УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Стефан Станивук

Једно решење Г.722 широкопојасног аудио кодера на наменској платформи

МАСТЕР РАД

Нови Сад, 2014

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**:		Стефан Станивук	
Ментор, **МН**:		др Јелена Ковачевић	
Наслов рада, **НР**:		Једно решење Г.722 широкопојасног аудио кодера на наменској платформи	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2014	
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)		7/33/9/2/15/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		Аудио систем, Г.722 кодек, Хамингов код, контрола грешке унапред	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У овом раду је приказано једно решење Г.722 широкопојасног аудио кодера на наменској *Ezairo 7100* платформи. За пренос аудио података се користи бежична веза. Проблем бежичне комуникације представљају електромагнетне сметње које утичу на исправност преношених података. Наведени проблем је решен реализовањем алгоритама за контролу грешке унапред: Хамингов код у симбиози са преплетањем / расплетањем података.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	др Илија Башичевић	
	Члан:	др Растислав Струхарик	Потпис ментора
	Члан, ментор:	др Јелена Ковачевић	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

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**:		Stefan Stanivuk	
Mentor, **MN**:		Jelena Kovacevic, PhD	
Title, **TI**:		Implementation of G.722 codec on embedded 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**:		2014	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/33/9/2/15/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Audio system, G.722 codec, Hamming code, forward error correction - FEC	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper describes one solution of G.722 wideband audio codec on embedded platform Ezairo 7100. For transmitting audio data, wireless technology is used. One of a major problem of wireless technology is an electromagnetic interferences which can corrupt transmitted data. To protect against that problem, FEC (Forward Error Correction) technique was developed: Hamming code with interleaving / deinterleaving.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Ilija Basicevic, PhD	
	Member:	Rastislav Struharik, PhD	Menthor's sign
	Member, Mentor:	Jelena Kovacevic, PhD	

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Проф. др Никола Јорговановић

Студент:	Стефан Станивук	Број индекса:	Е2 10/2013
Област:	Архитектуре и алгоритми ДСП 2		
Ментор:	Др Јелена Ковачевић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ МАСТЕР РАДА:

Једно решење Г.722 широкопојасног аудио кодера на наменској платформи

ТЕКСТ ЗАДАТКА:

Потребно је реализовати аудио систем на наменској платформи користећи Г.722 аудио кодек. За пренос аудио података између предајника и пријемника треба користити бежичну везу. Такође треба реализовати одређене алгоритме који ће имати за циљ смањење утицаја електромагнетних сметњи на аудио ток преношен бежичном везом.

Руководилац студијског програма:	Ментор рада:

Zahvalnost

Zahvaljujem se mentoru dr Jeleni Kovačević na ukazanom poverenju i utrošenom vremenu tokom izrade ovog rada.

Takođe se zahvaljujem, sada već kolegama, Đorđu Petroviću, Momčilu Kruniću i Nenadu Četiću na stručnoj pomoći i savetima kao i na odličnoj radnoj atmosferi za koju zasluge imaju i ostale kolege iz Cirrus Logic / Starkey tima.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

FEC - *Forward Error Correction*, kontrola greške unapred

SoC - *System on Chip*, integrisano kolo u koje su ugrađene sve komponente jednog računara ili nekog drugog elektronskog sistema

DSP - *Digital Signal Processing / Digital Signal Processor*, digitalna obrada signala / digitalni signal procesor

FIFO - *First In First Out*, memorijska struktura koja radi na principu reda – prvi koji dođe će biti i prvi uslužen

IOC - *Input / Output Controller*, ulazno / izlazni kontroler

IIR - *Infinite Impulse Response*, filter sa beskonačnim impulsnim odzivom

FIR - *Finite Impulse Response*, filter sa konačnim impulsnim odzivom

GPIO - *General Purpose Input / Output*, izvodi na integrisanim kolima koji se po potrebi mogu podesiti da budu ulazni, izlazni ili ulazno / izlazni

SPI - *Serial Peripheral Interface*, serijski sprežni sistem za periferije

UART - *Universal Asynchronous Receiver-Transmitter*, univerzalni asinhroni prijemnik i predajnik: serijski prenos paralelnih podataka

LSAD - *Low-Speed Analog to Digital*, analogno-digitalni konvertor

I2C - *Inter-Integrated Circuit*, serijska komunikaciona sprega

PCM - *Pulse-Code Modulation*, impulsna kodna modulacija

CVSD - *Continuously Variable Slope Delta modulation*, tehnika kodiranja signala govora

DMA - *Direct Memory Access*, direktan pristup memoriji

CRC - *Cyclic Redundancy Check*, tehnika za otkrivanje grešaka

ECC - *Error Correction Coding*, tehnika za ispravljanje grešaka

ITU-T - *International Telecommunication Union - Telecommunication Standardization Sector*, međunarodna unija za telekomunikacije - odsek za standardizaciju telekomunikacionih sistema

SB-ADPCM - *Sub-Band Adaptive Differential Pulse-Code Modulation*, podopsežna adaptivna diferencijalna impulsno-kodna modulacija

LSb - *Least Significant bit*, bit najmanje važnosti

RCA - *Radio Corporation of America*, tip priključka za prenos video i audio signala

LUT - *Look-Up Table*, niz koji zamenjuje izračunavanje sa jednostavnijim nizom indeksirane operacije

MIPS - *Million Instructions Per Second*, milion instrukcija po sekundi

USB - *Universal Serial Bus*, univerzalna serijska magistrala

API - *Application Programming Interface*, programska sprega

ISR - *Interrupt Service Routine*, rukovalac prekida

1. Uvod

U ovom radu je prikazano jedno rešenje G.722 širokopojasnog audio kodera (engl. *encoder*) sa kontrolom grešaka unapred (engl. *Forward Error Correction – FEC*) na namenskoj (engl. *embedded*) platformi koje će služiti za bežičan prenos audio podataka.

Kodiranje digitalnog audio signala se koristi radi omogućavanja efikasnijeg prenosa i čuvanja tog signala. To znači da audio koder predstavlja *mašinu* za sažimanje (engl. *compression*) digitalnih audio podataka, koja da bi to postigla obavlja nad njima određene DSP (engl. *Digital Signal Processing*) operacije. Tako dobijen audio bitski tok se može razumeti samo ako se nad njim izvrše suprotne DSP operacije koje obavlja odgovarajući audio dekoder.[1]

G.722 audio kodek je širokopojasni kodek koji je namenjen prvenstveno za kvalitetno kodiranje signala govora. Zbog svog šireg opsega daje veći kvalitet audio kodiranja u odnosu na starije standarde kao što je G.711 ili CVSD. Primenjuje se u Internet telefoniji (engl. *Voice Over Internet Protocol – VoIP*) kao i u raznim slučajevima bežičnog prenosa audio signala.

Zadatak ovog rada zahteva sledeće radnje: projektovanje i realizaciju čitavog audio sistema od audio izvora (npr. TV), G.722 audio kodera, bežičnog predajnika (engl. *wireless transmitter*), bežičnog prijemnika (engl. *wireless receiver*), G.722 audio dekodera (engl. *decoder*) i na kraju audio izlaza – zvučnika koji može predstavljati bežične slušalice ili slušni aparat. Bežični prenos je po svojoj prirodi podležan raznim elektromagnetnim smetnjama (engl. *electromagnetic interference*) koje dovode do grešaka u audio bitskom toku (engl. *audio bit stream*) pa je jedan od zahteva i realizacija algoritama za povećanje robusnosti audio sistema na moguće greške u audio bitskom toku. U ovom radu će biti predstavljeno Hamingovo kodiranje / dekodiranje zajedno sa prepletanjem / raspletanjem podataka čija simbioza čini jednu od osnovnih FEC

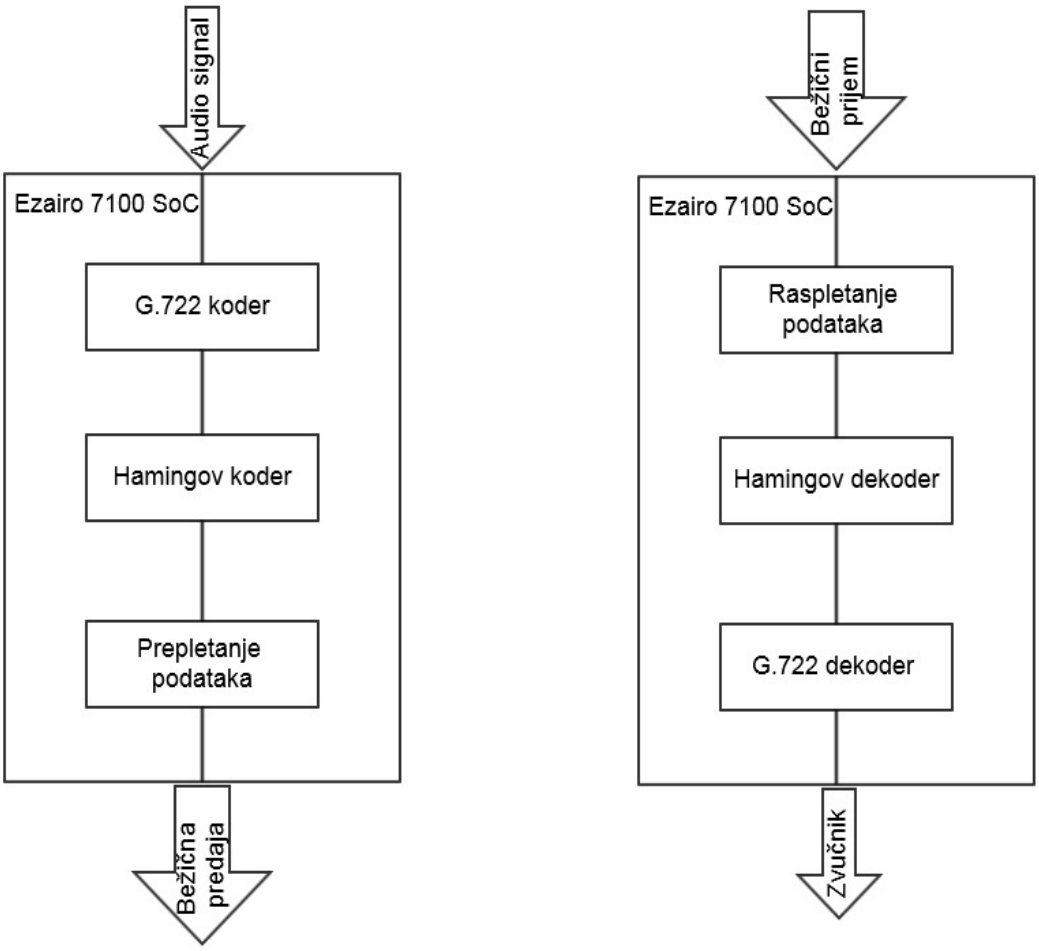
tehnika. Cilj ovog zadatka je uspešna realizacija sistema za efikasno bežično prenošenje audio bitskog toka, slika 1.1.

