Crystal DSP familije procesora firme *Cirrus Logic*, a za reprodukciju i snimanje signala korišćena je zvučna kartica Echo Audio 0. Na levoj strani slike je razvojna ploča CDB49X, u čijem centralnom delu je pozicioniran digitalni signal procesor, a sa desne strane se nalazi Echo Audio zvučna kartica.



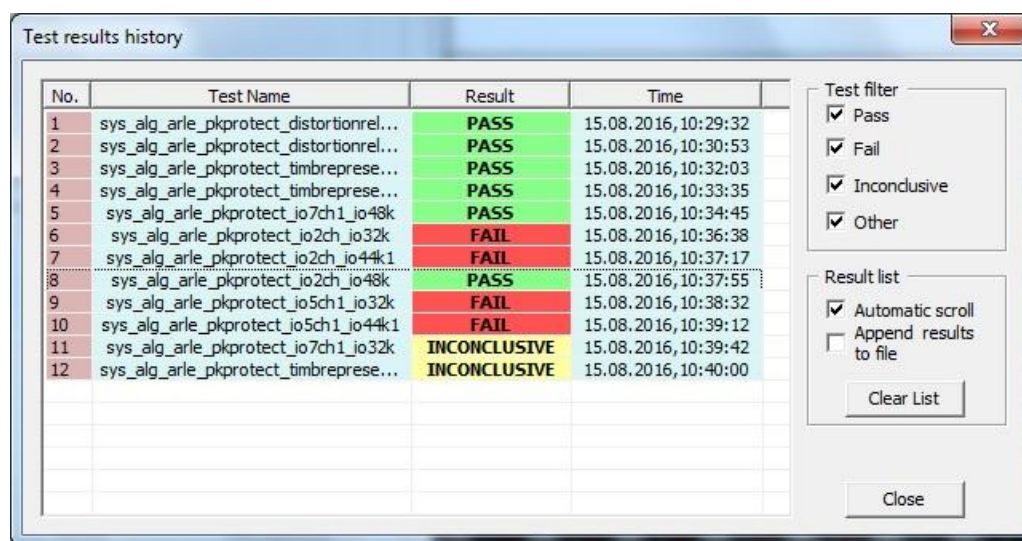
Slika 5.4 Ispitno okruženje

Sledeći korak podrazumeva sve aktivnosti vezane za pripremu ulaznog testnog vektora. U nekim slučajevima, ulazni testni vektor se nalazi u formatu koji se ne može direktno pustiti već se posebno priprema uz pomoć adekvatnog alata da bi se pustio preko Echo Audio zvučne kartice i bio dekodovan. Ponekad se dešava da je neophodno povećati frekvenciju odabiranja kod ulaznog testnog vektora.

U narednom koraku vrše se podešavanja Echo Audio zvučne kartice preko koje se obavlja reprodukcija i snimanje. Zvučna kartica koja je korišćena u ovom radu ograničena je da može reprodukovati testne vektore koji sadrže maksimalno osam kanala. Ukoliko ulazni testni vektor sadrži više od osam kanala, reprodukcija i snimanje se mora izvoditi u nekoliko iteracija.

Poslednji korak odnosi se na aktivnosti nakon snimanja izlaznog testnog vektora. Najčešće su to vremensko poravnanje snimljenog izlaza koje se vrši umetanjem sinhronizacione reči, kombinovanje testnih vektora snimljenih kroz nekoliko iteracija u jedan izlazni testni vektor i slično.

Mogući ishodi izvršavanja BBT testova su *PASS*, *FAIL* i *INCONCLUSIVE*. Ukoliko se svi koraci predviđeni testom uspešno izvrše, ishod testa biće *PASS*, sa druge strane, ukoliko se desi da prilikom izvršavanja nekog od koraka predviđenih testom dođe do greške ishod testa biće *FAIL*. Ukoliko se, pak, desi da ishod izvršavanja testa bude *INCONCLUSIVE* to znači da je izvršavanje testa prekinuto pre nego što su izvršeni svi predviđeni koraci. Primer prikaza ishoda izvršavanja testova generisanog od strane BBT alata dat je na Slika 5.5.



