



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Немања Обрадовић

**Реализација решења за мерење
грешке аудио/видео синхронизације
у дигиталном мултимедијалном
садржају на вишејезгарној IBM Cell
BE платформи**

МАСТЕР РАД

Нови Сад, 2013

	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:	Никола Јорговановић

Студент:	Немања Обрадовић	Број индекса:	11073
Област:	Програмска подршка у телевизији и обради слике 2		
Ментор:	Иштван Пап		

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:

Реализација решења за мерење грешке аудио/видео синхронизације у дигиталном мултимедијалном садржају на вишејезгарној IBM Cell BE платформи

ТЕКСТ ЗАДАТКА:

У оквиру задатка потребно је реализовати решење које ће мерити грешку аудио/видео синхронизације у дигиталном мултимедијалном садржају.

Приступ се заснива на поређењу анализираног сигнала са референтним сигналом који је без грешке. Решење је првенствено намењено анализи емитованог садржаја на страни пријемника, те је током реализације поступка мерења потребно уважити чињеницу да током емитовања може доћи до деградације сигнала.

Поступак је прво потребно верификовати развојем одговарајућег Matlab кода, а затим реализовати у програмском језику C и пренети на вишејезгарну циљну платформу IBM Cell BE која се користи у Playstation 3.

Руководилац студијског програма:	Ментор рада:

Примерак за: О - Студента; О - Ментора



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:	Немања Обрадовић
Ментор, МН:	Иштван Пап
Наслов рада, НР:	Реализација решења за мерење грешке аудио/видео синхронизације у дигиталном мултимедијалном садржају на вишејезгарној IBM Cell BE платформи
Језик публикације, ЈП:	Српски / латиница
Језик извода, ЈИ:	Српски
Земља публикаовања, ЗП:	Република Србија
Уже географско подручје, УГП:	Војводина
Година, ГО:	2013
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)	7/49/23/7/27/0/0
Научна област, НО:	Електротехника и рачунарство
Научна дисциплина, НД:	Рачунарска техника
Предметна одредница/Кључне речи, ПО:	Аудио/видео синхронизација, дигитална телевизија, IBM Cell BE платформа
УДК	
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У овом раду је приказан опис реализације програмске подршке за мерење грешке аудио видео синхронизације на IBM Cell BE платформи.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	Председник: проф. др Никола Теслић
	Члан: доц. др Милош Сланкаменац
	Члан, ментор: доц. др. Иштван Пап
	Потпис ментора



KEY WORDS DOCUMENTATION

Accession number, ANO :		
Identification number, INO :		
Document type, DT :	Monographic publication	
Type of record, TR :	Textual printed material	
Contents code, CC :	Master Thesis	
Author, AU :	Nemanja Obradović	
Mentor, MN :	Istvan Papp, PhD	
Title, TI :	Solution implementation for measurement of audio video synchronisation error in multimedia content on IBM Cell BE 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 :	2013	
Publisher, PB :	Author's reprint	
Publication place, PP :	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/49/23/7/27/0/0	
Scientific field, SF :	Electrical Engineering	
Scientific discipline, SD :	Computer Engineering	
Subject/Key words, S/KW :	Audio video synchronization, digital television, IBM Cell BE platform	
UC		
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N :		
Abstract, AB :	This paper describes implementation of software solution for audio video synchronization measurement on IBM Cell BE platform.	
Accepted by the Scientific Board on, ASB :		
Defended on, DE :		
Defended Board, DB :	President:	Nikola Teslić, PhD
	Member:	Milos Slankamenac, PhD
	Member, Mentor:	Istvan Papp, PhD
		Menthor's sign

Sadržaj

1.	Uvod	1
2.	Teoretske osnove	3
2.1	Uzroci	3
2.2	Značaj pravilne sinhronizacije.....	5
2.3	Međunarodna regulativa	6
2.3.1	Preporuke.....	6
2.4	Ciljna platforma.....	7
2.4.1	PPE	8
2.4.2	SPE	9
2.4.3	EIB magistrala	10
2.4.4	Memorijski i U/I kontroleri	10
2.4.5	Performanse.....	11
2.4.6	Optimizacija SPE koda.....	12
2.4.7	Sony Playstation 3	16
2.4.8	Virtuelizacija, GuestOS	16
2.4.9	Programski alati.....	16
2.4.10	Primena.....	18
3.	Koncept rešenja	20
3.1	Mrežni prijem i slanje.....	22
3.1.1	RT-RK RT-AV100.....	22
3.1.2	Mrežna oprema.....	23
3.2	PlayStation 3.....	23
3.2.1	Algoritmi	25
3.2.2	Video matching algoritam - varijansa	26
3.2.3	Audio matching algoritam - fazna korelacija	27
3.3	Radna stanica za praćenje rezultata.....	29
4.	Opis rešenja	30
4.1	NAViCSim	31
4.2	Adaptacija NAViC Simulatora.....	32
4.3	PQM Monitor	32
4.4	Adaptacija PQM Monitora	33
4.5	Metodologija razvoja.....	34