Slika 1.1 Prikaz audio sistema koji treba da bude realizovan u ovom radu

Rad je organizovan u sedam poglavlja. U drugom poglavlju su opisane teorijske osnove potrebne za razumevanje ovog rada: opis G.722 kodeka, opis tehnika za kontrolisanje / ispravljanje grešaka, opis korišćene namenske platforme. U trećem poglavlju je predstavljen koncept rešavanja problema navedenih u zahtevima rada. U četvrtom poglavlju su izneti detalji programske realizacije na ciljnoj platformi. U petom poglavlju opisan je postupak ispitivanja i verifikacije datog rešenja. U šestom poglavlju je ukratko opisano šta je urađeno i šta se može dodatno unaprediti. Sedmo, poslednje poglavlje, sadrži spisak korišćene literature.

2. Teorijske osnove

U ovom poglavlju su izložene teorijske osnove gradivnih elemenata ovog rada. Na slici 2.1 je prikazan sistem koji će služiti za bežični prenos kodiranih audio podataka i kojeg čine sledeći gradivni elementi: Ezairo 7100 SoC (engl. *System on Chip*), G.722 kodek, Hamingov kodek i tehnika prepletanja / raspletanja toka podataka (engl. *interleaving / deinterleaving*) – deo FEC.



Slika 2.1 Gradivni elementi: na strani predajnika (levo) i na strani prijemnika (desno)

2.1 G.722 audio kodek

G.722 je širokopojasni audio kodek sa gubicima (engl. *lossy*) standardizovan od strane ITU-T (engl. *International Telecommunication Union - Telecommunication Standardization Sector*). Standard opisuje sistem audio kodiranja (od 50 do 7000 Hz) koji se može koristiti u raznim govornim (engl. *speech*) aplikacijama u kojima se traži visok kvalitet. Sistem koristi podopsežnu adaptivnu diferencijalnu impulsno-kodnu modulaciju (engl. *Sub-Band Adaptive Differential Pulse-Code Modulation – SB-ADPCM*) bitske brzine 64 kbit/s. Zbog toga se sistem često naziva i sistem audio kodiranja bitske brzine 64 kbit/s (7 kHz). U SB-ADPCM tehnici, opseg učestanosti se deli na dva podopsega (viši podopseg i niži podopseg) gde se u svakom podopsegu signali kodiraju pomoću ADPCM tehnike. Sistem podržava tri osnovna načina rada koji zavise od bitske brzine korišćene u 7kHz audio kodiranju: 64 kbit/s, 56 kbit/s i 48 kbit/s. Poslednja dva načina rada omogućuju korišćenje pomoćnih tokova podataka bitskih brzina 8 kbit/s i 16 kbit/s respektivno, i to korišćenjem bita iz nižeg podopsega.[3]

G.722 standard je superiorniji ali i složeniji za realizaciju u odnosu na stariji CVSD standard koji se prvenstveno koristi za kodiranje govornog signala. Tabela 2.1 daje uporedni prikaz ova dva standarda:

	G.722	**CVSD**
Opseg učestanosti	50 Hz – 7 kHz	0 Hz – 4 kHz
Odnos Signal – Totalna distorzija	50 dB – 60 dB	30 dB – 35 dB
Učestanost odabiranja	16 kHz	64 kHz
Broj bita po odbirku	8	1
Bitska brzina	64 kbit/s	64 kbit/s
Broj podopsega	2	1
Broj bita po podopsegu	6 + 2 (viši + niži podopseg)	1

Tabela 2.1 Uporedni prikaz G.722 i CVSD audio standarda

2.2 FEC tehnika

U telekomunikacijama i teoriji kodiranja FEC predstavlja tehniku korišćenu za kontrolisanje grešaka u prenosu podataka preko nepouzdanih komunikacionih kanala. Osnova ove tehnike se bazira na tome da pošiljalac kodira svoju poruku na redundantan način, dodavajući dodatne podatke pre slanja, koristeći neki poznati ECC (engl. *Error Correction Coding*) algoritam.[4] Ti dodatni podaci omogućuju primaocu kako da otkrije određen broj grešaka u primljenoj poruci tako i da ih ispravi, bez potrebe za ponovnim slanjem iste poruke. To znači da FEC daje mogućnost primaocu da ispravi greške bez korišćenja komunikacionog kanala u suprotnom smeru preko kojeg bi primalac tražio od pošiljaoca da ponovi isporuku već poslate poruke. FEC tehnika za manu ima dodatno kašnjenje koje sama FEC obrada unosi kao i zahtev za većom propusnom (engl. *bandwidth*) moći komunikacionog kanala zbog unetih dodatnih podataka.[5]

Broj grešaka koji se može otkriti koristeći FEC tehniku zavisi od primenjenog ECC algoritma tj. od veličine dodatnih podataka.

Pionir ECC kodiranja je američki matematičar Ričard Haming (engl. *Richard Hamming*) koji je 1950. osmislio prvu ECC tehniku: Hamingov kod (7, 4).[6]

2.2.1 Hamingov kod

U telekomunikacijama Hamingov kod predstavlja sastavni deo FEC tehnike za otkrivanje i ispravljanje grešaka u bitskom toku podataka. Postoji više verzija Hamingovih kodova ali se svi baziraju na algoritmu koji je predstavljen prvi put u Hamingovom kodu (7, 4).

Hamingov kod (7, 4) kodira četvorobitnu reč podataka u sedmobitnu reč dodajući tri bita parnosti (engl. *parity bits / check bits*). Hamingov kod može da otkrije i ispravi jednobitne greške ili da otkrije dvobitne greške ali ne i da ih ispravi [7], što znači da je Hamingov kod efikasan tamo gde se ne javljaju sekvence uzastopnih bitskih grešaka (engl. *burst errors*).[8]

Cilj ove tehnike je da se dovoljan broj bitova parnosti uredi na takav način da pojava različitih pogrešnih bitova prouzrokuje različite greške, a to onda znači da se pogrešni bitovi mogu identifikovati. To za Hamingov kod (7, 4) koji proizvodi sedmobitnu reč, koja može imati sedam različitih jednobitnih grešaka, znači da tri bita parnosti ($2^3 = 8$) koji se u tom kodu koriste mogu odrediti ne samo da li je došlo do greške već i da identifikuju bit koj je prouzrokovao grešku.

Opšti algoritam se može opisati na sledeći način:

1. Sve pozicije bitova (pozicije kreću od broja 1) koje su stepen broja dva predstavljaju bitove parnosti (1, 2, 4, 8, 16, …).
2. Sve ostale pozicije bitova (3, 5, 6, 7, 9, …) predstavljaju bitove podataka.
3. Svaki bit podataka se nalazi u oblasti kontrole dva ili više bitova parnosti u zavisnosti od bitske pozicije:
    a. Bit parnosti 1 kontroliše sve bitske pozicije koje imaju postavljen LSb (engl. *Least Significant bit*) bit: 1, 3, 5, 7, 9, …
    b. Bit parnosti 2 kontroliše sve bitske pozicije koje imaju postavljen LSb + 1 bit: 2, 3, 6, 7, 10, …
    c. Bit parnosti 4 kontroliše sve bitske pozicije koje imaju postavljen LSb + 2 bit: 4, 5, 6, 7, 12, 13, 14, 15, …
    d. U opštem slučaju, svaki bit parnosti kontroliše one bite gde bitska operacija AND između pozicije bita parnosti kao jednog operanda i pozicije posmatranog bita kao drugog operanda daje vrednost različitu od nule.

Gore navedena pravila se mogu tabelarno prikazati na sledeći način:

Bitske pozicije		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Kodirani tok bita		**P1**	**P2**	**D1**	**P4**	**D2**	**D3**	**D4**	**P8**	**D5**	
Oblast kontrole bitova parnosti	**P1**	X		X		X		X		X	**…**
	P2		X	X			X	X			
	P4				X	X	X	X			
	P8								X	X	

Tabela 2.2 Tabelarni prikaz Hamingovog koda

- Px – označava bit parnosti koji se nalazi na bitskoj poziciji x
- Dx – označava bit podataka koji se nalazi na bitskoj poziciji x

U tabeli 2.2 je prikazana bitska reč od samo devet kodiranih bita, od kojih su četiri bitovi parnosti a pet bitovi podataka, zbog ograničenosti prostora, inače celo pravilo se može primenjivati do beskonačnosti.

Da bi se proverilo da li postoji greška u primljenoj bitskoj reči, potrebno je samo proveriti sve bitove parnosti. Ako su svi bitovi parnosti ispravni, greška ne postoji. U suprotnom, suma

bitskih pozicija neispravnih bitova parnosti daje poziciju neispravnog bita. Ako je samo jedan bit parnosti neispravan, onda je samo taj bit parnosti neispravan.[9]

Kako je već rečeno, postoji više verzija Hamingovg koda u zavisnoti od broja korišćenih bitova parnosti. Ako se koristi *n* bitova parnosti, dati Hamingov kod može da kontroliše kodirane bitove bitskih pozicija od 1 do $2^n - 1$. Kad se oduzmu bitovi parnosti iz kodirane reči, ostaje $2^n - n - 1$ bitova koji se mogu koristiti za podatke.

Da bi se omogućilo otkrivanje dvobitnih grešaka (ne i ispravljanje istih) uveden je sveobuhvatni bit parnosti. Uloga ovog bita parnosti je da ukaže da li je broj grešaka u kodiranoj reči paran ili neparan. Ako gore opisani Hamingov kod otkrije da postoji greška u primljenoj kodiranoj reči, a sveobuhvatni bit parnosti ukaže da je prisutan paran broj grešaka, onda je u pitanju dvobitna greška.