No.	Test Name	Result	Time
1	sys_alg_arle_pkprotect_distortionrel...	PASS	15.08.2016,10:29:32
2	sys_alg_arle_pkprotect_distortionrel...	PASS	15.08.2016,10:30:53
3	sys_alg_arle_pkprotect_timbrepres...	PASS	15.08.2016,10:32:03
4	sys_alg_arle_pkprotect_timbrepres...	PASS	15.08.2016,10:33:35
5	sys_alg_arle_pkprotect_io7ch1_io48k	PASS	15.08.2016,10:34:45
6	sys_alg_arle_pkprotect_io2ch_io32k	FAIL	15.08.2016,10:36:38
7	sys_alg_arle_pkprotect_io2ch_io44k1	FAIL	15.08.2016,10:37:17
8	sys_alg_arle_pkprotect_io2ch_io48k	PASS	15.08.2016,10:37:55
9	sys_alg_arle_pkprotect_io5ch1_io32k	FAIL	15.08.2016,10:38:32
10	sys_alg_arle_pkprotect_io5ch1_io44k1	FAIL	15.08.2016,10:39:12
11	sys_alg_arle_pkprotect_io7ch1_io32k	INCONCLUSIVE	15.08.2016,10:39:42
12	sys_alg_arle_pkprotect_timbrepres...	INCONCLUSIVE	15.08.2016,10:40:00

Slika 5.5 Prikaz ishoda izvršavanja testova

Da bi se omogućilo potpuno automatizovano ispitivanje prihvatljivosti implementiranog rešenja, proizvođač tehnologije je obezbedio i posebnu softversku komponentu, odnosno testni okvir, koja omogućuje spregu pomoću komandne linije (eng. Command-Line Interface - CLI) za izvršavanje ispitivanja. Osim za ispitivanje prihvatljivosti, testni okvir se koristi i za generisanje referentnih signala, koji će biti upoređivani sa snimljenim signalima. Testni okvir je potpuno automatizovan i ne zahteva dalju interakciju sa korisnikom nakon pokretanja.

Postoji više načina ispitivanja rezultata:

1. Slušni testovi – preslušavanje ispitnih testova, provera čujnih distorzija 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.

Kriterijum za poređenje pojedinačnih testnih slučajeva dat je u pomenutom testnom okviru. Spektralni testovi čine oko 77% ukupnog broja testova, testova identičnosti u bit ima oko

17%, dok ostalih, 6% testova, čine testovi koji se porede bitski u vremenskom domenu sa dozvoljenim određenim odstupanjima. Ovim testovima testiraju se pojedini delovi signala: tranzicije, amplitudski prelazi i sl.

Ispitivanje rezultata završava se pokretanjem glavne skripte koja generiše izveštaj na unapred definisanoj lokaciji, u obliku HTML datoteke. Izveštaj ima formu tabele, a najbitnija polja su: jedinstveni identifikator testnog slučaja, metoda evaluacije koja je upotrebljena u konkretnom testnom slučaju i status - *PASSED*, *FAILED*, *UNRESOLVED*. Ukoliko je status testa *FAILED* ili *UNRESOLVED*, potrebno je preduzeti određene koraka, i ispitati razlog zbog koga test ne prolazi.

Dobijeni rezultati nakon izvršene opisane procedure testiranja zadovoljili su unapred definisane kriterijume za verifikaciju, status svakog od 3621 testnog slučaja je *PASSED*, i na taj način pokazano je da sistem odgovara zahtevima.

6. Zaključak

U okviru ovog rada implementacija renderer algoritma je realizovana na višejezgarnoj platformi sa ograničenim resursima, CS49844 DSP procesorom firme *Cirrus Logic*. Polazna osnova bio je referentni kod napisan u višem programskom jeziku C. Prilagođenjem algoritma tako da zadovolji ograničenja nametnuta izborom ciljne platforme, kod je napisan u programskom jeziku assembler, integrisan u radno okruženje procesora i optimizovan.

