



УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: Срђан Шуваков
Број индекса: РА 174-2014

Тема рада: Реализација модула за рендеровање звучних објеката на SoC архитектури заснованој на ARM Cortex-A породици језгара

Ментор рада: доц. др Јелена Ковачевић

Нови Сад, јул, 2018



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Срђан Шуваков		
Ментор, МН:	др Јелена Ковачевић		
Наслов рада, НР:	Реализација модула за рендеровање звучних објеката на SoC архитектури заснованој на ARM Cortex-A породици језгара		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2018		
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)	7/26/14/5/15/0/0		
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 модула за рендеровања аудио објеката на рачунару заснованом на ARM Cortex-A породици језгара.		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	Проф. др Небојша Ђевалица	
	Члан:	Доц. др Иван Каштелан	Потпис ментора
	Члан, ментор:	Доц. др Јелена Ковачевић	



KEY WORDS DOCUMENTATION

Accession number, ANO:		
Identification number, INO:		
Document type, DT:	Monographic publication	
Type of record, TR:	Textual printed material	
Contents code, CC:	Bachelor Thesis	
Author, AU:	Srđan Šuvakov	
Mentor, MN:	PhD Jelena Kovečević	
Title, TI:	Realisation of an object aduiio rendering module on an SoC architecture based on ARM Cortex-A family of cores	
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:	2018	
Publisher, PB:	Author's reprint	
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/26/14/5/15/0/0	
Scientific field, SF:	Electrical Engineering	
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW:	DSP, audio objects, rendering	
UC		
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N:		
Abstract, AB:	In the last decade, there have been major changes in both quality and quantity of audio and video content consumption. One of the key features of newer technology is accurate spatial reproduction of the sound signal, regardless of the speaker configuration. This was made possible by the 3D object concept. Using rendering, objects are dynamically allocated to the speakers present in the system configuration. This paper describes an implementation of object audio rendering module on SoC architecture based on ARM Cortex-a family of cores.	
Accepted by the Scientific Board on, ASB:		
Defended on, DE:		
Defended Board, DB:	President:	Nebojša Pjevalica PhD
	Member:	Ivan Kaštelan PhD
	Member, Mentor:	Jelena Kovečević PhD
		Menthor's sign

Zahvalnost

Zahvaljujem se razvojno-istraživačkom institutu „RT-RK” iz Novog Sada, kao i mentoru Jeleni Kovačević, tehničkom mentoru Dejanu Bokanu i kolegama iz tima Nedeljku Mutlaku i Lazaru Švonji na stručnoj pomoći tokom izrade ovog rada.



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



SADRŽAJ

1.	Uvod	1
2.	Teorijske osnove.....	2
2.1	Audio zasnovan na objektima.....	2
2.2	Gstreamer	4
2.2.1	Osnovne karakteristike Gstreamer radnog okvira	5
2.2.2	Osnove radnog okvira Gstreamer	6
2.3	Procesor zasnovan na ARM arhitekturi	7
3.	Koncept rešenja	9
3.1	Pregovaranje i izmena mogućnosti (engl. <i>Caps, capabilities</i>).....	10
3.1.1	Konstrukcija bit maske u zavisnosti od prosleđene postavke zvučnika	11
3.2	Prenos meta podataka između elemenata Gstreamer okruženja	12
3.3	Obrada podataka	12
4.	Programsko rešenje.....	14
4.1	OAR modul	14
4.1.1	Inicijalizacija OAR modula.....	15
4.1.2	Inicijalizacija klase - <code>gst_oar_class_init</code>	15
4.1.3	Inicijalizacija strukture - <code>gst_oar_init</code>	15
4.1.4	Pregovaranje i izmena mogućnosti - <code>gst_oar_transform_caps</code>	16
4.1.5	Konstrukcija bit maske kanala na izlazu	16
4.1.6	Računanje veličine izlaznog bloka - <code>gst_oar_transform_size</code>	16
4.1.7	Funkcija obrade - <code>gst_oar_transform</code>	17
4.2	Pomoćni modul za učitavanje objektnih datoteka	18
4.2.1	Funkcija <code>gst_oarloadfile_start</code>	18
4.2.2	Funkcija <code>gst_oarloadfile_fill</code>	18
5.	Testiranje i rezultati	20
5.1	Slušni testovi	21



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



5.2	Poređenje signala u spektralnom domenu.....	21
5.3	Testovi identičnosti na nivou bita	22
5.4	Potrošnja sistemskih resursa	23
6.	Zaključak	25
7.	Literatura	26

SPISAK SLIKA

Slika 1 – Putanja jednog audio objekta.....	3
Slika 2 - Blok šema audio podistema na bazi objektnog renderovanja	4
Slika 3 – <i>Gstreamer</i> radni okvir	5
Slika 4 - Primer protočne strukture	6
Slika 5 - Komunikacija u <i>Gstreamer</i> radnom okviru	7
Slika 6 – Operacija sabiranja 4 cela broja kod <i>SIMD</i> strukture	8
Slika 7 - Blok dijagram protočne strukture sa OAR modulom.....	9
Slika 8 - Primer obrade jednog bloka odabiraka.....	12
Slika 9 - Izgled procesa obrade za jedan blok podataka	13
Slika 10 - Izgled strukture lokalnog konteksta klase	15
Slika 11 - <i>gst_oar_transform</i> funkcija	17
Slika 12 - Izgled ispinog okurženja	20
Slika 13 - Prikaz spektralne karakteristike referentnog i izlaznog signala koristeći alat Audacity	22
Slika 14 - Primer rezultata jednog od testiranja.....	22
Slika 15 - Process testiranja modula za jedan ulazni signal	23

SPISAK TABELA

Tabela 1 - Formati ulaza i izlaza.....	10
Tabela 2 - Primer ispregovaranih mogućnosti za 11 objekata na ulazu i 2 kanala na izlazu	11
Tabela 3 - Preklapanje kanala referentnog modula na gstreamer modul.....	11
Tabela 4 - Niz pročitanih podataka i izlazni niz podataka za dva kanala i maksimalan broj objekata četiri	19
Tabela 5 - Utrošak procesorskog vremena i trajanje obrade za različite ulazne tokove.	23

SKRAĆENICE

- OAR** – *Object audio renderer*, modul za renderovanje objektnog audija.
- RISC** – *Reduced instruction set computing*, računanje sa smanjenim brojem instrukcija
- ARM** – *Advanced RISC Machine*, napredna RISC mašina
- MIPS** – *Milion Instructions per Second*, milion instrukcija po sekundi
- XML** – *Extendensible Markup Language*, format jezika za tekstualnu predstavu podataka
- FTP** – *File transfer protocol*, protokol za prenos podataka
- HTTP** – *Hyper text transfer protocol*, protokol za brz prenos teksta
- SOC** – *System on chip* – integrисани sistem na čipu

1. Uvod

U ovom radu realizovana je implementacija modula za renderovanje audio objekata na SoC platformi zasnovanoj na ARM Cortex-A porodici jezgara.

Objektno audio kodovanje podrazumeva pakovanje zvuka u nezavisne zvučne objekte. Svaki zvučni objekat je praćen dodatnim informacijama o samom objektu, koje između ostalog sadrže položaj izvora zvuka u prostoru, relativno u odnosu na slušaoca.

Zadatak modula za renderovanje jeste da na osnovu primljenih objekata, dodatnih informacija i rasporeda zvučnika u prostoriji generiše zvučnu scenu, odnosno audio signale koje je potrebno proslediti na svaki od zvučnih kanala na izlazu.

Jedan od ciljeva ovog rada je omogućavanje izvršavanja realizovanog modula u realnom vremenu. U tu svrhu koristi se radni okvir *Gstreamer*. *Gstreamer* predstavlja multimedijalni radni okvir koji u sebi sadrži razne multimedijalne alate. *Gstreamer* predstavlja često korišćeno okruženje prilikom izrade rešenja za namenske uređaje koji koriste *Linux* operativni sistem. Primenuje se u multimedijalnim aplikacijama kao što su [1] i [2], ali i drugim sistemima kod kojih postoji potreba za reprodukcijom ili zapisom multimedijalnih podataka u realnom vremenu kao što je prikazano u [3] i [4].

Rad se sastoji iz sedam poglavlja. U drugom poglavlju je opisan koncept audio signala zasnovanih na objektima, *Gstreamer* radni okvir i dat je kratak opis ARM porodice jezgara. Treće poglavlje predstavlja koncept rešenja projekta gde je dat opis rešenja nekih od problema koji su se javili pri izradi rada, kao i opis algoritma za obradu signala. Četvrto poglavlje opisuje kontrolnu strukturu korišćenu u implementaciji modula i opisuje implementaciju algoritma prolazeći kroz glavne funkcije modula. U petom poglavlju predstavljene su metode koje su korišćene za testiranje ispravnosti implementiranog modula. Šesto poglavlje posvećeno je zaključku gde je opisan tok izrade rada i dalje potrebe modula. Poslednje poglavlje ovog rada sadrži literaturu korišćenu tokom izrade ovog rada.

2. Teorijske osnove

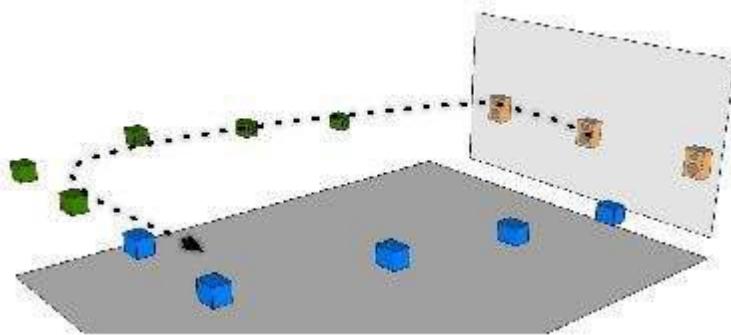
U ovom poglavlju opisani su teorijski pojmovi neophodni za razumevanje daljeg rada. U prvom delu poglavlja, predstavljene su audio tehnologije, zasnovane na audio objektima, kao i njihove razlike u odnosu na tradicionalne audio tehnologije, zasnovane na kanalima. U nastavku je dat opis *Gstreamer* okvira za reprodukciju multimedijalnog sadržaja. Deo zadatka jeste pokretanje realizovanog rešenja na *SoC* arhitekturi baziranoj na *ARM Cortex-A* porodici jezgara. Iz tog razloga u ovom poglavlju dat je i kratak opis *ARM* arhitekture kao i *Raspberry Pi* platforme koja je korišćena prilikom ispitivanja ispravnosti rešenja.