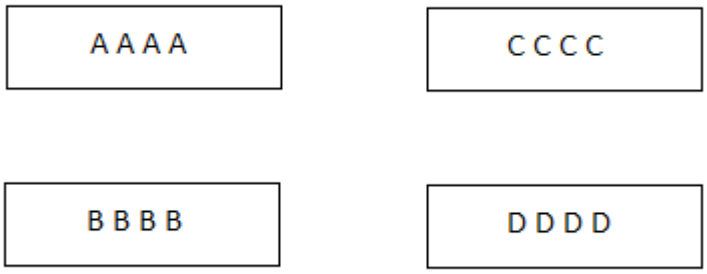
2.2.2 Prepletanje/raspletanje podataka

Kao što je već rečeno, Hamingov kod nije efikasan tamo gde se javljaju sekvence uzastopnih bitskih grešaka. Upravo iz tog razloga je uvedena tehnika prepletanja / raspletanja podataka kao deo FEC sistema koja uz Hamingov kod daje mnogo bolju kontrolu nad greškama.[8]

Cilj tehnike je da bitovi kodiranih reči međusobno zamene mesta na osnovu poznatog algoritma da bi se na strani prijemnika mogla primeniti suprotna operacija. Tim mešanjem bitova različitih kodiranih reči se povećava robusnost FEC sistema na sekvence uzastopnih bitskih grešaka, na način da ako se u jednoj kodiranoj reči, koja sadrži bitove premeštene iz drugih kodiranih reči, tokom prenosa javi više bitskih grešaka koje se nalaze na uzastopnim bitskim pozicijama, posle raspletanja na strani prijemnika te greške će biti raširene na više kodiranih reči i u većini slučajeva će postojeći Hamingov kod moći da se izbori sa tim greškama – jer su te greške iz te jedne kodirane reči širenjem na više kodiranih reči postale jednobitne greške.[4]

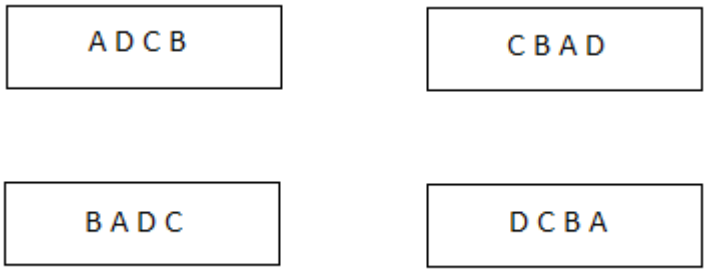
Mana ove tehnike je što prijemnik mora sačekati da prvo primi blok podataka određene veličine da bi bio uopšte u mogućnosti da primeni tehniku raspletanja što uvodi određeno kašnjenje, a prednost što ne uvodi dodatne podatke koji bi zahtevali veću propusnu moć.[5]

Na slici 2.2 su prikazane četiri četvorobitne kodirane reči nad kojima nije primenjena tehnika prepletanja:



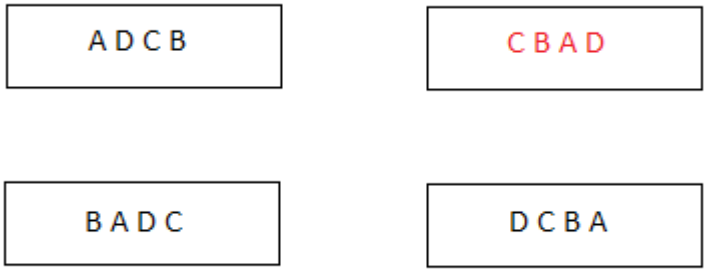
Slika 2.2 Kodirane reči bez prepletanih bitova

Sledeća slika, slika 2.3, prikazuje pomenute kodirane reči ali nad kojima je sad primenjena tehnika prepletanja kao zaštita od sekvence uzastopnih bitskih grešaka. Svaka novoformirana kodirana reč sada sadrži po jedan bit od svake kodirane reči:



Slika 2.3 Kodirane reči sa prepletanim bitovima

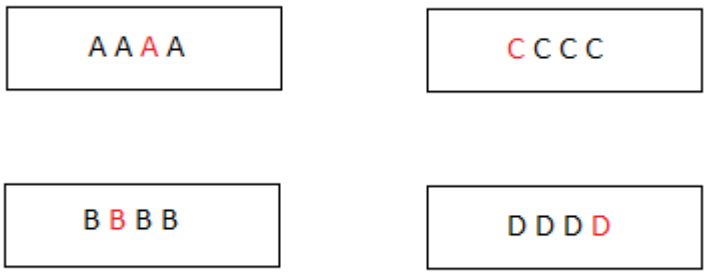
Te kodirane reči sa prepletenim bitovima se šalju prijemniku. Tokom prenosa može doći do pojave grešaka u bitskom toku. Na slici 2.4 su prikazane kodirane reči sa slike 2.3 ali sa pojavom četvorobitne greške u jednoj kodiranoj reči:



Slika 2.4 Kodirana reč sa četvorobitnom greškom (crvena boja)

Sam Hamingov kod, bez korišćenja tehnike prepletanja, ne bi mogao ni da otkrije ovakvu grešku jer kao što je već rečeno nije efikasan za sekvence uzastopnih bitskih grešaka (u ovom slučaju postoji četvorobitna greška na uzastopnim bitskim pozicijama).

Na slici 2.5 su prikazane kodirane reči nad kojima je po prijemu urađena tehnika raspletanja, gde se vidi da je sekvenca uzastopnih bitskih grešaka raspodeljena na jednobitne greške koje Hamingov kod efikasno otkriva i ispravlja:



Slika 2.5 Raspodela četvorobitne greške na jednobitne greške pomoću tehnike raspletanja

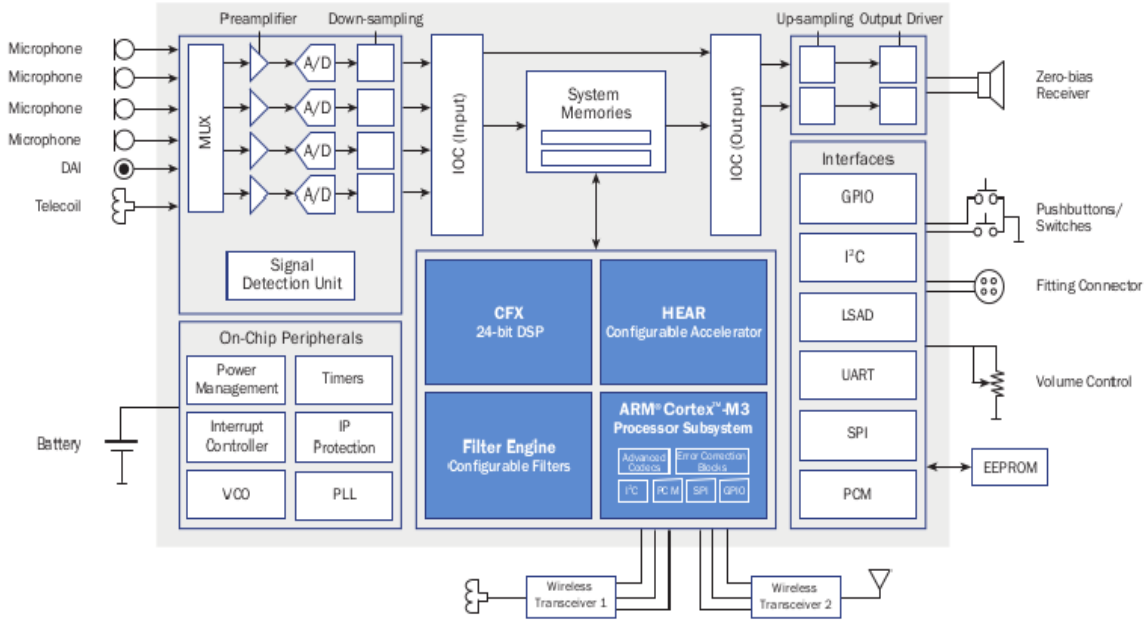
Gore je prikazan primer sa četvorobitnom greškom u bitskom toku koju Hamingov kod može efikasno rešiti, ali ako se u tom istom bitskom toku pojavi greška bitske širine veće od četiri bita, to će značiti da će se raspodelom te greške tehnikom raspletanja ta greška raširiti ne na jednobitne greške kao u prethodnom primeru nego na višebitne greške koje Hamingov kod neće moći efikasno otkriti i ispraviti. Zbog toga se uvodi dodatni stepen prepletanja ali ovaj put samih kodiranih reči koje će imati za cilj da još više raširi moguće greške u bitskom toku.

2.3 Ezairo 7100 SoC

Ezairo 7100 je proizvod kompanije *On Semiconductor* koji je projektovan isključivo za obradu audio podataka. Sistem sačinjava asimetrična višejezgarna arhitektura koja sadrži četiri procesorska jezgra: CFX digitalni signal procesor (engl. *Digital Signal Processor - DSP*), HEAR pomoćno procesorsko jezgro, ARM Cortex-M3 procesor i Filter Engine pomoćno procesorsko jezgro.

Data četiri procesorska jezgra imaju na raspolaganju kako lokalne tako i deljene memorijske module. CFX, ARM Cortex-M3 i HEAR jezgra obrađuju podatke koje dopremaju preko nekoliko FIFO (engl. *First In First Out*) struktura a kojima upravlja IOC (engl. *Input/Output Controller*) kontroler. [2]

U daljem tekstu su ukratko opisana data procesorka jezgra. Na slici 2.1 je prikazan Ezairo 7100 SoC:



Slika 2.6 Blok dijagram Ezairo 7100 sistema

2.3.1 CFX DSP

CFX DSP je primarni digitalni processor u sistemu. To je programabilni DSP opšte namene. CFX DSP ima više zadataka u sistemu: da deluje kao vodeći (engl. *master*) procesor u sistemu, da bude u sprezi sa svim periferijama sistema kao i sa svim spoljnim spregama (engl. *external interfaces*), da kontroliše ostala jezgra, da upravlja tokom podataka koji prolazi kroz sistem.

CFX DSP koristi poboljšanu Harvard (engl. *Dual-Harvard*) memorijsku arhitekturu radi obezbeđivanja posebnih memorija: jednu za program (PMEM) i dve za podatke (XMEM i YMEM). Programska memorija je širine 32 bita, dok su memorije za podatke širine 24 bita. Memorijama za podatke se moze pristupiti istovremeno dobijajući tako memoriju za podatke (XYMEM) širine 48 bita.

CFX jezgro podržava sledeće sprege: GPIO (engl. *General Purpose Input/Output*), SPI (engl. *Serial Peripheral Interface*), UART (engl. *Universal Asynchronous Receiver-Transmitter*), LSAD (engl. *Low-Speed Analog-to-Digital*).