4.5.1	Matlab simulacija	35
4.5.2	C programski kod za x86 arhitekturu	36
4.5.3	Programski kod sa posebnim funkcijama za x86 arhitekturu.....	37
4.6	Adaptacija programskog koda za Cell platformu	38
4.7	Opis Cell programskog rešenja	39
4.7.1	PPU.....	39
4.7.2	SPU.....	43
5.	Ispitivanje i rezultati	44
5.1.1	Verifikacija ispravnosti rada	44
5.1.2	Merenje performansi	46
6.	Zaključak	47
7.	Literatura	48

Spisak slika

Slika 1 – Uneto kašnjenje tipičnih komponenti u AV lancu [2]	4
Slika 2 – Uticaj AV sinhronizacije na subjektivne ocene gledalaca[3]	5
Slika 3 – Nalazi istraživanja objavljeni u ITU BT.1359-1 [3]	6
Slika 4 – Unutrašnja arhitektura Cell/BE mikroprocesora	8
Slika 5 – Komponente PPE procesora	9
Slika 6 – Komponente SPE procesora	9
Slika 7 – Element Interconnect Bus	10
Slika 8 – SISD i SIMD instrukcije	13
Slika 9 – Ilustacija zauzeća procesora kod jednostruko spregnutog punjenja	15
Slika 10 – Ilustracija zazeća procesora kod dvostruko spregnutog punjenja	15
Slika 11 – Prevođenje Cell izvornog koda	17
Slika 12 – FFT performanse PS3 3.2 GHz Cell B/E, 6 SPE jezgara [23]	19
Slika 13 – FFT performanse 3.0 GHz Intel Xeon Core Duo 64-bit [23]	19
Slika 14 – Koncept rešenja	21
Slika 15 – Ulazni priključci RT-AV 100	22
Slika 16 – Unutrašnja arhitektura RT-RK RT-AV100	23
Slika 17 – Gigabitni mrežni svič sa pet priključaka	23
Slika 18 – Odnos kašnjenja i greške sinhronizacije	24
Slika 19 – Poređenje rezultata unakrsne i fazne korelacije	29
Slika 20 – Pregled fizičkih komponentata rešenja	30
Slika 21 – Glavni i video prozor NAViC Simulatora	31
Slika 22 – PQM monitor (levo) i modifikovana verzija (desno)	33
Slika 23 – Metodologija razvoja i verifikacije	34
Slika 24 – Rezultat rada Matlab simulacije algoritma fazne korelacije	36
Slika 25 – Verifikacija ispravnosti prenosa slike i zvuka	45
Slika 26 – Verifikacija ispravnog rada algoritama	45
Slika 27 – Sistem korišćen za merenje performansi	46

Spisak tabela

Tabela 1 – <i>Preporuke međunarodnih organizacija</i>	7
Tabela 2 – <i>Realna brzina prenosa podataka između SPE jezgara</i>	12
Tabela 3 – <i>Kašnjenje i instrukcija u protočnoj strukturi [22]</i>	14
Tabela 4 – <i>Raspoloživi ulazno-izlazni uređaji</i>	20
Tabela 5 – <i>Širina propusnog opsega za video formate</i>	21
Tabela 6 – <i>Širina propusnog opsega za audio formate</i>	21
Tabela 7 – <i>Performanse programskog rešenja</i>	46

Skraćenice

DSP – *Digital signal processing*
TV – *Television*
VQA – *Video Quality Analyzer/Assessment*
STB – *Set-top Box*
PQM – *Picture Quality Monitor*
NAViC – *Network Attached Video Capture*
PS3 – *Sony Playstation 3*
Cell/BE – *Cell Broadband Engine*
PPE – *Power Processing Element*
PPU – *Power Processing Unit*
SPE – *Synergistic Processor Element*
SPU – *Synergistic Processor Unit*
EIB – *Element Interconnect Bus*
MFC – *Memory Flow Controler*
FFT – *Fast Fourier Transform*
SISD – *Single Instruction Single Data*
SIMD – *Single Instruction Multiple Data*
IDE – *Integrated Development Environment*
API – *Application Programming Interface*
Codec – *Coder/Decoder*
FPS – *Frames per second*

1. Uvod

Pojam sinhronizacije zvuka i slike (eng. *audio video synchronisation*) se odnosi na relativnu vremensku usklađenost zvuka (audio) i slike (video) tokom stvaranja, post produkcije, prenosa, prijema i reprodukcije audio-video materijala.