2.1 Audio zasnovan na objektima

Postoje dva pristupa reprodukcije 3D audio scena, kanalno i objektno orijentisani pristup [5]. Kod kanalno orijentisanog audio sistema broj zvučnika kao i njihova pozicija u prostoru su predefinisani. Pri procesu obrade mora se odabrati ciljna postavka zvučnika i signal se mora obraditi tako da savršeno odgovara odabranoj postavci. Ovakav pristup implicira da se prethodno pripremljeni signal dobro reprodukuje samo na ciljnoj postavci, dok pri reprodukciji na drugim postavkama može doći do gubitka informacija. Ovaj problem je moguće rešiti upotrebom objektno orijentisanog pristupa audio reprodukciji [6]. Prednost kanalno orijentisanog pristupa je što zahtevaju relativno jednostavne uređaje pri reprodukciji sadržaja, jer je potrebno samo reprodukovati zvuk na zvučnicima. Zbog nemogućnosti skaliranja 3D audio scene, kanalno orijentisani pristup je primenljiv samo ako je 3D scena statična i nije ju moguće menjati od strane korisnika (primer pesma ili film), i ako je poznata postavka zvučnika na kojoj će se reprodukovati (npr. slušalice, mono, 5.1 postavka zvučnika).

Kod objektno orijentisanog sistema audio objekti se prave nezavisno od postavke zvučnika. Audio sadržaj se sastoji iz audio objekata i pratećih meta podataka. Audio objekti sadrže audio podatke koje je potrebno reprodukovati, dok meta podaci opisuju poziciju kao i

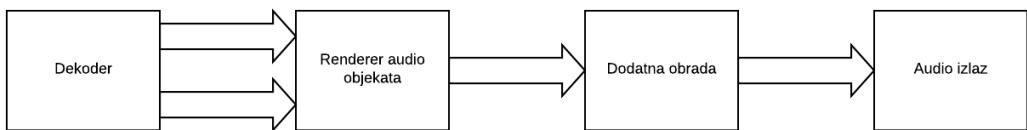
svojstva audio objekta kao što je usmerenost objekta. Ovi meta podaci sadrže informacije koje su promenljive u toku vremena, i koje kontrolišu proces renderovanja, kao što su prostorne koordinate, pojačanje i slično. Ovako pripremljen sadržaj se sa strane korisnika prilagođava trenutnoj postavci zvučnika procesom renderovanja. Zahvaljujući ovome objektno orijentisani pristup omogućava reprodukciju iste audio scene na velikom broju uređaja sa različitim postavkama zvučnika. Velika prednost objektno orijentisanog pristupa je mogućnost izmene 3D scene od strane korisnika jer su audio objekti kao i njihovi meta podaci dostupni korisniku pre reprodukcije, što čini objekti pristup mnogo boljim za interaktivne primene kao što je virtualna stvarnost. Dodatna prednost objektnog pristupa je što se svaki objekat u sceni može kodirati korišćenjem njemu prikladnog audio kodera (npr. govor i muzika se mogu kodirati različitim koderima). Međutim, manja objektno orijentisanog pristupa je što se povećava kompleksnost uređaja odgovornih za reprodukciju scene, jer je sada na uređaju da izvrši renderovanje za određenu postavku zvučnika na osnovu meta podataka.



Slika 1 – Putanja jednog audio objekta[7]

Tehnologije novije generacije se sve više zasnivaju na audio objektima jer postoji potreba za tehnologijom koja može da se prilagodi trenutnoj postavci zvučnika porasla. Naime korisnik koji želi da reprodukuje neku audio datoteku može da poseduje različite uređaje na kojima želi da reprodukuje tu istu datoteku. Zbog ograničenja kanalnog pristupa to dovodi do smanjenja kvaliteta sveukupnog doživljaja. Dok kod objektno orijentisanog pristupa ovo ne predstavlja problem.

Modul za renderovanje audio objekata (OAR) na ulazu prima audio objekte praćene dodatnim informacijama koje opisuju taj objekat, njegov sadržaj i parametre koji opisuju na koji način taj objekat utiče na scenu, na osnovu kojih OAR modul vrši renderovanje dobijenih objekata za postojeću postavku zvučnika na izlazu. OAR modul je u lancu obrade ko audio prijemnika koji koristi tehnologije zasnovane na audio objektima, smešten odmah nakon dekoderskog modula, a pre primene algoritama završne obrade. Primer blok šeme audio podsistema koji sadrži OAR modul prikazan je na slici 2.



Slika 2 - Blok šema audio podistema na bazi objektnog renderovanja

Audio podaci na ulazu u OAR modul predstavljeni su u nekom od nekompresovanih formata, tako da se podacima koji su deo nekog objekta može pristupiti nezavisno od ostalih objekata. Dodatne informacije o audio objektu se nazivaju meta podaci, ovi meta podaci mogu sadržati u sebi informacije o poziciji u prostoru, promeni pozicije ako postoji, vrsti kodera koji je korišćen za kodiranje objekta. U zavisnosti od sadržaja meta podataka audio objekti se dele na:

1. Fiksirani audio objekti (engl. *bed*) – objekti koji se prosleđuju na tačno određeni zvučnik.
2. Dinamički audio objekti – sadrže dodatnu informaciju o poziciji u prostoru, ta informacija se može menjati u vremenu.
3. ISF objekat - format koji omogućava predstavu čitave audio scene, a ne pojedinačnih objekata.

2.2 Gstreamer

Gstreamer [8] je radni okvir (engl. *framework*) za pravljenje medijskih aplikacija za reprodukciju u realnom vremenu. Radni okvir se bazira na modulima koji pružaju razne funkcionalnosti kao što su audio ili video dekoderi, filteri, itd. Osnovna funkcionalnost *Gstreamer* radnog okvira je da obezbedi funkcionalnost modula i rukovanje protokom podataka. Pored osnovne namene, a to je olakšavanje razvoja multimedijalnih aplikacija koristeći već realizovane biblioteke, *Gstreamer* obezbeđuje i spregu za pisanje korisničkih modula.

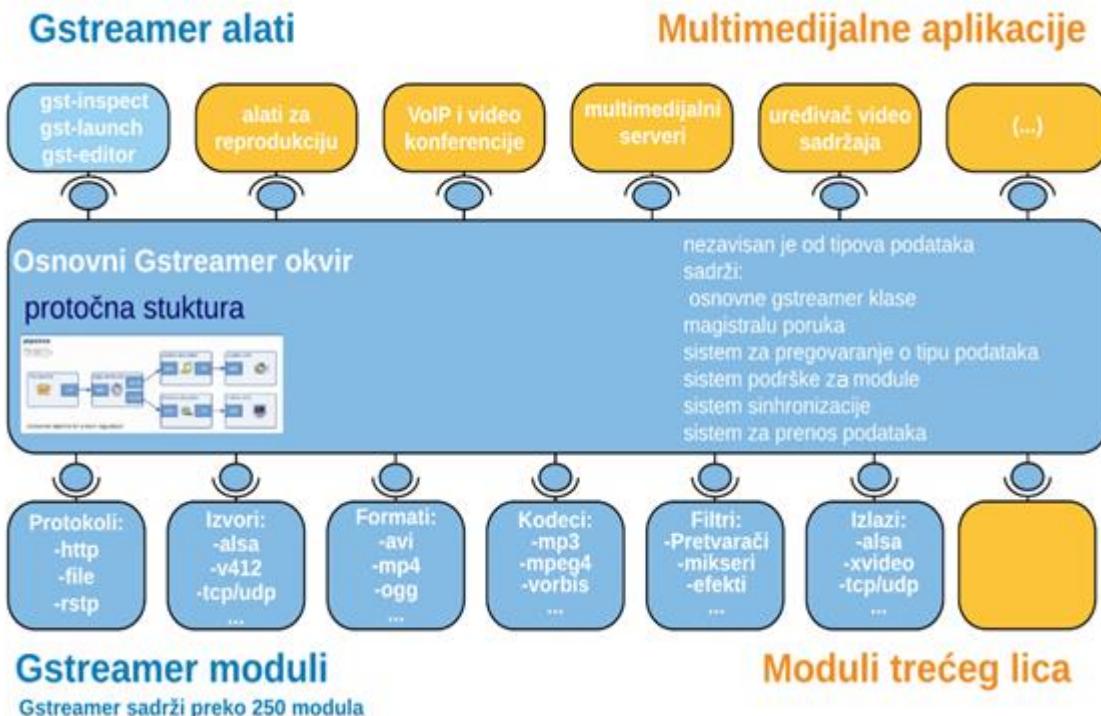
Gstreamer obezbeđuje

- Arhitekturu za rukovanje modulima.
- Arhitekturu za rukovanje protočnom strukturom.
- Mehanizam sinhronizacije.

Gstreamer moduli se dele na:

- **Rukovaoce protokolom prenosa podataka** – Moduli koji omogućavaju prenos podataka određenim protokolom (na primer *FTP*).
- **Izvore (uključuje i rukovaoce protokolom)** – Moduli koji omogućavaju dobavljanje podataka sa određenog izvora (npr. koristeći *HTTP*).
- **Formatere** – Moduli koji menjaju format ulaznih podataka kao što su parseri, multiplekseri i demultiplekseri.
- **Kodere i dekodere** – Moduli za kodiranje i dekodovanje podataka.
- **Filtere** – Moduli koji vrše završnu obradu signala (engl. *post-processing*).
- **Izlaze** – Moduli koji omogućavaju slanje podataka na neki od mogućih izlaza (npr. zvučnike).