Maksimalna radna učestanost je 10,24 MHz.[2]

2.3.2 ARM Cortex-M3

ARM Cortex-M3 je opštenamenski programabilni procesor koji se u ovom sistemu koristi kao pomoćni procesor u cilju obezbeđivanja dodatne procesorske moći. Ova dodatna procesorska moć se može iskoristiti za kontrolisanje spoljnih bežičnih primopredajnika kao i za obradu podataka dobijenih preko pomenutih primopredajnika – što i predstavlja glavnu svrhu postojanja ovog procesora u Ezairo 7100 sistemu.

ARM Cortex-M3 podržava razne sprege: I2C (engl. *Inter-Integrated Circuit*), GPIO, PCM (engl. *Pulse-Code Modulation*), SPI. Takođe, za periferije sadrži sledeće blokove koji su realizovani u samoj fizičkoj arhitekturi:

- Blok za audio kodovanje: G.722 i CVSD (engl. *Continuously Variable Slope Delta modulation*)
- Blok za DMA (engl. *Direct Memory Access*) kontroler
- Blok za CRC (engl. *Cyclic Redundancy Check*)
- Blok za ispravljanje grešaka (engl. *Error Correction Coding – ECC*)

Memorija procesora je širine 32 bita i podeljena je na dva dela (Harvard arhitektura): memoriju za podatke i memoriju za program.

CFX DSP kontroliše ARM Cortex-M3 procesor tako što kontroliše njegov takt, stanje jezgra pri izvršavanju, memoriju ARM jezgra, digitalne ulaze / izlaze.[2]

2.3.3 HEAR

HEAR jezgro omogućava dodatnu funkcionalnost pri obradi podataka u Ezairo 7100 sistemu. HEAR jezgro je projektovano tako da omogući kako običnu obradu podataka tako i kompleksnu obradu kroz razne filter banke i time da rastereti glavno jezgro – CFX.

HEAR 24-bitno jezgro nije direktno programabilno već se podešava preko predefinisanih mikroprogramskih modula.

U HEAR je ugrađen blok koji radi sa pokretnim zarezom (engl. *floating point*) zbog potrebe za radom sa većim dinamičkim opsezima.

Kao i ARM Cortex-M3, i HEAR se kontroliše preko CFX jezgra.

HEAR koristi Harvard memorijsku arhitekturu, gde je memorija podeljena na memoriju za mikroprogram i memoriju za podatke. [2]

2.3.4 Filter Engine

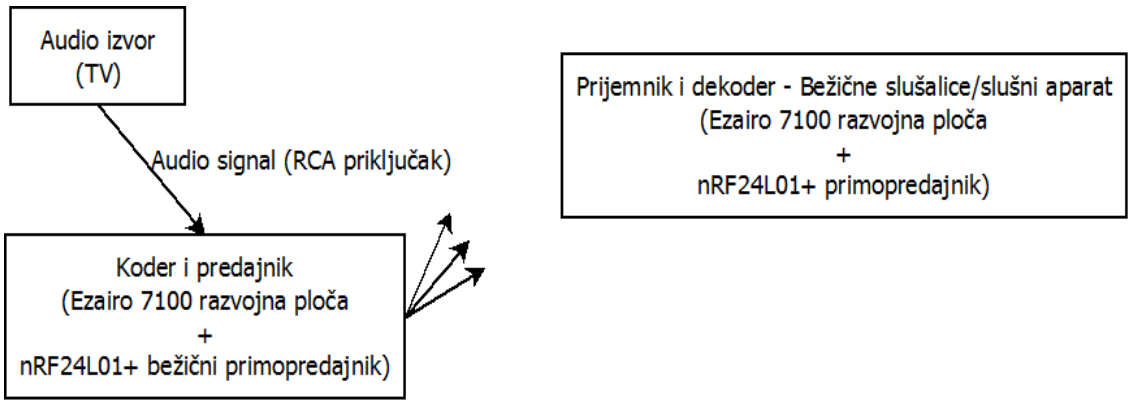
Filter Engine je programabilno specijalizovano jezgro napravljeno za korišćenje u audio obradi. Primarno je projektovan radi obavljanja filterskih operacija u vremenskom domenu, kao što je IIR (engl. *Infinite Impulse Response*) i FIR (engl. *Finite Impulse Response*) filtriranje.

Filter Engine sadrži programsku memoriju širine 32 bita i dve memorije širine 24 bita: jednu za koeficijente filtera a drugu za stanja filtera.

CFX jezgro takođe kontroliše Filter Engine jezgro.[2]

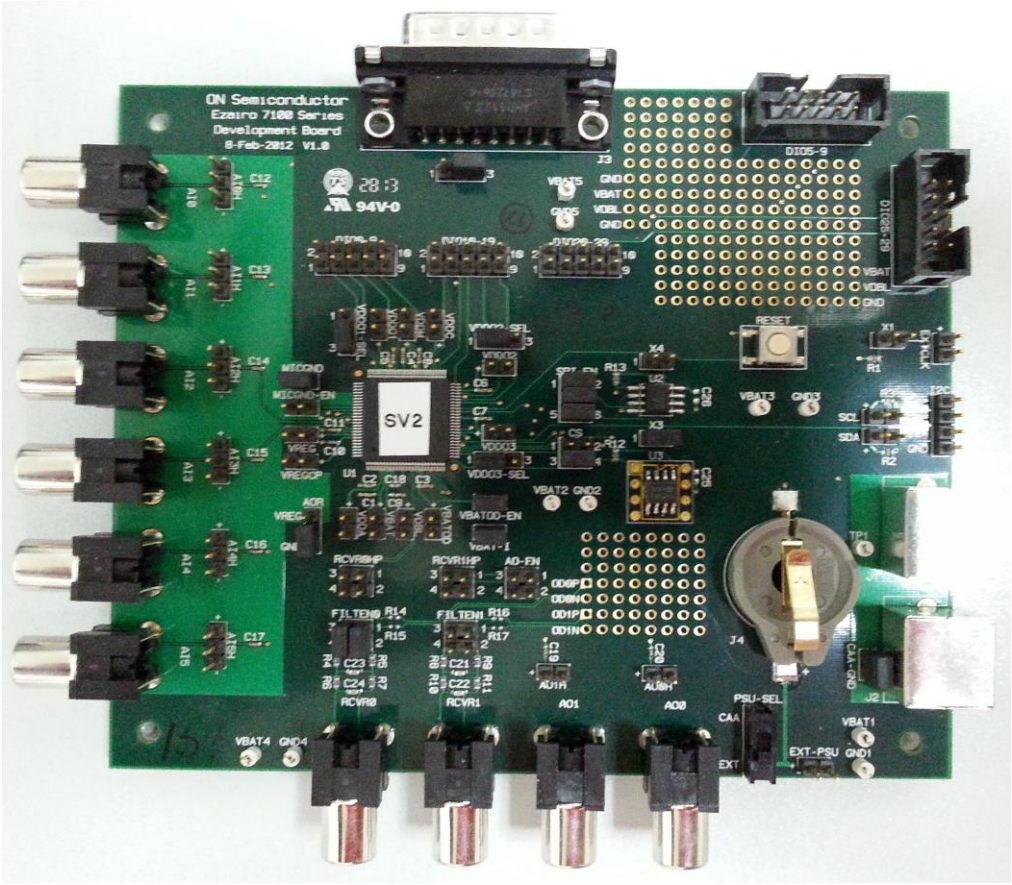
3. Koncept rešenja

Koncept audio sistema koji se treba realizovati je prikazan na slici 3.1:

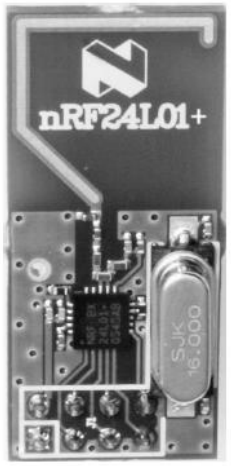


Slika 3.1 Koncept predloženog audio sistema za bežični prenos podataka

Za prijem audio signala sa TV, kodiranje tog signala G.722 koderom kao i za dodatno FEC kodiranje (koje sadrži Hamingovo kodiranje i prepletanje podataka) koristiće se Ezairo 7100 razvojna ploča, slika 3.2, koja će biti spregnuta preko SPI sprege sa 2,4 GHz bežičnim primopredajnikom nRF24L01+ REFMOD kompanije *Nordic Semiconductor*, slika 3.3, zaduženim za bežično slanje podataka.



Slika 3.2 Ezairo 7100 razvojna ploča



Slika 3.3 nRF24L01+ bežični primopredajnik

Bežične slušalice ili slušni aparati će biti predstavljeni takođe preko jedne Ezairo 7100 razvojne ploče, čiji će biti zadatak raspletanje podataka, Hamingovo dekodiranje i G.722 dekodiranje, spregnute takođe sa jednim 2,4 GHz bežičnim nRF24L01+ REFMOD primopredajnikom, koji će imati ulogu bežičnog prijemnika podataka.

Kao što je već rečeno, Ezairo 7100 SoC se sastoji od četiri različita procesorska jezgra različite namene. Jezgro koje će se ovde koristiti kao sprega sa nRF24L01+ REFMOD

primopredajnikom je ARM Cortex-M3 jezgro, koje će takođe biti zaduženo za G.722 kodiranje / dekodiranje, Hamingovo kodiranje / dekodiranje kao i za prepletanje / raspletanje podataka. CFX jezgro će biti zaduženo za inicijalizaciju i upravljanje celog Ezairo 7100 sistema. Jezgra HEAR i Filter Engine se neće koristiti u ovom radu.

Audio izvor, TV, je spojen sa Ezairo 7100 razvojnom pločom preko RCA (engl. *Radio Corporation of America*) priključka. Kad audio signal stigne do Ezairo 7100 razvojne ploče vrši se analogno-digitalno pretvaranje signala. Formirani odbirci se posredstvom IOC kontrolera prebacuju u memoriju (FIFO strukture) gde su dostupni jezgrima radi dalje obrade. Za komunikaciju jezgara sa periferijama koriste se DMA jedinice.

3.1 G.722 koder/dekoder

Kao što je već rečeno, ARM Cortex-M3 jezgro kao svoju periferiju sadrži G.722 koder / dekoder blokove realizovane u samoj fizičkoj arhitekturi. U ovom radu će se koristiti ti blokovi i to G.722 koder na strani predajnika i G.722 dekoder na strani prijemnika.

Ulaz G.722 kodera zahteva reč širine 32 bita (dva audio odbirka sa po 16 bitova) na osnovu koje proizvodi kodiranu reč od 8 bitova. To znači i da G.722 dekoder zahteva na svom ulazu reč širine 8 bitova na osnovu koje proizvodi dva audio odbirka po 16 bitova tj. jednu reč širine 32 bita.