Realizovani sistem sastoji se iz tri modula: dekođer, renderer i blok obrade. Zbog memorijskih zahteva, kao i zahteva u pogledu procesorskog vremena, sistem je izdđljen po jezgrima upravo po pomenutim funkcionalnim celinama. Dekoder troši oko 80% resursa DSPA jezgra. Blok renderer troši oko 50% procesorskog vremena i 80% memorijskih resursa jezgra DSPB, dok blok obrade koristi polovinu, odnosno, oko 50% resursa DSPC jezgra.

Rešenje je verifikovano koristeći namenski alat za automatsko izvršavanje testova (BBT) koji radi na principu crne kutije. Nakon uspešno izvršene unapred definisane testne procedure koja se sastoji od 3621 testnih slučajeva, a potom i analize dobijenih rezultata, zaključeno je da sistem daje iste rezultate kao i referentni kod.

Ovakav sistem predstavnik je novije audio tehnologije koja je doprinela značajno kvalitetnijoj obradi zvučnog signala. Uvođenjem koncepta 3D objekta omogućena je tačna prostorna reprodukcija audio signala nezavisno od ciljne postavke zvučnika. Obezbeđena je fleksibilnost, bolji kvalitet obrade i veća interakcija sa korisnikom.

U okviru daljih unapređenja spada optimizacija koda u cilju smanjenja MIPS-a, zauzimanje manje memorije i mogućnost realizacije na drugim platformama sa ograničenim resursima.

7. Literatura

- [1] A.D. Blumlein, Improvements in and relating to sound – reproduction systems, british patent 394,325, 1933.
- [2] J. C. Steinberg and W. B. Snow : Auditory Perspective – Physical Factors, Electrical Engineering, pp 12-17, January 1934.
- [3] M.A. Gerzon, Ambisonics: Part two: Studio techniques, Stydio Sound August 1975:24-30, 1975.
- [4] Dolby, Dolby digital – the sound of the future here today, Dolby Laboratories Inc.,1998.
- [5] Guillaume Potard: 3D-audio object oriented coding, University of Wollongong, 2006.
- [6] J. Vilkamo, A. Kuntz, and S. Füg, “Reduction of spectral artifacts in multichannel downmixing with adaptive phase alignment,” *J. Audio Eng. Soc.*, vol. 62, no. 7/8, pp. 516–526, Jul. 2014.
- [7] S. Zielinski, F. Rumsey, and S. Bech, “Effects of down-mix algorithms on quality of surround sound,” *J. Audio Eng. Soc.*, vol. 51, no. 9, pp. 780–789, Sep. 2003.
- [8] C. Avendano and J.-M. Jot, “A frequency-domain approach to multichannel upmix,” *J. Audio Eng. Soc.*, vol. 52, no. 7/8, pp. 40–749, Jul. 2004.
- [9] *Coding of Audio-Visual Objects*, ISO/IEC JTC1/SC29/WG11 (MPEG), International Standard ISO/IEC IS 14496, 1999.
- [10] ITU-R, Recommendation-BS.775-2. Multichannel stereophonic sound system with and without accompanying picture" ITU-R BS.1770, 2006.
- [11] Rumsey, F., *Spatial Audio*", Focal Press, Oxford, 2001.
- [12] Florian Hollerweger, *An Introduction to Higher Order Ambisonic*, 2008.

-
- [13] Ville Pulkki, Virtual sound source positioning using vector base amplitude panning, Audio Engineering Society, Inc., 1997
- [14] CS49844 Data Sheet, Cirrus Logic, Inc., May 2012.
- [15] V. Kovačević, M. Temerinac, M. Popović, N. Teslić, Arhitekture i algoritmi DSP-a I, Novi Sad, FTN, 2004.
- [16] J. Kovačević, D. Bokan, Arhitekture i algoritmi digitalnih signal procesora, Zbirka zadataka i laboratorijski priručnik, Novi Sad, 2016.
- [17] M. Popović, J. Kovačević, A Statistical Approach to Model-Based Robustness Testing, 14th Annual IEEE International Conference and Workshop on Engineering of Computer Based Systems, 2007.
- [18] Owner's Manual Version 2.2 for Windows, Echo Digital Audio Corporation, September 2009.