Na slici 2 prikazana je organizacija radnog okvira *Gstreamer*, i njegovih pojedinačnih komponenti.



Slika 3 – *Gstreamer* radni okvir

2.2.1 Osnovne karakteristike *Gstreamer* radnog okvira

Gstreamer radni okvir se pridržava objektnog modela *Gobject* [9] zasnovanog na biblioteci Glib 2.0. *Gstreamer* radni okvir koristi mehanizme signalizacije i osobina objekata. Jedna veoma bitna karakteristika *Gstreamer* radnog okvira je proširivost. Svi objekti *Gstreamer* radnog okvira mogu se proširiti metodom nasleđivanja karakterističnim za objektni model *Gobject*. Dodatno, svi moduli se učitavaju dinamički što omogućava njihovo nezavisno

proširivanje i unapređivanje. Iako se *Gstreamer* radni okvir pretežno koristi za rad sa multimedijalnim podacima, on podržava sve tipove podataka.

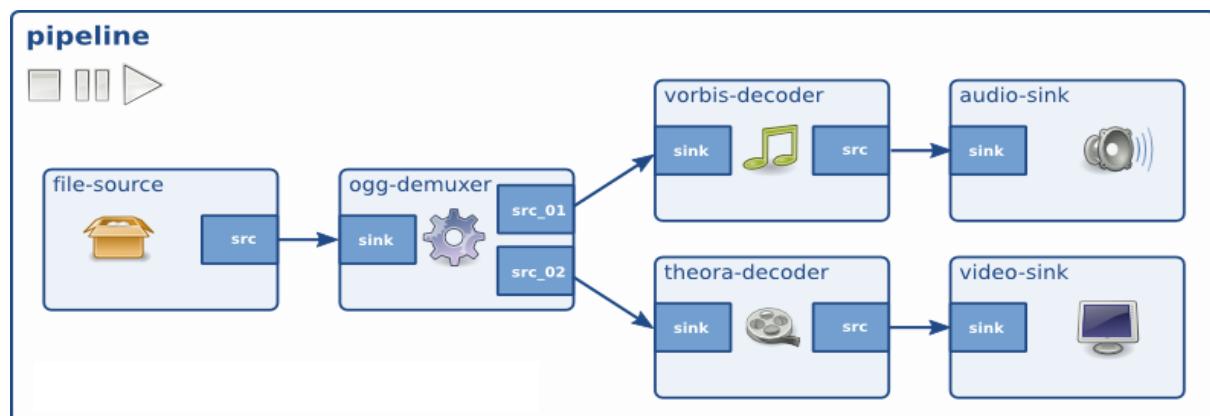
2.2.2 Osnove radnog okvira *Gstreamer*

Gstreamer radni okvir se zasniva na nekoliko gradivnih celina, razumevanje ovih celina je neophodno za dalji rad sa *Gstreamer* radnim okvirom. Najbitnija gradivna celina je pojam elementa. Svaki element ima jednu specifičnu funkciju, koja može biti čitanje iz datoteke, dekodovanje podataka, obrada tih podataka ili slanje tih podataka ka zvučniku ili datoteci. Ulančavanjem više elemenata stvara se protočna struktura (engl. *pipeline*) koji služi nekoj specifičnoj svrsi kao na primer snimanje ili reprodukcija medija.

Na ulazu ili izlazu elementa se nalaze prolazi (engl. *Pad*). Prolazi služe za pregovaranje veza između elemenata. Svaki prolaz ima specifičan niz mogućnosti (engl. *capabilities*, skraćeno *caps*), prolaz može da zabrani protok određenih tipova podataka kroz njega, ili da specificira tačno koji tip može da prolazi kroz njega. Uvezivanje elemenata je samo dozvoljeno ako im se poklapaju mogućnosti na odgovarajućim prolazima. Ovo poklapanje se proverava procesom pregovaranja mogućnosti (engl. *Caps negotiation*).

Elementi u *Gstreamer* radnom okviru mogu da se grupišu u skupove (engl. *Bins*). Prednost ovakvog pristupa je u tome što je ovakvim skupovima moguće upravljati kao i sa elementima. Skupovi omogućavaju uvođenje dodatnog nivoa apstrakcije, koji omogućava predstavu složenih blokova (delova lanca obrade) kao pojedinačnih elemenata što olakšava rukovanje njima. Na primer, moguće je promeniti stanja svih elemenata u jednom skupu elemenata menjanjem stanja tog skupa.

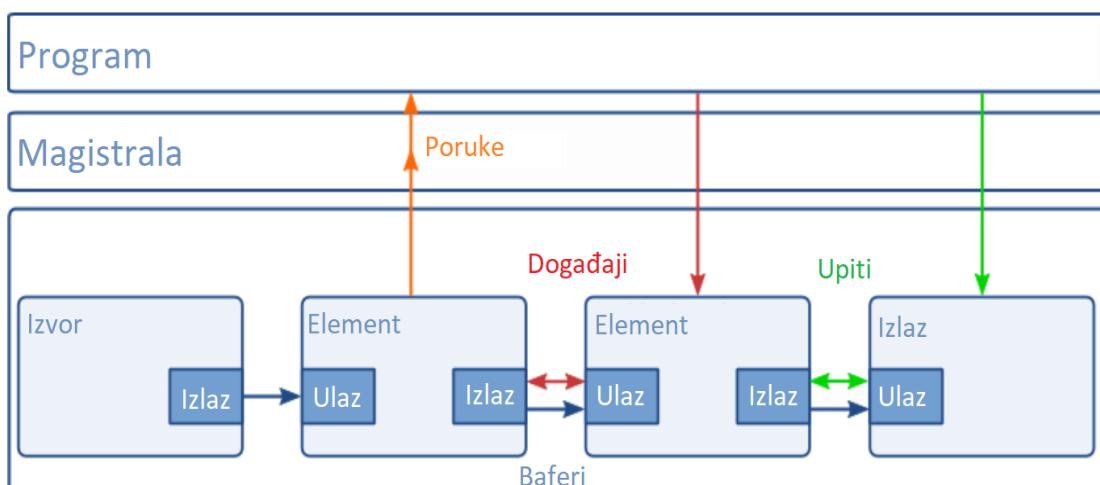
Protočna struktura (engl. *pipeline*) je skup elemenata na najvišem nivou (engl. *top-level*). Ona obezbeđuje magistralu za protok podataka i stara se o sinhronizaciji između elemenata. Nakon pokretanja, protočne strukture rade u zasebnoj niti.



Slika 4 - Primer protočne strukture

Gstreamer radni okvir pruža nekoliko mehanizama komunikacije i prenosa podataka između elemenata, kao i između protočne strukture i programa koji je pokreće.

- **Baferi** su objekti koji služe prenosu podataka između elemenata preko protočne strukture.
- **Događaji** su objekti koje elementi razmenjuju međusobno ili koje program šalje elementima.
- **Poruke** su objekti koje elementi šalju na magistralu poruka protočne strukture koja ih zadržava dok ih program ne pokupi. Poruke se koriste za prenos informacija o greškama, promenama stanja, preusmeravanju protoka i sličnim informacijama vezanim za rad protočne strukture.
- **Upiti** dozvoljavaju programu da zahteva informacije od protočne strukture poput informacije o trajanju reprodukcije i trenutnoj poziciji reprodukcije. Takođe elementi mogu međusobno prosleđivati upite kako bi dobili informacije o signalu koji se trenutno nalazi u protočnoj strukturi. Mogu se koristiti u oba smera u protočnoj strukturi ali su upiti upućeni u smeru početka protočne strukture češći.



Slika 5 - Komunikacija u Gstreamer radnom okviru

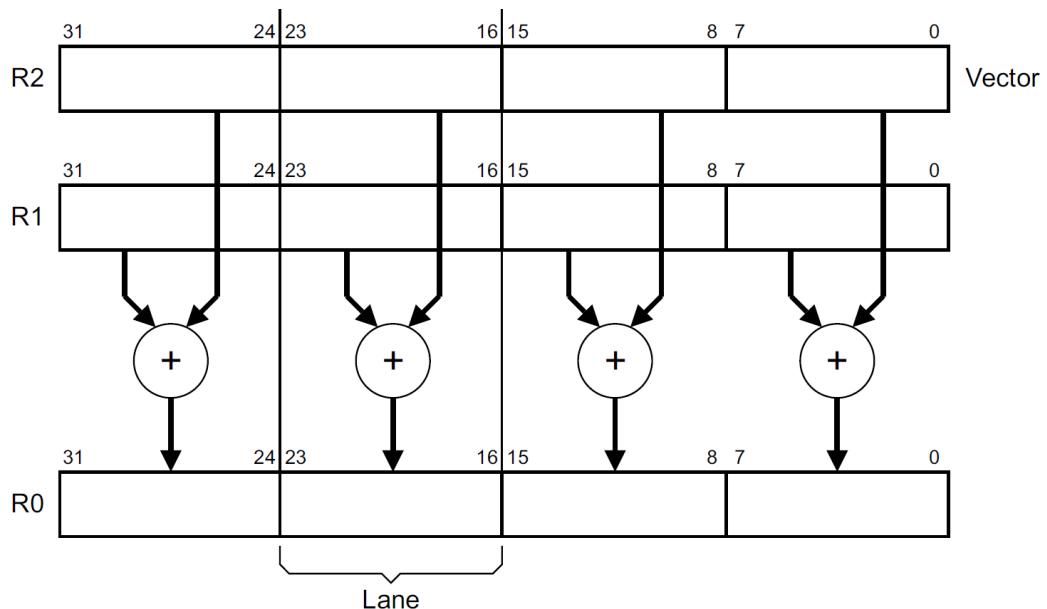
2.3 Procesor zasnovan na ARM arhitekturi

ARM (engl. *Advanced RISC Machine* – Napredna RISC mašina) procesor potiče iz porodice procesora zasnovanih na RISC arhitekturi. (engl. *Reduced Instruction Set Computer - RISC*)[10]. RISC arhitekture karakterišu jednostavne instrukcije koje se izvršavaju za svega nekoliko instrukcionih ciklusa. Činjenica da predstavljaju RISC arhitekture označava i da ARM procesori zahtevaju manji broj tranzistora u svojim integrisanim kolima, što dovodi do smanjene veličine samog čipa i smanjene potrošnje energije. Bitna karakteristika RISC arhitekture je iskorišćenje paralelizma na nivou instrukcije. RISC arhitektura se često naziva i

učitaj/skladišti (engl. *Load/Store*) arhitekturom. Smanjena veličina čipa, manja kompleksnost i potrošnja električne energije čine ovaj procesor idealnim za prenosive uređaje kao što su pametni telefoni i satovi, ali i za male računare poput *Raspberry Pi* računara.