3.2 Hamingov koder/dekoder

Namenski sistemi u odnosu na sisteme opšte namene imaju ograničene resurse. To znači da programer mora voditi računa o efikasnosti samog programskog koda. Složeniji kod zahteva više memorije, obično se duže izvršava pa samim tim i troši više energije što je vrlo bitno jer veliki broj namenskih sistema ima baterijsko napajanje.

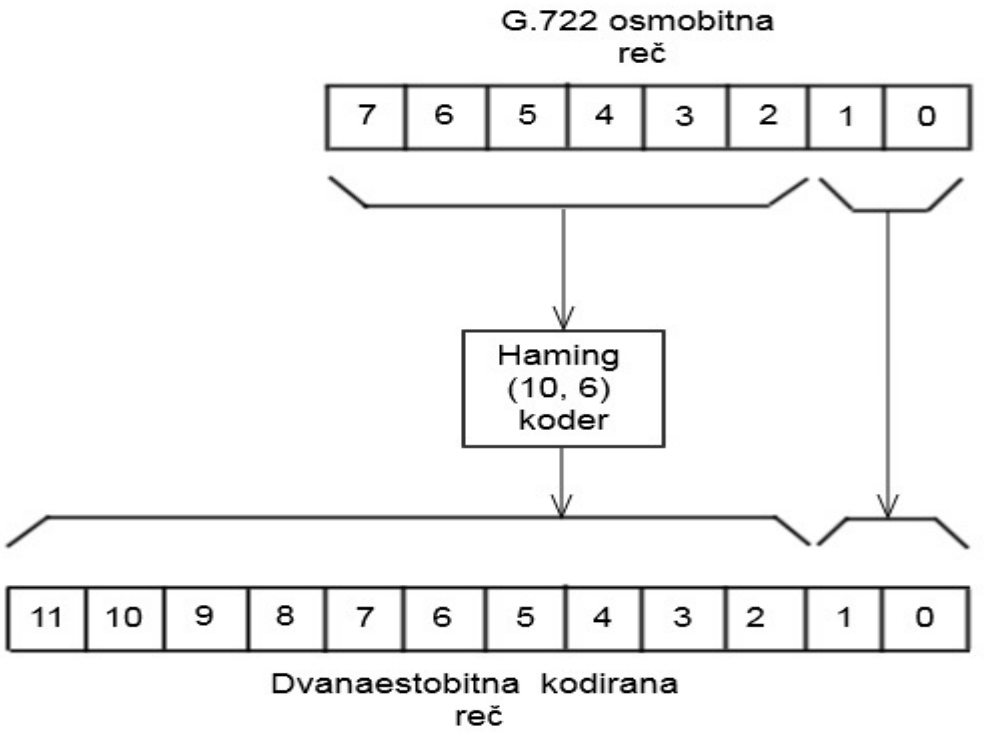
Zbog gore navedenog, izabrano je da se Hamingov koder / dekoder realizuje pomoću LUT (engl. *Look-Up Table*) tabele kojom će se izbeći izračunavanja u realnom vremenu i time smanjiti potrošnja MIPS (engl. *Million Instructions Per Second*). Sledeću stvar koju treba uzeti u obzir je koju verziju Hamingovog koda izabrati. Kao što je gore rečeno G.722 koder proizvodi reči širine 8 bitova a te reči će predstavljati ulaze u Hamingov koder. To znači da nam treba Hamingov kod (12, 8) koji će osmobitne reči kodirati u dvanaestobitne reči. Problem je u tome što ovaj Hamingov kod zahteva LUT tabelu sa $2^{12} = 4096$ vrednosti koja će morati biti smeštena

u memoriju na strani Hamingovog dekodera, dok će koderska strana zahtevati LUT tabelu sa 2^8 = 256 vrednosti. Da bi se smanjila veličina tabele izabran je Hamingov kod (10, 6) koji zahteva LUT tabelu sa 2^{10} = 1024 vrednosti na dekoderskoj strani i 2^6 = 64 vrednosti na koderskoj strani, koji kodira šestobitnu reč u desetobitnu. Korišćenje ovog Hamingovog koda omogućilo nam je istraživanje gde je pokazano da donja dva bita osmobitne G.722 kodirane reči ne igraju ulogu u percepciji kvaliteta audio signala većine ispitanika. To znači da je potrebno kodirati samo gornjih 6 bitova svake G.722 osmobitne reči sa Hamingovim (10, 6) koderom, dok se donja dva bita mogu ostaviti nezaštićena. Matrica proizvođač koja će se koristiti za proizvodnju kodiranih Hamingovih reči ima sledeći oblik:

$$\{I|P\} = \left(\begin{array}{cccccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right)$$

U pitanju je matrica standardnog ili sistematskog oblika što znači da je prvih 6 bitova novonastale kodirane reči jednako 6 bitova ulazne reči tj. reči koja se kodira, a poslednja četiri bita predstavljaju bitove parnosti.

Hamingov koder (10, 6) daje kao rezultat reč širine 10 bitova na koju treba dodati donja dva nezaštićena bita iz osmobitne G.722 kodirane reči, čime se od početne G.722 osmobitne reči dobija dvanaestobitna reč spremna za dalju obradu (u ovom slučaju je to prepletanje), slika 3.4:



Slika 3.4 Formiranje dvanaestobitne reči na izlazu Hamingovog kodera

Na dekoderskoj strani će biti LUT tabela sa 1024 vrednosti širine 4 bita, koje će biti smeštene u 128 memorijskih lokacija širine 32 bita radi uštede memorijskog prostora. To znači da će svaka povratna vrednost iz LUT tabele imati 16 mogućih vrednosti: od 0 do 15. Vrednosti od 0 do 5 će ukazivati na pozicije jednobitnih grešaka nad bitovima podataka, vrednosti od 6 do 9 će ukazivati na pozicije jednobitnih grešaka nad bitovima parnosti, vrednost 14 da ne postoji greška a vrednost 15 će ukazivati na dvobitne greške koje se ne mogu ispraviti.

Jednobitne greške će biti ispravljene na način da će se bitu na bitskoj poziciji na koju ukaže Hamingov dekoder promeniti vrednost, a kodirane reči sa pogrešna dva bita će biti zamenjene vrednošću 0xFE koja će isključiti (engl. *mute*) audio signal na kratko, tj. proizvešće tišinu. U slučaju pojave paketa koji sadrži više kodiranih reči sa dvobitnim greškama nego što je to dozvoljeno određenim pragom, ceo paket će biti zamenjen vrednostima koje će isključiti audio signal u trajanju tog paketa i time će se sprečiti pojava izobličenog audio signala.

Reči dekodirane Hamingovim dekoderom se šalju G.722 dekoderu, posle kojeg se dobijaju audio odbirci spremni za puštanje preko zvučnika.

3.3 Prepletanje/raspletanje podataka

Na ulaz bloka za prepletanje će biti dovedeni izlazi Hamingovog kodera, a to su u ovom slučaju dvanaestobitne reči. Proizvedena dvanaestobitna reč na izlazu Hamingovog kodera će biti uparena sa još jednom takvom dvanaestobitnom reči i tako zajedno će biti smeštene u jednu memorijsku lokaciju širine 32 bita radi uštede memorijskog prostora i lakše manipulacije pri prepletanju.

Tehnika prepletanja zbog svog načina rada zahteva obradu nad više kodiranih reči (jer se bitovi prepliću sa bitovima iz drugih kodiranih reči) što predstavlja blokovsku obradu. Ovaj blok ne sme da bude predugačak da ne bi uveo kašnjenje audio signala na strani prijemnika a isto tako ne sme da bude ni previše mali jer se time gubi efikasnost obrade.

Koristiće se dve tehnike prepletanja / raspletanja: prepletanje / raspletanje bitova i prepletanje / raspletanje kodiranih reči.

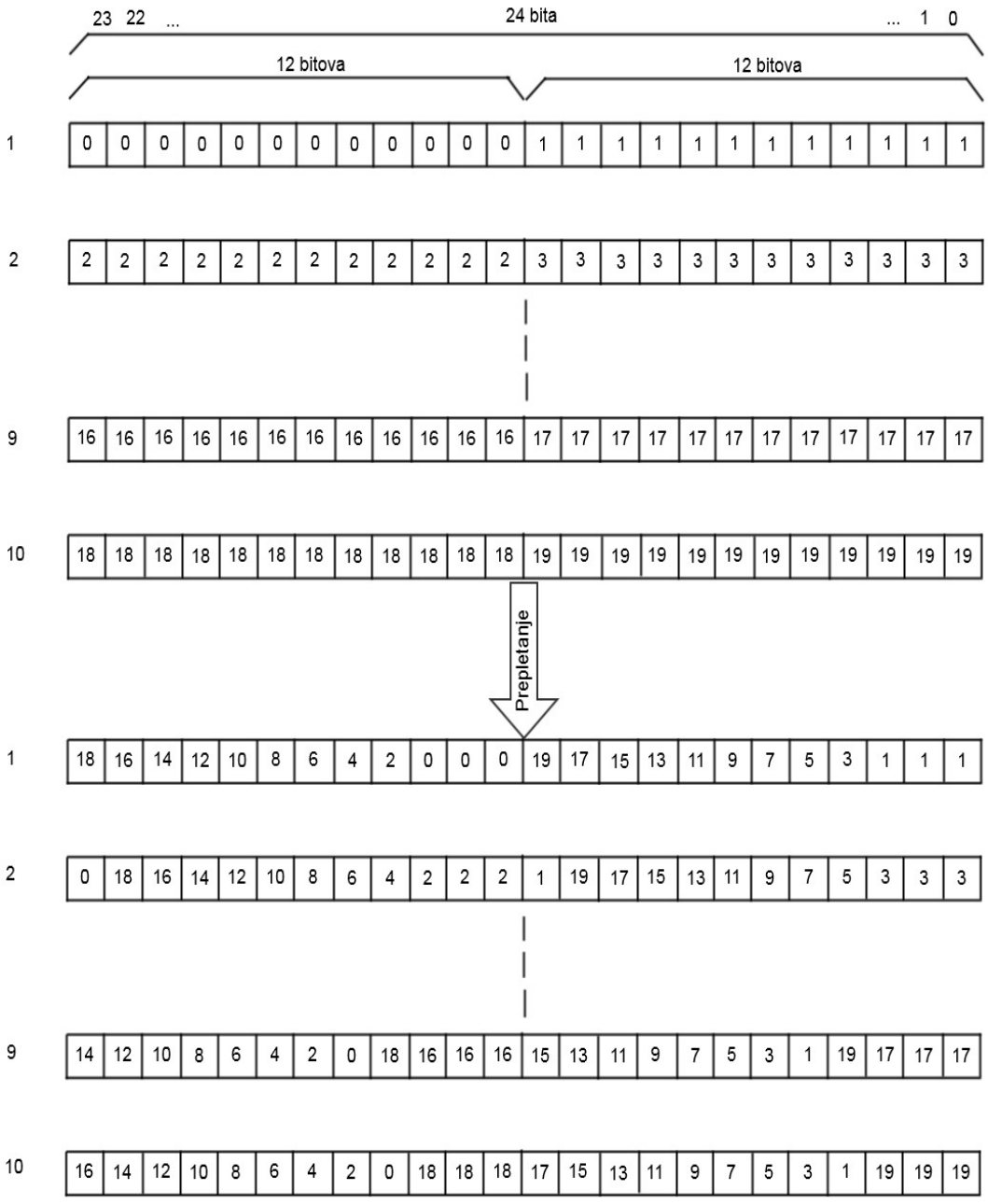
3.3.1 Prepletanje/raspletanje bitova

Za prepletanje bitova koristiće se blok od deset reči širine 24 bita. Kao što je već objašnjeno, svaka ta reč širine 24 bita predstavlja 2 reči sa izlaza Hamingovog kodera širine 12 bitova.