U okviru ovog rada koncentracija je usmerena na ARMv7-A arhitekturu, sa klasičnim ARM 32-bitnim instrukcijskim setom [11]. Bitna odlika ARM arhitektura jeste mehanizam ko-procesora, čijom upotreborom je moguće implementirati nova procesorska proširenja bez ponovnog projektovanja cele arhitekture. Jedna od namena ko-procesorskih proširenja jeste i omogućavanje izvršavanja vektorskih instrukcija (SIMD). Ko-procesorska proširenja za vektorske instrukcije čine ARM procesore pogodne za obavljanje zadataka koji se tiču obrade signala. Postoje dva tipa ko-procesora za vektorske instrukcije:

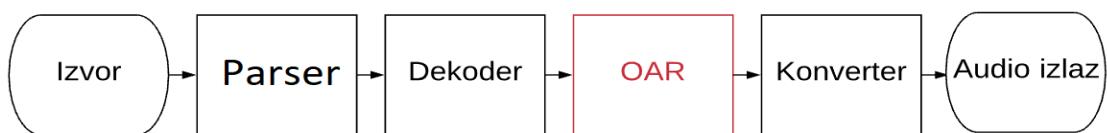
- (1) *Vector Floating Point* – Je ko-procesor koji obezbeđuje obradu brojeva sa pokretnim zarezom (engl. *Floating Point*) sa jednostrukom ili dvostrukom preciznošću. Naime ovo proširenje uvodi poseban skup 32-bitnih registara za jednostruku preciznost, od kojih se svaka dva mogu koristiti u paru za dvostruku preciznost.
- (2) ARM NEON je ko-procesorsko proširenje za *ARM Cortex-A* seriju procesora zasnovano na SIMD (engl. *Single instruction multiple data*) arhitekturi. NEON proširenje je zasnovano na vektorskim operacijama nad ulaznim podacima, tj. instrukcije NEON ko-procesora izvršavaju iste operacije nad svim putanjama vektora. Stoga je ovo proširenje veoma pogodno za poslove koji zahtevaju ponavljanje iste instrukcije nad različitim nizovima podataka, što je čest slučaj kod audio i video obrade.[12]



Slika 6 – Operacija sabiranja 4 cela broja kod SIMD strukture[13]

3. Koncept rešenja

Cilj ovog rada je implementacija OAR modula za uređaje zasnovane na *ARM Cortex A53* porodici jezgara na osnovu priloženog referentnog modula. Kao osnova za implementaciju modula izabran je *Gstreamer* radni okvir jer pruža alate koji pored činjenice da olakšavaju implementaciju samog modula, pruža podršku za rad u realnom vremenu.



Slika 7 - Blok dijagram protočne strukture sa OAR modulom

Zadatak OAR modula je da signal sa ulaza kao i njegove meta podatke obradi za specifičnu postavku zvučnika i prosledi obrađeni signal na izlaz. U zavisnosti od konfiguracije zvučnika i meta podataka koji opisuju signal, format signala na izlazu može biti u potpunosti različit u odnosu na onaj na ulazu. *Gstreamer* nudi određene predefinisane šablonske klase koje omogućavaju lakšu realizaciju novih modula. Jedna takva klasa je *BaseTransform*. *BaseTransform* je služi kao osnova filterskim elementima kod kojih su veličina i mogućnosti izlaza poznate u potpunosti po mogućnostima ulaza i po veličinama bafera. To su elementi koji direktno menjaju jedan bafer u drugi, elementi koji vrše promenu bafera direktno (engl. *in-place*), i elementi koji spajaju više ulaznih bafera u jedan izlazni bafer ili prave više izlaznih bafera na osnovu jednog ulaznog. *BaseTransform* pa samim tim i OAR modul spada u kategoriju filtera. Osnovni zadaci koje je neophodno realizovati u okviru OAR modula su:

- Inicijalizacija početnog stanja modula.
- Pregovaranje i određivanje mogućnosti na ulaznim i izlaznim prolazima modula u zavisnosti od mogućnosti susednih elemenata, sadržaja bitskog toka i parametara modula.

- Izdvajanje i parsiranje meta podataka iz ulaznog bitskog toka.
- Obrada podataka.
- Prosleđivanje obrađenih podataka narednom modulu u lancu obrade.

3.1 Pregovaranje i izmena mogućnosti (engl. *capabilities*)

U tabeli 1 dati su šabloni formata podržanih na ulazu i izlazu OAR modula. Ulaz OAR modula se nasleđuje od prošlog elementa u protočnoj strukturi i zavisi od broja audio objekata koji će biti prosleđivani modulu, kao i formata u kom su ti audio objekti predstavljeni. Izlazne mogućnosti se dobijaju iz konfiguracionih parametara koji se prosleđuju modulu.

Primer mogućnosti ulaza i izlaza nakon pregovaranja dat je u tabeli 2. Radni okvir *Gstreamer* zahteva usklađenost mogućnosti svih elemenata protočne strukture radi njihovog uvezivanja. Kao što se u tabeli 2 može videti, dogovoren formati ulaza i izlaza se mogu razlikovati. Na primer, velika je verovatnoća da će se broj kanala na ulazu i izlazu biti različiti. Ova razlika nastaje jer OAR modul na ulazu očekuje audio objekte, kako *Gstreamer* ne zna za pojam audio objekata, svaki ulazni objekat se prenosi u posebnom kanalu. Prilikom slanja jednog bloka od N odbiraka, šalje se po N odbiraka svakog od objekata koji su u kanale pakovani kao da su N odbiraka tog kanala. Kanali očekivani na ulazu su poređani redom kojim su objekti postavljeni, tj. ne obraća se pažnja na kom kanalu se nalazi koji objekat jer krajnja pozicija zvuka zavisi od meta podataka. Da bi se ovako definisan broj kanala ulaza poklapao sa brojem kanala izlaza dobijenim iz konfiguracionih parametara potrebno je prilagoditi broj kanala na ulazu. Osim broja kanala može se razlikovati i raspored zvučnih odbiraka u ulaznom nizu podataka, tačnije odbirci jednog kanala mogu biti poređani jedni za drugim ili učešljani. Svaka stavka koju je potrebno prilagoditi, se prilagođava tako što se iz strukture koja predstavlja mogućnosti jednog prolaza izbaci ta stavka i ubaci odgovarajuća stavka koja se dobija sa izlaza. Prilikom izmene broja kanala mora se i promeniti bit maska zvučnika kako bi odgovarala novom broju kanala.

Mogućnosti prolaza	Ulaz	Izlaz
Tip signala	Audio/x-raw	Audio/x-raw
Format	S32LE	S32LE
Brzina odabiranja	1 do 96000	1 do 96000
Broj kanala	1 do 23	1 do 23
Raspored odbiraka	Učešljjan, jedan za drugim	Učešljjan

Tabela 1 - Formati ulaza i izlaza

Mogućnosti prolaza	Ulaz	Izlaz
Tip signala	Audio/x-raw	Audio/x-raw
Format	S32LE	S32LE
Brzina odabiranja	48000	48000
Broj kanala	11	2
Raspored odabiraka	Učešljjan	Učešljjan

Tabela 2 - Primer ispregovaranih mogućnosti za 11 objekata na ulazu i 2 kanala na izlazu

3.1.1 Konstrukcija bit maske u zavisnosti od prosleđene postavke zvučnika

OAR modul ima podršku za obradu do 35 kanala, dok *Gstreamer* ima podršku za 28 različitih kanala. Iz tog razloga potrebno je prilagoditi bit masku OAR modula kako bi se mogla koristiti u *Gstreamer* radnom okviru. Bit maska OAR modula se formira u zavisnosti od karakternog niza koji je prosleđen *Gstreamer* radnom okviru preko komandne linije, taj isti karakterni niz se koristi za formiranje bit maske koja se prosleđuje na izlaz iz modula. Koriste se samo kanali koje *Gstreamer* podržava.

Oznaka kanala	Gstreamer kanal	kanal referentnog modula
L	GST_AUDIO_CHANNEL_POSITION_FRONT_LEFT	AR_SPEAKER_L
R	GST_AUDIO_CHANNEL_POSITION_FRONT_RIGHT,	AR_SPEAKER_R
C	GST_AUDIO_CHANNEL_POSITION_FRONT_CENTER	AR_SPEAKER_C
LFE	GST_AUDIO_CHANNEL_POSITION_LFE1,	AR_SPEAKER_LFE
Lrs	GST_AUDIO_CHANNEL_POSITION_REAR_LEFT,	AR_SPEAKER_LRS
Rrs	GST_AUDIO_CHANNEL_POSITION_REAR_RIGHT,	AR_SPEAKER_RRS
Lc	GST_AUDIO_CHANNEL_POSITION_FRONT_LEFT_OF_CENTER,	AR_SPEAKER_LC
Rc	GST_AUDIO_CHANNEL_POSITION_FRONT_RIGHT_OF_CENTER,	AR_SPEAKER_RC
Cs	GST_AUDIO_CHANNEL_POSITION_REAR_CENTER,	AR_SPEAKER_CS
Lcs	GST_AUDIO_CHANNEL_POSITION_SIDE_LEFT,	AR_SPEAKER_LCS
Rcs	GST_AUDIO_CHANNEL_POSITION_SIDE_RIGHT,	AR_SPEAKER_RCS
Ltf	GST_AUDIO_CHANNEL_POSITION_TOP_FRONT_LEFT,	AR_SPEAKER_LTF
Rtf	GST_AUDIO_CHANNEL_POSITION_TOP_FRONT_RIGHT,	AR_SPEAKER_RTF
Ltr	GST_AUDIO_CHANNEL_POSITION_TOP_REAR_LEFT,	AR_SPEAKER_LTR
Rtr	GST_AUDIO_CHANNEL_POSITION_TOP_REAR_RIGHT,	AR_SPEAKER_RTR
Ltm	GST_AUDIO_CHANNEL_POSITION_TOP_SIDE_LEFT,	AR_SPEAKER_LTM
Rtm	GST_AUDIO_CHANNEL_POSITION_TOP_SIDE_RIGHT,	AR_SPEAKER_RTM
Lsc	GST_AUDIO_CHANNEL_POSITION_BOTTOM_FRONT_LEFT,	AR_SPEAKER_LSC
Rsc	GST_AUDIO_CHANNEL_POSITION_BOTTOM_FRONT_RIGHT,	AR_SPEAKER_RSC
Lw	GST_AUDIO_CHANNEL_POSITION_WIDE_LEFT,	AR_SPEAKER_LW
Rw	GST_AUDIO_CHANNEL_POSITION_WIDE_RIGHT,	AR_SPEAKER_RW
Ls	GST_AUDIO_CHANNEL_POSITION_SURROUND_LEFT,	AR_SPEAKER_LS
Rs	GST_AUDIO_CHANNEL_POSITION_SURROUND_RIGHT	AR_SPEAKER_RS

Tabela 3 - Preklapanje kanala referentnog modula na gstreamer modul

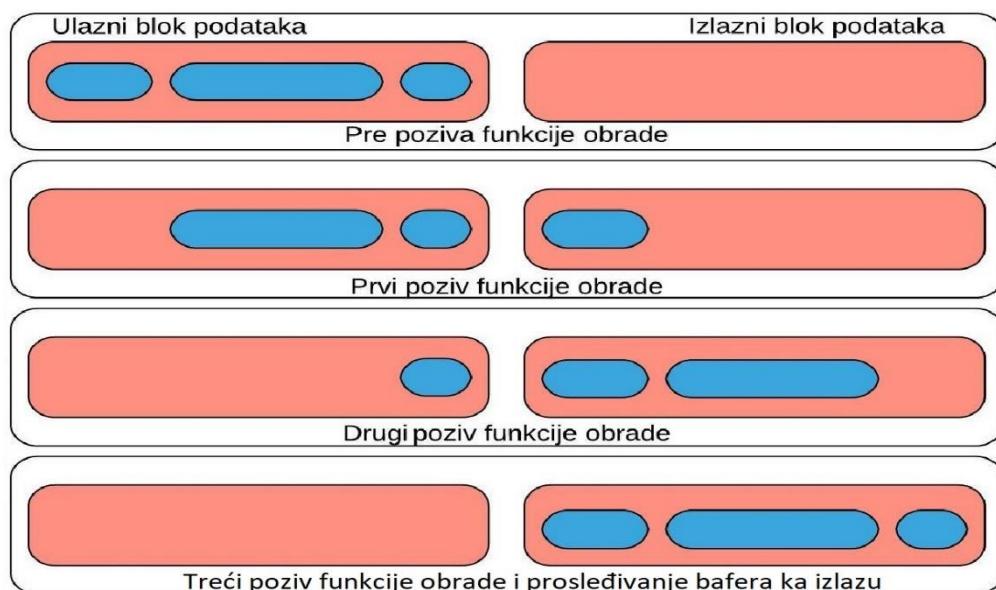
3.2 Prenos meta podataka između elemenata Gstreamer okruženja

U okviru *Gstreamer* okruženja postoji podrška za pridruživanje dodatnih podataka baferima koji služe za prenos signala između elemenata protočne strukture. Međutim ovaj mehanizam podrazumeva dodavanje kratkih podataka, poput vremenskih oznaka, kratkih opisa sadržaja i slično, i nije prilagođen za pridruživanje veće količine pratećih informacija, kao što su meta podaci. Zbog toga bilo je neophodno definisati način za prenos meta podataka uz audio objekte.

Da bi se omogućio prenos meta podataka iskorišćena je mogućnost logičnog povezivanja bafera, odnosno vezivanja životnog veka jednog bafera za drugi. To se postiže pravljenjem novog bafera dovoljne veličine da obuhvati sve meta podatke koje je potrebno preneti i povezivanjem ovog bafera sa baferom audio podataka. Ovako uvezan bafer međutim ne prenosi sve informacije koje su potrebne OAR modulu za renderovanje zvučne scene, već samo samo „teret“ koji prenose meta podaci, tj. Informacije o pozicijama odbiraka u prostoru, jačini zvuka, itd. Za dodatne meta podatke koje zahteva OAR modul pravi se mehanizam za proširivanje strukture meta podataka. Ovaj mehanizam proširuje strukturu meta podataka koju koristi radni okvir *Gstreamer* i dodaje funkcije koje obavljaju inicijalizaciju i popunjavanje strukture kao i funkcije za registraciju imena i izgleda strukture u radnom okviru *Gstreamer*.

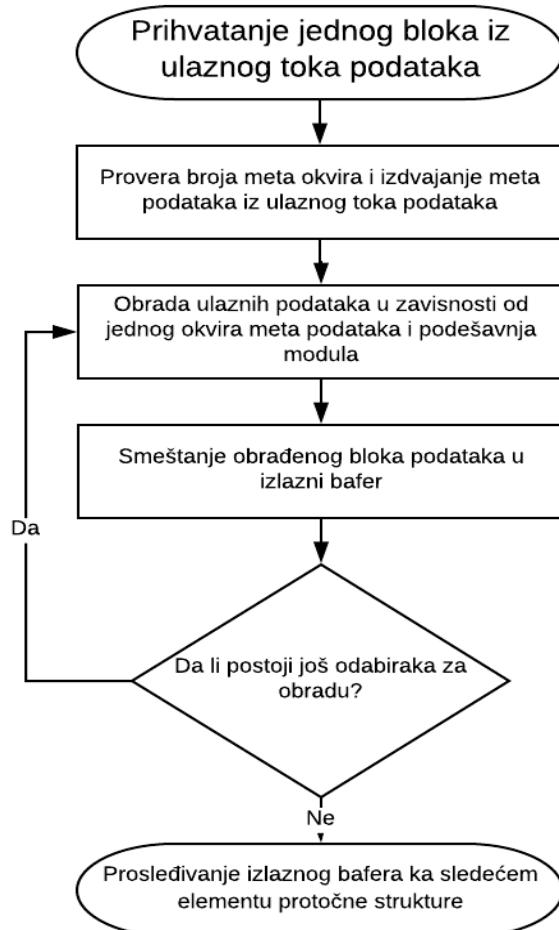
Za ovako upakovane meta podatke je u OAR modulu realizovan mehanizam za iščitavanje i parsiranje.

3.3 Obrada podataka



Slika 8 - Primer obrade jednog bloka odabiraka

Proces obrade se pokreće nakon što se prihvati jedan blok iz ulaznog toka. Prvo što je potrebno uraditi jeste proveriti koliko okvira meta podataka je prihvaćeno, jer je moguće da jedan blok audio podataka sadrži više od jednog okvira meta podataka. Nakon što imamo tačan broj okvira meta podataka, pristupa se izdvajaju meta podataka.



Slika 9 - Izgled procesa obrade za jedan blok podataka

Obrada se vrši blok po bloku kako pristižu. Po pozivu funkcije obradi se onoliko odbiraka koliko opisuje jedan okvir meta podataka, pa se funkcija obrade poziva dok se ne obradi maksimalan broj odbiraka po bloku kao što je prikazano na slici 8 ili dok se ne obrade svi odbirci spremni za obradu, tj. odbirci za koje postoje meta podaci, dok se ostali odbirci čuvaju za obradu sledećeg bloka odbiraka. Izlazni bafer koji služi za skladištenje obrađenih odbiraka ima unapred definisanu veličinu. Ukoliko se desi da se nakon obrade svih odbiraka ne popuni ceo izlazni bafer, prosleđuju se samo obrađeni odbirci.

4. Programsко rešenje

U ovom radu su realizovana dva modula, OAR modul koji služi za renderovanje audio objekata i *oarloadfile* modul koji služi za učitavanje audio objekata iz datoteka.

4.1 OAR modul

OAR modul je realizovan kao deljena biblioteka koja se može uvezati u program realizovan u jeziku C ili se može pozvati preko alata za konstrukciju i pokretanje protočne strukture radnog okvira *Gstreamer*, pod nazivom *gst-Launch*. Primer komandne linije kojom se poziva OAR modul je:

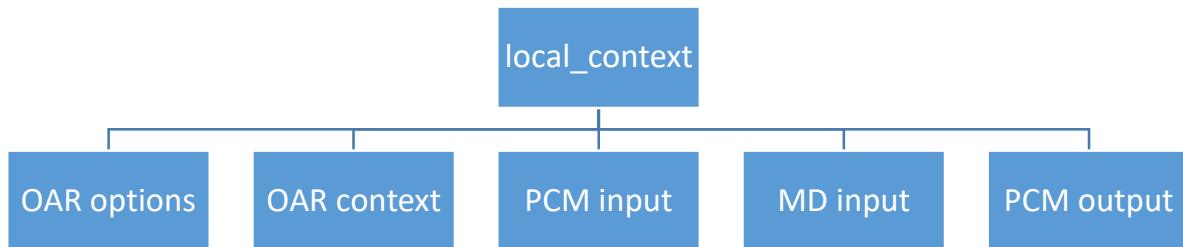
```
gst-launch-1.0 oarloadfile ! oar frameSize=1536 speakerConf=LR  
! audioconvert ! audio/x-raw,format=S24_32LE ! alsasink
```

Konfiguracioni parametri koji se mogu proslediti OAR modulu su:

- *positionSmoothing* – (TRUE/FALSE) – Ublažavanje greške prilikom promene pozicije objekta.
- *limiter* – (TRUE/FALSE) – Ograničenje opsega.
- *bassExtraction* – (0 – 200) – Nivo izdvajanja niskofrekventnog sadržaja.
- *surroundTrim* – (TRUE/FALSE) – Umanjenje intenziteta zvuka okruženja.
- *latencyCompensation* – (TRUE/FALSE) – Kompenzacija kašnjenja.
- *frameSize* – (32 – 4096) – Veličina jednog okvira meta podataka predstavljena u odbircima.
- *speakerConf* – (primer: „LR“) - karakterni niz postavke zvučnika.