Prepletanje bitova neće promeniti njihovu bitsku poziciju u reči, nego će ih samo prebaciti u drugu reč. Donja dva bita svake dvanaestobitne reči neće biti obuhvaćena prepletanjem jer nisu zaštićena Hamingovim kodom, što znači da će ostati na svojim mestima.

Slika 3.5 prikazuje tehniku prepletanja nad 10 reči širine 24 bita. Posle prepletanja će svaka kodirana reč imati tačno po jedan bit iz svake kodirane reči, ne računajući nezaštićene bitove nad kojima se ne primenjuje tehnika prepletanja.

Na strani prijemnika se vrši suprotna operacija, tehnika raspletanja.



Slika 3.5 Tehnika prepletanja bitova

3.3.2 Prepletanje/raspletanje kodiranih reči

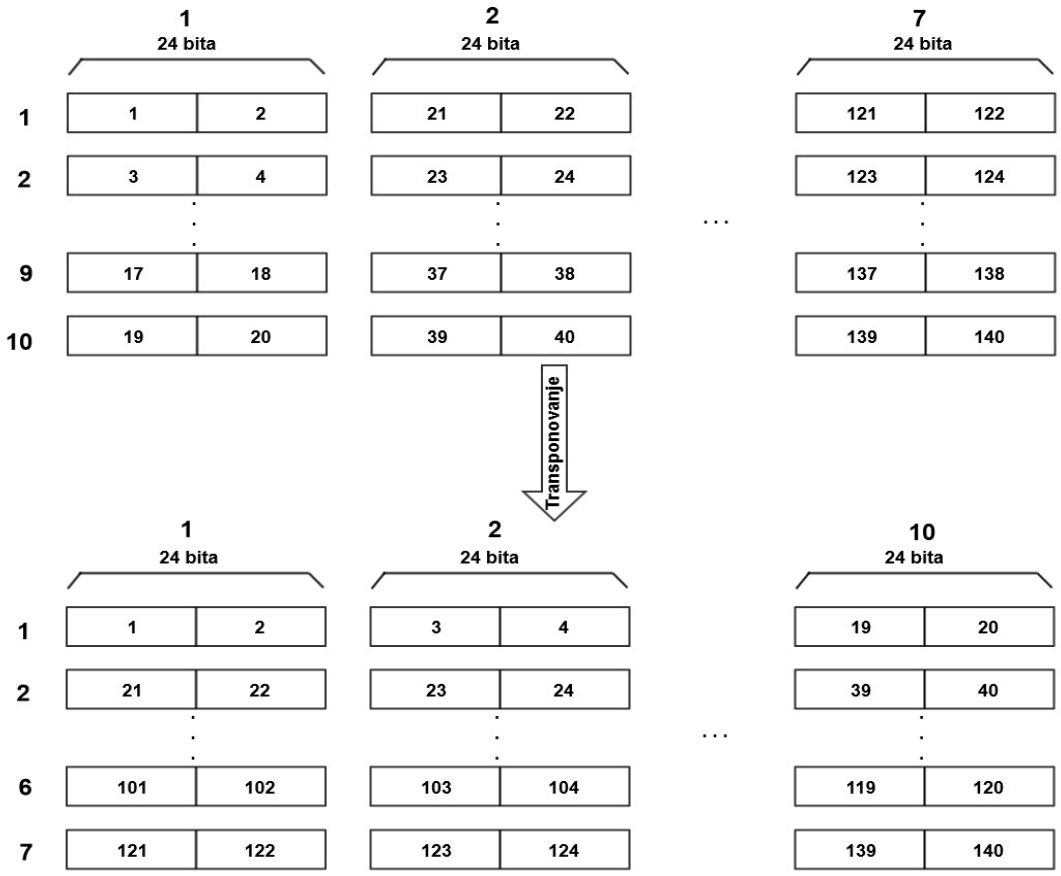
Posle prepletanja bitova u kodiranim rečima potrebno je preplesti još i same kodirane reči jer se time na strani raspletanja greške koje zauzimaju više uzastopnih bitskih pozicija u primljenom bitskom toku šire i raspodeljuju na više kodiranih reči i time omogućuju Hamingovom dekoderu efikasan rad.

Kako je ranije rečeno, prepletanje zahteva obradu nad blokom podataka određene veličine. Kako će taj blok podataka postati paket koji će se prenositi bežičnim putem do prijemnika ne sme da bude predugačak jer će uvesti kašnjenje audio signala na strani prijemnika a ni prekratak zbog efikasnosti obrade.

Za prepletanje kodiranih reči korisitće se 7 blokova od kojih će svaki sadržati 20 dvanaestobitnih reči dobijenih posle primene Hamingovog kodera. Slika 3.5 pokazuje jedan takav blok. To znači da će paket za slanje do prijemnika biti dužine 7 x 20 x 12 = 1680 bitova.

Prepletanje kodiranih reči će biti izvršeno koristeći operaciju transponovanja matrice. Matrica će biti predstavljena kao matrica sa sedam kolona i deset redova, gde će kolone predstavaljati blokovi sa slike 3.5 a redove reči širine 24 bita iz tih blokova.

Na slici 3.6 je prikazana realizacija prepletanja kodiranih reči:



Slika 3.6 Tehnika prepletanja reči

Ovim prepletanjem je dobijeno da svaka od 10 kolona posmatrane matrice, koja predstavlja paket spreman za slanje do prijemnika, sadrži samo po jedan bit iz svake kodirane reči. To znači da ako se u prenosu date matrice / paketa ošteti cela jedna kolona (168 bitova) posle raspletanja na strani prijemnika ta greška će se raspodeliti tako da će svaka kodirana reč sadržati samo jednobitnu grešku što Hamingov dekoder efikasno otkriva i ispravlja.

4. Programsko rešenje

Programsko rešenje ovog rada je podeljeno na više delova na sledeći način:

- Programska podrška vezana za predajničku / kodersku stranu
- Programska podrška vezana za prijemničku / dekodersku stranu

Svaka ta navedena programska podrška se sastoji od dva dela:

- Programska podrška koja se izvršava na CFX procesorskom jezgru
- Programska podrška koja se izvršava na ARM Cortex-M3 procesorskom jezgru

U narednim odeljcima će biti predstavljene samo osnovne funkcionalne jedinice sa svojim osnovnim funkcijama.

4.1 Programska podrška na predajničkoj/koderskoj strani

4.1.1 Programska podrška za CFX procesorsko jezgro

Programska podrška koja se izvršava na CFX procesorskom jezgru čini srž celog sistema, jer kao što je već rečeno CFX jezgro je glavno jezgro u sistemu koje upravlja svim drugim delovima sistema.

Osnovne funkcionalne jedinice koje čine ovu programsku podršku su:

- **app_init.cfs** – koristi se za inicijalizaciju i konfiguraciju svih korišćenih delova Ezairo 7100 sistema.

```
Initialize
```

Funkcija koja inicijalizuje i konfiguriše ceo sistem.
Parametri: nema
Povratna vrednost: nema

- **app_memory.cfs –** koristi se za zauzimanje i inicijalizaciju potrebne memorije sistema.
- **app_isr.cfs –** sadrži funkcije koje upravljaju određenim prekidima (engl. *Interrupt Service Routine – ISR*).
- **app.cfs –** glavna funkcionalna jedinica, sadrži *main* funkciju koja upravlja celim sistemom.

```
main
```

Glavna funkcija koja se izvršava na CFX jezgru. Pokreće i upravlja celim sistemom.
Parametri: nema
Povratna vrednost: nema

4.1.2 Programska podrška za ARM Cortex-M3 procesorsko jezgro

Programska podrška koja se izvršava na ARM Cortex-M3 procesorskom jezgru predstavlja realizaciju Hamingovog kodera kao i prepletanja bitova i reči. Takođe, ova programska podrška treba da obezbedi efikasnu komunikaciju ARM Cortex-M3 procesorskog jezgra sa bežičnim predajnikom.

Čine ga sledeće osnovne funkcionalne jedinice:

- **initialize.c –** koristi se za inicijalizaciju i konfiguraciju svih periferija i sprega koje su pod nadzorom ARM Cortex-M3 jezgra.

```
void initialize(void);
```

Funkcija koja inicijalizuje i konfiguriše sistem vezan za ARM Cortex-M3 jezgro.
Parametri: nema
Povratna vrednost: nema

- **main.c –** glavna funkcionalna jedinica, sadrži *main* funkciju koja upravlja celim ARM Cortex-M3 sistemom.

```
void main(void);
```

Glavna funkcija koja se izvršava na ARM Cortex-M3 jezgru. Pokreće ceo sistem i upravlja celim sistemom.
Parametri: nema
Povratna vrednost: nema

- **hamming_coder106.c –** koristi se za realizaciju Hamingovog (10, 6) kodera.

```
void hamming106_coder(uint16_t data, uint16_t *codeword);
```

Funkcija koja realizuje Hamingov (10, 6) koder.
Parametri:

1. `data` – reč širine 6 bitova koja će se kodirati
2. `*codeword` – kodirana reč širine 10 bitova

Povratna vrednost: nema

- **interleaver.c –** koristi se za realizaciju prepletanja bitova i reči.

```
void bit_interleaver(uint32_t *iblock, uint32_t *oblock);
```

Funkcija koja realizuje prepletanja bitova.
Parametri:

1. `*iblock` – memorijski niz koji sadrži podatke potrebne za prepletanje
2. `*oblock` – memorijski niz koji sadrži isprepletane podatke

Povratna vrednost: nema

```
void word_interleaver(uint32_t *iblock, uint32_t *oblock);
```

Funkcija koja realizuje prepletanje reči.
Parametri:

1. `*iblock` – memorijski niz koji sadrži podatke potrebne za prepletanje

2. `*oblock` – memorijski niz koji sadrži isprepletane podatke

Povratna vrednost: nema

4.2 Programska podrška na prijemničkoj/dekoderskoj strani

4.2.1 Programska podrška za CFX procesorsko jezgro

I ovde programska podrška koja se izvršava na CFX procesorskom jezgru čini srž celog sistema, jer kao što je već rečeno CFX jezgro je glavno jezgro u sistemu koje upravlja svim drugim delovima sistema.