Radi praćenja trenutnog stanja OAR modula uvedena je kontrolna struktura *local_context*. Na slici 9 koja prikazuje izgled ove strukture date su samo celine koje su opisane u toj strukturi da bi se izbeglo nabranjanje svakog polja strukture kao i njenih pod struktura.

- *OAR interface options* – obuhvata podešavanja vezana za modul
- *OAR context* – je podstruktura koja prati stanje modula tokom obrade bloka
- *PCM input* – obuhvata pokazivač na ulazni niz podataka, kao i informacije o formatu ulaznog niza
- *MD input* – obuhvata informacije niz vezane za meta podatke i pokazivač na ulazni niz meta podataka
- *PCM output* – obuhvata pokazivač na izlazni niz podataka, kao i informacije o formatu izlaznog niza



Slika 10 - Izgled strukture lokalnog konteksta klase

4.1.1 Inicijalizacija OAR modula

Inicijalizacija priključka se vrši u dve funkcije koje poziva sam *Gstreamer* radni okvir:

- `gst_oar_class_init`
- `gst_oar_init`

4.1.2 Inicijalizacija klase - `gst_oar_class_init`

Funkcija za inicijalizaciju klase se stara o nasleđivanju klase `BaseTransform`. Dodatno u okviru ove funkcije realizovano je preklapanje svih metoda koje očekuje `BaseTransform` kao i metoda koje očekuju potklase koje `BaseTransform` nasleđuje. Osim nasleđivanja `class_init` funkcija se stara o postavljanju mogućnosti prolaza i dodavanjem podrazumevanih svojstava klase.

4.1.3 Inicijalizacija strukture - `gst_oar_init`

Funkcija `gst_oar_init` vrši inicijalizaciju OAR modula i podešava početne postavke. Na osnovu početnih postavki `gst_oar_init` funkcija vrši inicijalizaciju strukture `local_context` koja sadrži informacije o ulaznim i izlaznim podacima, meta podacima i čuva pokazivač na `oari_state` strukturu koja u sebi sadrži sve informacije o trenutnom stanju modula. Tokom inicijalizacije se takođe računa i zauzima memorija potrebna za `oari_state` strukturu kao i privremena memorija potrebna za jedan poziv funkcije obrade.

4.1.4 Pregovaranje i izmena mogućnosti - `gst_oar_transform_caps`

Pregovaranje i izmena mogućnosti se vrši u funkciji `gst_oar_transform_caps`. Funkcija `gst_oar_transform_caps` se poziva kada god se desi promena mogućnosti na ulaznom prolazu, kako bi se omogućilo modulu da na osnovu promene konfiguriše mogućnosti na izlazu. Ukoliko dođe do promene mogućnosti na ulazu potrebno je izvršiti izmenu informacije o broju kanala, načinu na koji su odbirci raspoređeni u izlaznom nizu podataka i bit masku. Ove informacije se menjaju u zavisnosti od konfiguracionih parametara OAR modula. Kako bi se olakšala transformacija strukture podataka koja nosi informacije o mogućnostima realizovane su dodatne pomoćne funkcije:

- `remove_layout_from_structure`
- `remove_channels_from_structure`
- `add_other_channels_to_structure`
- `add_other_layout_to_structure`
- `add_other_channel_mask_to_structure`

Nakon uspešnog pregovaranja mogućnosti priključka poziva se funkcija `gst_oar_set_caps` koja obaveštava ostatak protočne strukture o podešenim mogućnostima na ulazu i izlazu.

4.1.5 Konstrukcija bit maske kanala na izlazu

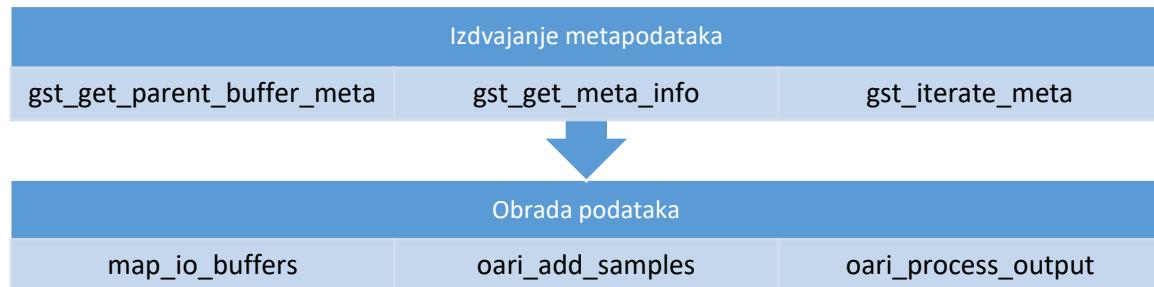
Kao što je u prethodnom poglavlju rečeno bit maske OAR modula i *Gstreamer* radnog okvira se razlikuju. Za formiranje bit maske na izlazu u zavisnosti od bit maske OAR modula koristi se funkcija `channels_and_mask_from_string`. U njoj se iščitava karakterni niz sa simbolima zvučnika, formira se maska koju koristi modul i čuva se niz simbola koji predstavljaju postavku zvučnika. Taj niz simbola se prosleđuje `string_array_to_mask` funkciji koja od tih simbola konstruiše bit masku koju *Gstreamer* koristi na izlazu.

4.1.6 Računanje veličine izlaznog bloka - `gst_oar_transform_size`

Računanje veličine izlaznog bloka se vrši u funkciji `gst_oar_transform_size`. Logika funkcije se zasniva na očuvanju broja odbiraka koji se prenose u protočnoj strukturi. Ovo se postiže tako što se dobavlja količina memorije u bajtima i podeli se sa brojem bajtova po jednom okviru prenosa, dobijeni broj se onda množi sa brojem bajtova po jednom okviru prenosa na izlazu iz modula i dobija se veličina izlaznog bloka u bajtima. Veličina memorije u bajtima se dobija pozivom funkcije `gst_oar_get_unit_size`.

4.1.7 Funkcija obrade - `gst_oar_transform`

Funkcija `gst_oar_transform` vrši obradu ulaznih audio objekata. Logika funkcije je podeljena u dve celine. Prva celina jeste izdvajanje meta podataka iz ulaznog bafera, nakon čega se pristupa renderovanju zvučne scene na osnovu izdvojenih meta podataka.



Slika 11 - `gst_oar_transform` funkcija

4.1.7.1 Izdvajanje meta podataka iz ulaznog niza podataka

Da bi se izdvojili meta podaci neophodno je prvo izdvojiti niz meta podataka iz niza ulaznih podataka funkcijom `gst_buffer_get_parent_buffer_meta`. Nakon što su meta podaci izdvojeni iz ulaznog niza, dobavlja se struktura meta podataka po imenu funkcijom `gst_meta_get_info`. Kad se dobije izgled strukture meta podataka iterira se po strukturama meta podataka iz prethodno izdvojenog niza meta podataka i traži se struktura koja odgovara strukturi dobijenoj funkcijom `gst_meta_get_info` koje se smešta u `gst_meta_from_parrent` strukturu tipa `GstOarMeta`. Nakon ovako izdvojene strukture meta podataka popunjava se niz meta podataka koji je potreban u obradi podataka u zavisnosti od broja dobijenih blokova meta podataka.

4.1.7.2 Obrada podataka - `gst_oar_transform`

Obrada podata se vrši u funkciji `gst_oar_transform` u funkcijama `oari_add_samples` i `oari_process_output` nakon izdvajanja meta podataka.

Kako su struktura ulaznog niza podataka i struktura koju ove dve funkcije različite potrebno je smestiti ulazne podatke u odgovarajuću strukturu. Da bi se izbeglo pravljene lokalnih nizova podataka za smeštanje odbiraka koje bi se izvršavalo svaki put kad se dobavi blok na ulazu i koristilo previše sistemskih resursa, napravljene su pomoćne funkcije `map_input_buffer` i funkcija `map_output_buffer` koje popunjavaju potrebna polja struktura i uvezuju pokazivače na podatke iz struktura na pokazivače na podatke ulaznog i izlaznog bloka odbiraka potrebne za funkcije `oari_add_samples` i `oari_process_output`. `oari_add_samples` na osnovu meta podataka i veličine ulaznog bloka podataka sprema i prosleđuje ulazne odbirke zvuka ka strukturi `p_oari`. `oari_process_output` vrši obradu odbiraka opisanih jednim okvirom meta podataka. Funkcija `oari_process_output` se poziva onoliko puta koliko je okvira meta podataka.

podataka prosleđeno u bloku koji se trenutno obrađuje. Nakon što je obrađen poslednji okvir meta podataka u jednom bloku, ukoliko je broj obrađenih odbiraka različit od veličine izlaznog bloka odbiraka koristi se funkcija `gst_buffer_set_size` da bi se promenila veličina izlaznog bloka.

4.2 Pomoćni modul za učitavanje objektnih datoteka

Za potrebe testiranja ispravnosti implementiranog modula napravljen je pomoćni modul nazvan `oarloadfile`. Modul `oarloadfile` nasleđuje `basesrc` klasu `Gststreamer` radnog okvira i proširuje je funkcionalnostima koje su omogućile `Gststreamer` radnom okviru da učitava audio objekte kao i njihove meta podatke koristeći XML datoteke kao konfiguracione datoteke koje sadrže lokacije i imena ovih datoteka na disku. Parametri koji se mogu proslediti `oarloadfile` modulu su:

- *Location* – relativna ili apsolutna putanja do XML konfiguracione datoteke.
- *frameSize* – (32 – 4096) – Veličina jednog okvira meta podataka predstavljena u odabircima.

Logika pomoćnog modula je razdeljena na tri funkcije:

- `gst_oarloadfile_init`
- `gst_oarloadfile_start`
- `gst_oarloadfile_fill`

4.2.1 Funkcija `gst_oarloadfile_start`

Otvara XML datoteku sa listom ulaznih datoteka koje predstavljaju audio objekte pozivom funkcije `oari_input_from_file`. Funkcija `oari_input_from_file` prvo iščitava XML datoteku a zatim učitava jedan po jedan objekat na osnovu liste objekata iz XML datoteke.

Nakon prolaska kroz celu XML listu, funkcija `start` postavlja veličinu izlaznog bloka na osnovu broja kanala, broja bajtova po odbirku i veličine bloka koji modul za obradu zvuka zasnovanog na objektima obrađuje. Na kraju funkcije se poziva funkcija `gst_caps_new_simple` i postavljaju se mogućnosti prolaza na izlazu iz modula na osnovu informacija iščitani iz audio datoteka.

4.2.2 Funkcija `gst_oarloadfile_fill`

Funkcija `gst_oarloadfile_fill` vrši čitanje jednog po jednog bloka zadate veličine iz ulaznih datoteka, čitanje meta podataka i dodavanje niza meta podataka u niz podataka sa odbircima. U zavisnosti od ulazne datoteke jedan blok meta podataka može da opisuje različit broj odbiraka zvuka. Meta podaci se čitaju funkcijom `oari_get_next_evo_metadata` sve dok je

broj odbiraka koji opisuju manji od broja podataka koji se obrađuje u modulu za obradu, kada se pročita dovoljan broj meta podataka, ukoliko je broj odbiraka koji je opisan pročitanim meta podacima veći od broja odbiraka koji se mogu obraditi, pamti se višak odbiraka koji će se dodati pri sledećem pozivu funkcije `oarloadfile_fill`.

Čitanje podataka se vrši funkcijom `dlb_wave_int_read`, čita se onoliko odbiraka koliko opisuju pročitani meta podaci iz svakog objekta. Nakon što su pročitani meta podaci kao i odabiraće koje oni opisuju, meta podaci se funkcijom `gst_buffer_add_oar_meta` dodaju pomoćnom nizu podataka koji se nakon toga dodaje u niz podataka na izlazu funkcijom `gst_buffer_add_parent_buffer_meta`.

Kako se niz za čitanje odbiraka inicijalizuje za potencijalni maksimum broja objekata, on popunjava onoliko odbiraka uzastopno koliko ima objekata a ostatak do maksimuma ostavlja prazno. Da bi se izbeglo nepotrebno prebacivanje praznih odbiraka potrebno je taj niz prepakovati u manji niz koji nema praznih polja.

Niz pročitanih odbiraka	Izlazni niz odbiraka
Kanal 1 Odbirak 1	Kanal 1 Odbirak 1
Kanal 2 Odbirak 1	Kanal 2 Odbirak 1
Prazno	Kanal 1 Odbirak 2
Prazno	Kanal 2 Odbirak 2
Kanal 1 Odbirak 2	Kanal 1 Odbirak 3
Kanal 2 Odbirak 2	Kanal 2 Odbirak 3
Prazno	Kanal 1 Odbirak 4

Tabela 4 - Niz pročitanih podataka i izlazni niz podataka za dva kanala i maksimalan broj objekata četiri

5. Testiranje 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. Postoji više načina ispitivanja rezultata, metode korišćene prilikom testiranja ispravnosti u ovom radu su:

1. Slušni testovi – subjektivna metoda provere tačnosti
2. Spektralna analiza – Poređenje signala u spektralnom domenu
3. Bit identični testovi – poređenje izlaza implementacije na ciljnoj platformi sa referentnim izlazima

Za izvršavanje ovih testova korišćen je *Raspberry Pi 2, model B v1.2*, koji u sebi ima procesor BCM2837 sa 4 Cortex A53 procesorska jezgra, koja rade na taktu 900MHz.



Slika 12 - Izgled ispinog okurženja

Za potrebe slušnih testova korišćena je analogna izlazna sprega na *Raspberry Pi* računaru, kojoj se pristupalo kroz ALSA zvučnu spregu. Primer poziva *Gstreamer* okvira koji koristi datoteku kao izvor, a ALSA sloj kao izlaznu spregu, dat je u poglavlju 4.1. Za potrebe spektralne analize i sprovođenje testova identičnosti na nivou bita, izlazni tok se upisivao u datoteku. Nakon toga vršeno je poređenje izlazne datoteke sa izlazom iz referentnog modula. Primer komandne linije za pravljenje protočne strukture korišćene za upisivanje izlaznog toka u datoteku je:

```
gst-launch-1.0 oarloadfile ! oar frameSize=1536 speakerConf=LR !
audioconvert ! audio/x-raw,format=S24_32LE ! audioconvert ! wavenc
! filesink
```

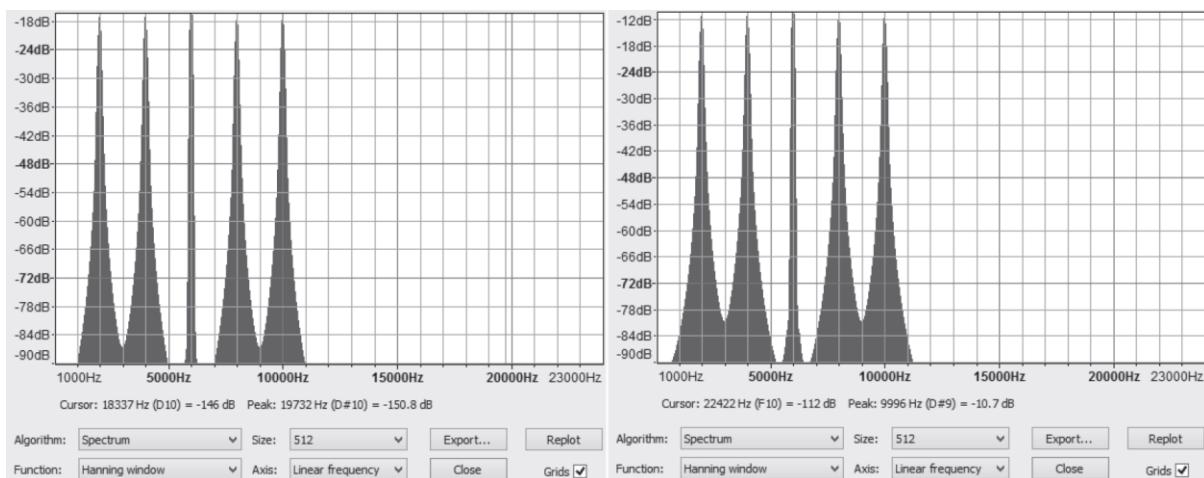
5.1 Slušni testovi

Slušni testovi spadaju u kategoriju subjektivnih metoda ispitivanja i formalnom smislu zahtevaju veliki broj obučenih slušalaca u kontrolisanom okruženju. Čujna razlika između originalnog signala i signala koji se ispituje, posmatra se kao oštećenje i ocenjuje se u skladu sa skalom **ITU-R BS.1284** standarda.[14] U slučaju ovog rada slušni testovi nisu izvršavani od strane obučenih slušalaca, niti su izvršavani u kontrolisanom okruženju i stoga oni nisu bili pouzdan način utvrđivanja ispravnosti implementacije, nego su se koristili radi lakšeg pronalaženja greški u implementaciji.

Zbog ograničenja *Raspberry Pi* računara i njegove izlazne analogne audio sprege, koja podržava maksimalno 2 izlazna kanala, bilo je moguće u potpunosti ispitati modul samo kada radi sa konfiguracijom sa 2 ili manje zvučnika. Kako bi se mogao poslušati i sadržaj ostalih izlaznih kanala kod konfiguracija sa većim brojem zvučnika, korišćena je mogućnost *audioconvert* modula za promenu redosleda izlaznih kanala (*mix-matrix*). Na taj način bilo je moguće odabrati koja dva kanala od ponuđenih na izlazu iz OAR modula će biti reprodukovana kroz izlaznu audio spregu.

5.2 Poređenje signala u spektralnom domenu

Ispitivanje vektora u spektralnom domenu se primenjivalo u slučajevima u kojim smo pretpostavili da razlike u izlaznim vektorima na nivou bita nisu posledica greške, već smanjene tačnosti. Spektralna analiza je vršena upotrebotom alata *Audacity*.