Osnovne funkcionalne jedinice koje čine ovu programsku podršku su:

- **app_init.cfs** – koristi se za inicijalizaciju i konfiguraciju svih korišćenih delova Ezairo 7100 sistema.

`Initialize`

Funkcija koja inicijalizuje i konfiguriše ceo sistem.
Parametri: nema
Povratna vrednost: nema

- **app_memory.cfs –** koristi se za zauzimanje i inicijalizaciju potrebne memorije sistema.
- **app_isr.cfs –** sadrži funkcije koje upravljaju određenim prekidima (engl. *Interrupt Service Routine – ISR*).
- **app.cfs –** glavna funkcionalna jedinica, sadrži *main* funkciju koja upravlja celim sistemom.

`main`

Glavna funkcija koja se izvršava na CFX jezgru. Pokreće i upravlja celim sistemom.
Parametri: nema
Povratna vrednost: nema

4.2.2 Programska podrška za ARM Cortex-M3 procesorsko jezgro

Programska podrška koja se izvršava na ARM Cortex-M3 procesorskom jezgru predstavlja realizaciju Hamingovog dekodera kao i raspletanja bitova i reči. Takođe, ova programska podrška treba da obezbedi efikasnu komunikaciju ARM Cortex-M3 procesorskog jezgra sa bežičnim prijemnikom.

Čine ga sledeće osnovne funkcionalne jedinice:

- **initialize.c –** koristi se za inicijalizaciju i konfiguraciju svih periferija i sprega koje su pod nadzorom ARM Cortex-M3 jezgra.

```
void initialize(void);
```

Funkcija koja inicijalizuje i konfiguriše sistem vezan za ARM Cortex-M3 jezgro.
Parametri: nema
Povratna vrednost: nema

- **main.c –** glavna funkcionalna jedinica, sadrži *main* funkciju koja upravlja celim ARM Cortex-M3 sistemom.

```
void main(void);
```

Glavna funkcija koja se izvršava na ARM Cortex-M3 jezgru. Pokreće i upravlja celim sistemom.
Parametri: nema
Povratna vrednost: nema

- **hamming_decoder106.c –** koristi se za realizaciju Hamingovog (10, 6) dekodera.

```
int hamming106_decoder(uint16_t codeword, uint16_t *data);
```

Funkcija koja realizuje Hamingov (10, 6) dekoder.
Parametri:

1. `codeword` – reč širine 10 bitova koja će se dekodirati
2. `*data` – dekodirana reč širine 6 bitova

Povratna vrednost:

1. 0 – nema greške
2. 1 – popravljiva greška
3. 2 – nepopravljiva greška

- **deinterleaver.c –** koristi se za realizaciju raspletanja bitova i reči.

```
void bit_deinterleaver(uint32_t *iblock, uint32_t * oblock);
```

Funkcija koja realizuje raspletanje bitova.
Parametri:

1. `*iblock` – memorijski niz koji sadrži podatke potrebne za raspletanje
2. `*oblock` – memorijski niz koji sadrži raspletene podatke

Povratna vrednost: nema

```
void word_deinterleaver(uint32_t *iblock, uint32_t *oblock);
```

Funkcija koja realizuje raspletanje reči.
Parametri:

1. `*iblock` – memorijski niz koji sadrži podatke potrebne za raspletanje
2. `*oblock` – memorijski niz koji sadrži raspletene podatke

Povratna vrednost: nema

5. Ispitivanja i rezultati

U cilju ispitivanja i verifikacije realizovanog rešenja na ciljnoj platformi isprojektovan je i realizovan automatski radni okvir za ispitivanje (engl. *automated test framework*).

Ispitivanje realizovanog sistema je vršeno upoređivanjem izlaza Ezairo 7100 sistema kao i izlaza referentnog sistema – ispitivanje obuhvata verifikaciju realizacije Hamingovog kodiranja / dekodiranja i prepletanja / raspletanja bitova i reči.

Programska podrška, Hamingovo kodiranje / dekodiranje i prepletanje / raspletanje G.722 audio odbiraka, koje se izvršava na Ezairo 7100 platformi je prvo realizovano u vidu C programa pomoću *Microsoft Visual Studio* razvojnog okruženja. Za verifikaciju tog C programa je korišćen ispitni program, koji je takođe napisan u C programskom jeziku, koji sadrži proizvođača uzastopnih brojeva koji proizvodi brojeve od 0 do 255 i deo za dodavanje jendobitnih i dvobitnih grešaka. Ti proizvedeni brojevi predstavljaju osmobitne G.722 audio odbirke. Da bi se ispitala glavna karakteristika realizovanog Hamingovog koda a to je otkrivanje i ispravljanje jednobitnih grešaka i otkrivanje dvobitnih, napravljena je provera iz dva dela. Prvi deo je zadužen za ispitivanje dela vezanog za otkrivanje i ispravljanje jednobitinih grešaka, i to na sledeći način. Svaki proizvedeni osomobitni broj tj. samo gornjih 6 bitova (64 moguće vrednosti) je kodirano Hamingovim (10, 6) koderom, a svakom desetobitnom izlazu Hamingovog kodera je dodeljena jednobitna greška promenom vrednosti bita na određenoj bitskoj poziciji. Ta jednobitna greška je dodeljivana Hamingovoj kodiranoj reči na način da su sve moguće kombinacije pojave jednobitne greške na određenoj bitskoj poziciji pokrivene. To znači da je svaka desetobitna Hamingova kodirana reč proizvodila 10 novih desetobitnih reči a gde je svaka imala bit greške na različitoj bitskoj poziciji. Ove kodirane reči su zapisane u datoteke da bi se kasnije mogle koristiti i za ispitivanje realizovanog rešenja na ciljnoj platformi. Te kodirane reči su zatim dekodirane Hamingovim (10, 6) dekoderom a izlaz iz dekodera je upoređen sa ulazom u

Hamingov koder a rezultat upoređivanja je potvrdio da su jednobitne greške uspešno otkrivene i ispravljene. Dekodirane reči su takođe zapisane u datoteku da bi se mogle koristiti za ispitivanje realizovanog rešenja na ciljnoj platformi. Drugi deo zadužen za ispitivanje dela vezanog za otkrivanje dvobitnih grešaka je realizovan na sličan način. Izlazima Hamingovog kodera je dodeljivana dvobitna greška i to tako da su sve moguće kombinacije pojavljivanja dvobitne greške na određenim bitskim pozicijama pokrivene. To znači da je svaka Hamingova kodirana reč proizvodila 45 novih reči a gde je svaka imala dvobitnu grešku postavljenu na različitim bitskim pozicijama. Te kodirane reči su takođe dekodirane Hamingovim dekoderom a broj otkrivenih grešaka je odgovarao stvarnom broju. Ovim je verifikovana C realizacija Hamingovog kodera / dekodera. Realizacija prepletanja / raspletanja je verifikovana na sledeći način. Nad Hamingovim kodiranim rečima sa dodatim jednobitnim i dvobitnim greškama je izvršena operacija prepletanja a zatim i operacija raspletanja i Hamingovog dekodiranja. Delovanje operacije prepletanja se poništava operacijom raspletanja tako da krajnji rezultat treba da je identičan početnim vrednostima, što je ovim ispitivanjem i potvrđeno. Podaci posle Hamingovog kodiranja i prepletanja su zapisani u jednu datoteku i oni čine referentne ulazne podatke a podaci dobijeni posle raspletanja i Hamingovog dekodiranja su zapisani u drugu datoteku i oni čine referentne izlazne podatke. Te datoteke će se koristiti za ispitivanje pomenutih realizacija na ciljnoj platformi. Ovim ispitivanjem je verifikovan C program i proglašen je referentnim rešenjem.

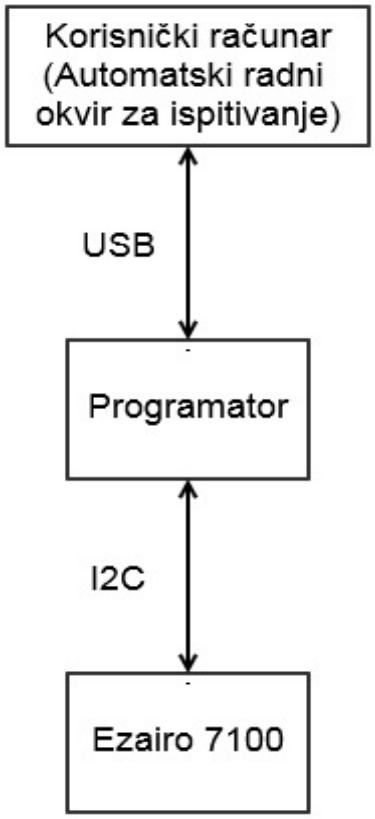
Referentno rešenje je nakon verifikacije prilagođeno Ezairo platformi.

Automatski radni okvir za ispitivanje je realizovan u potpunosti pomoću programskog jezika Pajton (engl. *Python*). Realizovani radni okvir za ispitivanje za komuniciranje sa Ezairo 7100 razvojnom pločom i njenim komponentama koristi prolaz za otkrivanje i ispravljanje grešaka (engl. *debug port*), a kao programsku spregu (engl. *Application Programming Interface – API*) koristi CTK (engl. *Communication Toolkit*) spregu koju je obezbedio, kao i ostale potrebne alate za rad sa Ezairo 7100 platformom, *On semiconductor*.