Slika 13 - Prikaz spektralne karakteristike referentnog i izlaznog signala koristeći alat Audacity

5.3 Testovi identičnosti na nivou bita

Testovi identičnosti na nivou bita predstavljaju najprecizniju i najbržu metodu testiranja ispravnosti implementacije. Testovi identičnosti na nivou bita podrazumevaju poređenje izlaza referentnog modula sa izlazom implementiranog modula na nivou bita. Kako su i referentni modul i implementacija pokretani na ciljnoj platformi, očekivani rezultat je potpuna identičnost na nivou bita, što je i dobijeno. Korišćeno je više različitih ulaznih signala kako bi se utvrdila funkcionalnost implementiranog rešenja za različite slučajeve upotrebe. Da bi se izbeglo ručno pokretanje svih testova svaki put kada se ukaže potreba za time, napravljena je skripta u *Bash* komandnom jeziku koja pokreće izvršavanje bit identičnih testova za sve izlazne signale i zapisuje rezultate u tekstualnu datoteku radi kasnije analize.

```

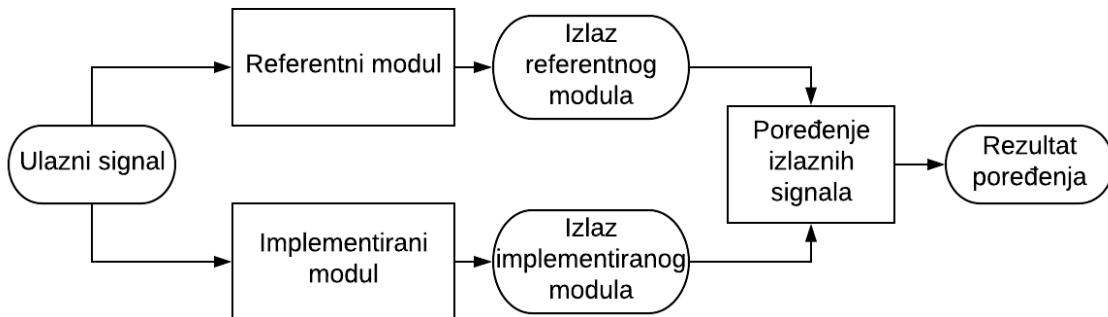
18 Calculating differences based on 24 bit accuracy:
19
20
21 overall: 0          0.00%          (0/0)
22
23
24 overall_Ch0: 0  0.00%          (0/0)
25
26
27 overall_Ch1: 0  0.00%          (0/0)
28
29 no errors

```

Slika 14 - Primer rezultata jednog od testiranja

Odrađeno je 11 testova za različite slučajeve testiranja. Testovi su rađeni tako što je svaki od ulaznih signala propušten kroz referentni modul i kroz implementirani modul, nakon obrade poredili su se izlazi. Proces testiranja ulaznog signala za jedan slučaj testiranja prikazan je na slici 13. Za utvrđivanje ispravnosti implementacije korišćeni su testovi identičnosti na nivou

bita. Za sve izvršene testove, razlika između dobijene i referentne datoteke je bila 0 bita, odnosno uspešnost izvršenja testova je **100%**.



Slika 15 - Process testiranja modula za jedan ulazni signal

5.4 Potrošnja sistemskih resursa

Ulagni tok	Broj ulaznih objekata	Broj kanala na izlazu	Vreme trajanja ulaznog toka [s]	Vreme izvršavanja testova [s]	Približan utrošak procesorskog vremena [mips]
Elevation_f40	1	8	2	1,071	344,64025
Elevation_f1536	1	8	2	1,03	331,96941
General_listening_5_1_2_f40	11	2	30	23,141	176,27245
General_listening_5_1_2_f40	11	8	30	23,921	172,59549
General_listening_5_1_2_f1536	11	8	30	23,479	178,68850
Inactive_objects_f40	5	6	2	1,24	312,56278
Inactive_objects_f1536	5	6	2	1,23	281,50090
Metadata_timing_48_f40	3	8	1,008	0,61	383,98645
Reserved_oa_element_f1536	1	8	2	1,009	284,93670

Tabela 5 - Utrošak procesorskog vremena i trajanje obrade za različite ulazne tokove

Nakon završetka modula za renderovanje audio objekata urađena je procena utrošenog procesorskog vremena za obradu različitih ulaznih tokova. Procesorsko vreme potrebno za obradu u OAR modulu se računalo po sledećoj funkciji:

$$MIPS = \frac{\text{broj_ciklusa} * \frac{OAR}{100}}{1000000 * vr_trajanja}$$

Gde *broj_ciklusa* predstavlja ukupan broj ciklusa koji je potreban za izvršavanje protočne strukture za jedan ulazni tok. *OAR* procenat utrošenog vremena potreban samo za OAR modul izražen u procentima u odnosu na ukupnu potrošnju protočne strukture pokrenute koristeći *gst-launch* alat, iz tog razloga se deli sa 100. Vrednost utrošenog vremena izražena u procentima se dobija korišćenjem programa za profilisanje pod nazivom *perf* ili *linux-perf* [15]. *vr_trajanja* je ukupno vreme izvršavanja protočne strukture za jedan ulazni tok.

Kako su se svi pokrenuti testovi završili uspešno, potvrđena je funkcionalnost implementacije na ciljnoj platformi. Računanjem utroška procesorskog vremena potvrđeno je da realizovano rešenje može da se izvršava u realnom vremenu. Iz tabele 5 se može videti da je prilikom testiranja, najsporije se izvršavao primer *Metadata_timing_48_f40* sa 383.98645MIPS, što predstavlja ukupno opterećenje jednog procesorskog jezgra od 42,66%. Odatle vidimo da nam je preostalo slobodnog procesorskog vremena koje može da se iskoristi za izvršavanje još nekog elementa iz lanca obrade, na primer dekoderskog modula ili nekog elementa završne obrade.

6. Zaključak

U okviru ovog rada je implementiran modul za renderovanje audio objekata na procesoru *Cortex A53 ARM* kompanije koristeći *Gstreamer* kao radni okvir za implementaciju. Polazna tačka je bio referentni kod napisan u jeziku C. Ovaj kod je bilo potrebno prilagoditi *Gstreamer* radnom okviru. U tom cilju napravljen je *Gstreamer* modul koji sadrži funkcionalnosti potrebne za rad OAR modula.

Za potrebe testiranja ispravnosti implementiranog modula napravljen je pomoćni modul nazvan *oarloadfile*. Modul *oarloadfile* sa funkcionalnostima koje su omogućile radnom okviru *Gstreamer* da učitava audio objekte kao i njihove meta podatke koristeći XML datoteke kao konfiguracione datoteke koje sadrže lokacije i imena ovih datoteka na disku.

Odrađeni su testovi opisani u prethodnom poglavlju. Najpreciznija metoda testiranja ispravnosti implementiranog modula bili su bit identični testovi. Očekivani rezultat ovih testova bila je potpuno poklapanje signala dobijenih na izlazu referentnog i implementiranog modula, što je i ispunjeno.

Krajnja ideja iza ovog rada jeste omogućavanje rada sa audio objektima u realnom vremenu na uređajima zasnovanim na ARM porodici jezgara. Na kraju je izvršeno merenje potrošnje procesorskog vremena čime je potvrđeno da se modul implementiran u radnom okviru *Gstreamer* koji pruža mogućnost izvršavanja implementacije u realnom vremenu može izvršavati u realnom vremenu. Dalja proširenja rada koja će biti urađena jeste uvezivanje realizovanog modula sa dekoderskim modulima kako bi se kompletirao lanac obrade jedne audio tehnologije zasnovane na audio objektima. S obzirom da je audio spregna na *Raspberry PI* računaru ograničena na 2 analogna izlazna kanala (odnosno 6 kanala preko HDMI sprege), neophodno je rešenje dodatno verifikovati koristeći neku platformu koja podržava složeniju izlaznu spregu za audio signale jedan

7. Literatura

- [1] Haibin Zhang, Hui Li, Dan Wu, Husheng Yuan, Tao Sun, Peng Yi, Hongchao Hu i Bingqiang Wang. *The design and implementation of an embedded high definition player.* The 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapur, 2010.
- [2] Hai Wang, Fei Hao, Chunsheng Zhu, Joel J. P. C. Rodrigues, and Laurence T. Yang. *An Android Multimedia Framework Based On Gstreamer.* International Conference on Green Communications and Networking. GreeNets, Čongking, Kina, 2011.
- [3] Gupta Suyash, Purshotam Chauhan, Indona Vinita Barua, Luv Shorey, and Bhawna Ahuja, *BlueARMStreamer: A Real Time Streaming Approach.* International Conference on Computational Intelligence and Communication Networks, Gvalor, Indija 2011.
- [4] Kevin, Ananta Kurniawan, Darmawan Utomo, i Saptadi Nugroho *Direction Control System on a Carrier Robot Using Fuzzy Logic Controller.* International Conference on Soft Computing, Intelligence Systems, and Information Technology (ICSIIT), Bali, Indonezija, 2015.
- [5] Potard, Guillaume, *3D-audio object oriented coding*, doktorska disertacija, Fakultet za elektrotehniku, računarstvo i telekomunikacije, Univerzitet u Volongongu, Australija, 2006.
- [6] *Multichannel sound technology in home and broadcasting applications* , International Telecommunication Union, Report ITU-R BS.2159-4 (05 /2012), Ženeva, Švajcarska 2012
- [7] Brian Claypool, Wilfred, Van Baelen, Bert Van Daele, *Auro 11.1 versus object-based sound in 3 D: All aspects compared*, 2012

- [8] *Gstreamer – open source multimedia framework*, <https://gstreamer.freedesktop.org/documentation/> [pristupljeno: Jul 2018.]
- [9] *Gobject reference manual*, <https://developer.gnome.org/gobject/stable> [pristupljeno: Jul 2018.]
- [10] Vojin G. Oklobdzija, *Reduced Instruction Set Computers*, Berkli, Univerzitet Kalifornije, 1999.
- [11] *Cortex-A7 MPCore – technical refference manual*, ARM, 2012, dostupno na http://infocenter.arm.com/help/topic/com.arm.doc.ddi0464d/DDI0464D_cortex_a7_mpcore_r0p3_trm.pdf [pristupljeno: Jul 2018.]
- [12] *ARM NEON technology overview*, <https://developer.arm.com/technologies/neon> [pristupljeno: Jul 2018.]
- [13] *Introducing NEON development article*, http://infocenter.arm.com/help/topic/com.arm.doc.dht0002a/DHT0002A_introducing_neon.pdf [pristupljeno: Jul 2018.]
- [14] Jelena Kovačević, Dejan Bokan, *Arhitekture i algoritmi digitalnih signal procesora I - Zbirka zadataka i laboratorijski priručnik*, FTN, Novi Sad, 2016
- [15] *perf: Linux profiling with performance counters*, <https://perf.wiki.kernel.org/>, [pristupljeno: Jul 2018.]