Realizacija automatskog radnog okvira za ispitivanje se može opisati na sledeći način:

1. Automatski radni okvir za ispitivanje napisan u Pajton programskom jeziku se izvršava na korisničkom računaru.
2. Korisnički računar je preko programatora povezan sa Ezairo 7100 razvojnom pločom. Programator za komunikaciju sa Ezairo 7100 razvojnom pločom koristi prolaz za otkrivanje i ispravljanje grešaka. Ova sprega se koristi za slanje komandi i ulaznih podataka Ezairo 7100 sistemu kao i za izvlačenje izlaznih podataka iz datog sistema.

Na slici 5.1 je prikazan blok dijagram realizovanog automatskog radnog okvira za ispitivanje:



Slika 5.1 Automatski radni okvir za ispitivanje

Ispitivanje je podeljeno u dve faze koje se realizuju na isti način samo sa različitim podacima:

- Ispitivanje i verifikacija predajničke / koderske strane
- Ispitivanje i verifikacija prijemničke / dekoderske strane

Automatski radni okvir za ispitivanje zahteva dve datoteke kao svoj ulaz koje su napravljene na gore opisan način pomoću *Microsoft Visual Studio* razvojnog okruženja. Jedna datoteka sadrži podatke, referentne ulazne podatke, koji će biti slani do Ezairo 7100 razvojne ploče preko programatora i CTK sprege. Ti podaci predstavljaju ulazne podatke za blok obrade *Hamingov koder – prepletanje bitova i reči* u fazi ispitivanja predajničke / koderske strane, ili za blok obrade *raspletanje reči i bitova – Hamingov dekoder* u fazi ispitivanja prijemničke / dekoderske strane. Druga datoteka sadrži referentne izlazne podatke koji će biti upoređivani na nivou bita sa izlaznim podacima proizvedenim u Ezairo 7100 sistemu a koji će biti iščitani iz memorije Ezairo 7100 sistema pomoću već pomenutog programatora i CTK sprege. Posle

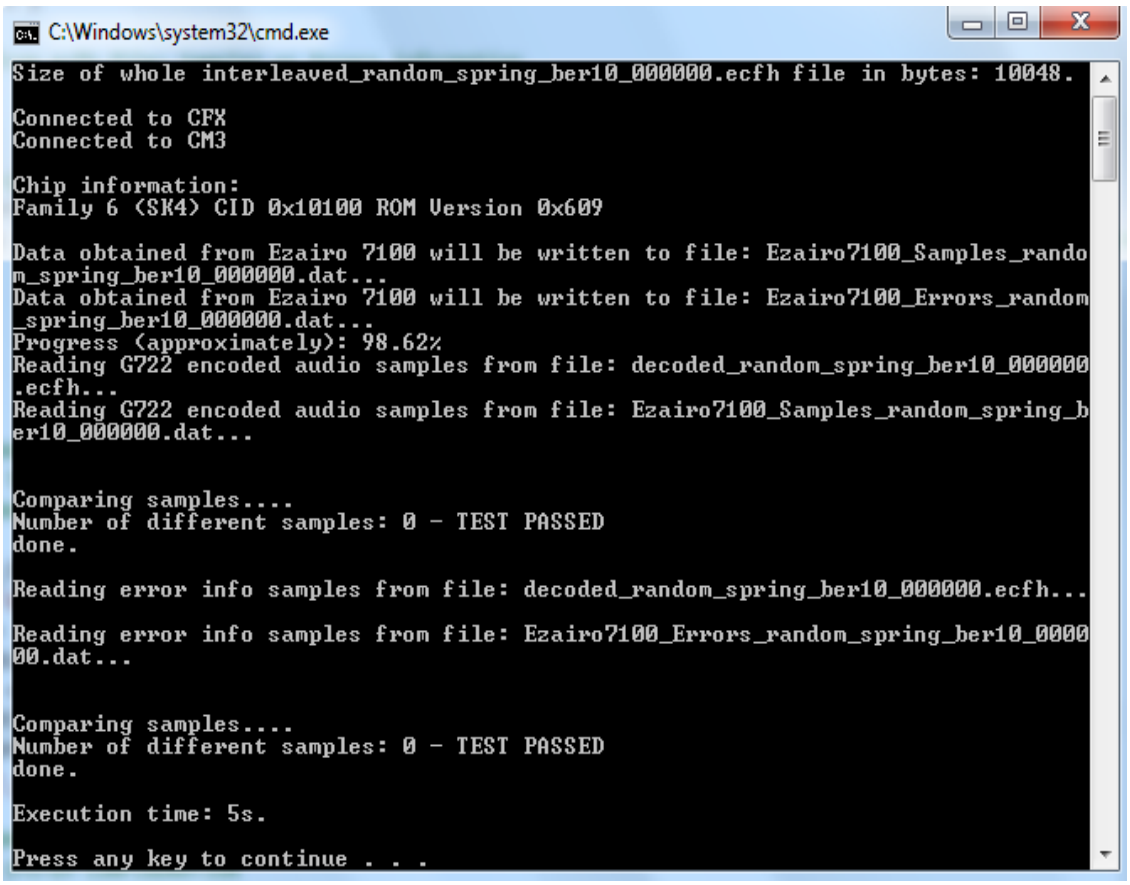
upoređivanja datih podataka, automatski radni okvir za ispitivanje ispisuje rezultate ispitivanja i verifikacije na ekran korisničkog računara.

Pored dve datoteke koje automatski radni okvir zahteva kao svoj ulaz, potrebno je podesiti i sledeće osnovne parametre koji su potrebni za funkcionisanje istog:

1. Izabrati koje se procesorsko jezgro koristi u ispitivanju tj. na kojem procesorskom jezgru se izvršava programska podrška koja se želi ispitati i verifikovati.
2. Navesti ime funkcije date programske podrške čija se realizacija želi ispitati i verifikovati.
3. Navesti ime ulaznog memorijskog niza kao i njegovu veličinu, koji služi za prihvat ulaznih podataka.
4. Navesti ime izlaznog memorijskog niza kao i njegovu veličinu, koji će sadržati rezultate određene obrade i iz kojeg će se iščitavati ti izlazni podaci.
5. Navesti imena izlaznih datoteka u koje će biti smešteni podaci iščitani iz Ezairo 7100 sistema.
6. Navesti ime promenljive čijim podešavanjem se omogućuje ispitivanje na ciljnoj platformi.

Nakon ovog podešavanja, automatski radni okvir za ispitivanje automatski šalje blok po blok ulaznih podataka Ezairo 7100 sistemu, zatim zahteva određenu obradu nad njima tj. pokreće zadatu funkciju, iščitava izlazne podatke kao rezultate obrade, zapisuje ih u datoteke i vrši upoređivanje sa referentnim podacima a na kraju ispisuje rezultat upoređivanja.

Na ovaj način je potvrđena bit egzaktnost između referentnog programskog rešenja i programske podrške realizovane na Ezairo 7100 platformi, kako na strani predajnika / kodera tako i na strani prijemnika / dekodera, slika 5.2.



Slika 5.2 Prikaz rezultata ispitivanja

6. Zaključak

U ovom radu je prikazano jedno rešenje G.722 širokopojasnog audio kodera sa kontrolom grešaka unapred na namenskoj platformi Ezairo 7100 komapnije *On Semiconductor*. Cilj zadatka je bio da se realizuje audio sistem koji će omogućiti efikasno bežično prenošenje audio signala. Rešenje prikazano u ovom radu uspešno realizuje dati cilj.

Na osnovu zadatih zahteva u zadatku i mogućnosti ciljne platforme realizovan je Hamingov (10, 6) koder / dekoder kao što je i isprojektovan i realizovan blok za prepletanje i raspletanje kako pojedinačnih bitova tako i kodiranih reči a sve u cilju povećanja robusnosti audio bitskog toka na greške tokom bežičnog prenosa između predajnika i prijemnika. Za kodiranje audio odbiraka u G.722 format kao i za dekodiranje G.722 audio kodiranih reči je korišćen G.722 blok realizovan u fizičkoj arhitekturi same Ezairo 7100 platforme.

Za potpuno ispitivanje i verifkaciju realizovanog sistema je isprojektovan i realizovan automatski radni okvir za ispitivanje koristeći Pajton programski jezik. Ispitivanje i verifikacija prikazanog rešenja je obuhvatila dve faze:

1. Ispitivanje i verifikacija rešenja na predajničkoj / koderskoj strani
2. Ispitivanje i verifikacija rešenja na prijemničkoj / dekoderskoj strani

Automatski radni okvir za ispitivanje je upoređivao na nivou bita referentne podatke (proizvedene na korisničkom računaru) i podatke iščitane iz memorije Ezairo 7100 sistema i time uspešno verifikovao ispravnost realizacije datog rešenja.

Poboljšanja ovog rada se mogu realizovati uvođenjem efikasnijih FEC algoritama, korišćenjem bezičnih primopredajnika drugih opsega učestanosti (korišćeni 2,4 GHz opseg spada u grupu opsega slobodnih za korišćenje na svetskom nivou dok neki drugi opsezi zahtevaju licenciranje) kao i korišćenjem drugih antena radi prilagođenja određenim zahtevima (potrošnja, domet).

7. Literatura

[1] Dai Tracy Yang, Chris Kyriakakis, C.-C. Jay Kuo: *High-Fidelity Multichannel Audio Coding*, Hindawi Publishing Corporation, Egypt, 2006, pp. 3.

[2] On Semiconductor: *Hardware Reference manual for Ezairo 7100*, http://www.onsemi.com/PowerSolutions/content.do?id=18352, jul 2014

[3] Recommendation ITU-T G.722: *7 kHz audio-coding within 64 kbit/s*, http://www.itu.int/rec/T-REC-G.722-201209-I/en, jul 2014

[4] M. E. Madkour, S. E. Soliman, M. I. Moawad, F. E. Abd El-Samie, P. K. Varshney: *Coding and Interleaving Schemes for Wireless Sensor Networks*, The 8th International Conference on Informatics and Systems, 2012

[5] Colin Perkins, Orion Hodson, Vicky Hardman: *A Survey of Packet Loss Recovery Techniques for Streaming Audio*, IEEE Network, 1998

[6] R. W. Hamming: *Error Detecting and Error Correcting Codes*, The Bell System Technical Journal, USA, 1950

[7] Yuanyuan Cui, Xunying Zhang: *Research and implementation of interleaving grouping Hamming code algorithm*, International Conference on Signal Processing, Communication and Computing, 2013

[8] Jianwu Zhao, Yibing Shi: *A Novel Approach to Improving Burst Errors Correction Capability of Hamming Code*, International Conference on Communications, Circuits and Systems, 2007

[9] U. K. Kumar, B. S. Umashankar: *Improved Hamming Code for Error Detection and Correction*, 2nd International Symposium on Wireless Pervasive Computing, 2007