Jedan od prioritetnih ciljeva digitalne televizije je da isporuči dobro sinhronizovan audio-video signal krajnjem korisniku. Pravilna audio-video sinhronizacija je veoma važna za gledaoca jer utiče na percepciju kvaliteta snimka. Sinhronizacija je takođe bitna i potrošačkoj industriji jer nove tehnologije stvaraju primetno kašnjenje u obradi audio i video signala. Kada primete probleme poput greške sinhronizacije, efekta blokovskih grešaka (eng. *blocking artefact*) ili crnog ekrana gledaoci često promene kanal. Zbog toga je važno otkriti i otkloniti greške pre nego što ih gledalac primeti.

Televizijski i predajni sistemi su kompleksni lanci uređaja za snimanje, obradu, kompresiju i skladištenje. Svaka komponenta u audio-video lancu unosi neki stepen kašnjenja u signal [1]. Kašnjenje audio i video signala je najčešće nejednako pa se može reći da svaka komponenta u audio-video lancu uzrokuje grešku u sinhronizaciji na svom izlazu. Algoritmi za ispravljanje greške sinhronizacije moraju uzeti u obzir kašnjenja kod obrade kako bi se oba signala isporučila gledaocu u pravo vreme i time omogućilo bolje iskustvo za gledaoca. Posebno je problematično što deo ukupnog kašnjenja nastaje kada signal napusti televizijski studio u distribucionom sistemu kao i pri naknadnoj obradi u samom televizoru ili STB (eng. *Set-Top Box*) uređaju.

U audio-video lancu slika i zvuk su razdvojeni i na strani predajnika i na strani prijemnika. Njihova obrada je takođe nezavisna pa se greška sinhronizacije može javiti između primljenog i poslatog (referentnog) signala. Stoga se referentni i signal koji se testira upoređuju (eng. *matching*) u cilju merenja greške audio-video sinhronizacije.

U oblasti procene kvaliteta slike (eng. *Video Quality Assesment*), uključujući i merenje greške sinhronizacije, algoritmi za procenu kvaliteta se dele na:

- Pristup sa punom referencom
- Pristup sa smanjenom referencom
- Pristup bez reference

Pristup sa punom referencom se zasniva na upoređivanju ulaznog AV (eng. audiovideo - AV) signala sa referentnim signalom. Meri se kašnjenje video signala tako što se porede slike (eng. *frames*) referentnog i ulaznog video signala. Takođe se meri kašnjenje audio signala poređenjem referentnog i ulaznog audio signala. Ukoliko je kašnjenje audio i video signala nejednako konstatuje se greška sinhronizacije.

Pristup sa smanjenom referencom podrazumeva prisustvo određenih mera ili osobina referentnog signala ali sam signal nije prisutan u vreme merenja. U slučaju merenja audio-video sinhronizacije obično se koristi mera audio-video korelacije referentnog signala kao osnov za poređenje sa ulaznim signalom.

Pristup bez reference koristi samo podatke koji se mogu dobiti iz ulaznog signala. Referentni signal kao i informacije o njemu nisu prisutni u vreme merenja. Primena metrike bez reference za merenja audio-video sinhronizacije je ograničena samo na pojedine tipove signala tj. u opštem slučaju ne daje dovoljno dobre rezultate.

U ovom radu prikazan je razvoj i implementacija algoritma sa punom referencom za otkrivanje i merenje greške sinhronizacije audio-video signala. Motiv za rad je automatizacija merenja greške sinhronizacije između referentnog signala i signala koji se testira. Tipična primena predloženog rešenja je merenje kašnjenja i greške sinhronizacije koji nastaju nakon nekog bloka za obradu signala, distributivne mreže ili samog prijemnika. Upotrebom opisanog rešenja je takođe moguće automatizovati praćenje sinhronizacije za veliki broj signala kao i alarmiranje ukoliko se pojavi greška sinhronizacije veća od definisanog praga tolerancije.

2. Teoretske osnove

Napretkom tehnologije a naročito dolaskom digitalne televizije kvalitet zvuka i slike se dramatično popravio učinivši greške audio-video sinhronizacije приметnim.

2.1 Uzroci

U tipičnom televizijskom sistemu zvuk i slika se prenose odvojenim putanjama što dovodi do različitih kašnjenja u obradi.

Putanja obrade zvuka je manje zahtevna u smislu unošenja kašnjenja. Zbog jednostavne obrade, kašnjenje zvuka najčešće je uslovljeno brzinom prostiranja zvuka i propagacije električnog signala. Brzina prostiranja zvuka pri temperaturi od 20 stepeni celzijusa je 340 metara u sekundi. Zbog toga udaljenost mikrofona od izvora zvuka može uneti značajno kašnjenje zvuka, naročito u slučaju sportskih priredbi i snimanja na otvorenom, kada je udaljenost izvora zvuka i mikrofona velika. U slučaju sportske priredbe poput teniskog meča, kašnjenje zvuka izazvano udaljenošću mikrofona od izvora zvuka, iznosi oko 50ms. U slučaju postavke mikrofona na stolu ispred govornika, kašnjenje zvuka je oko 1 ms [2].

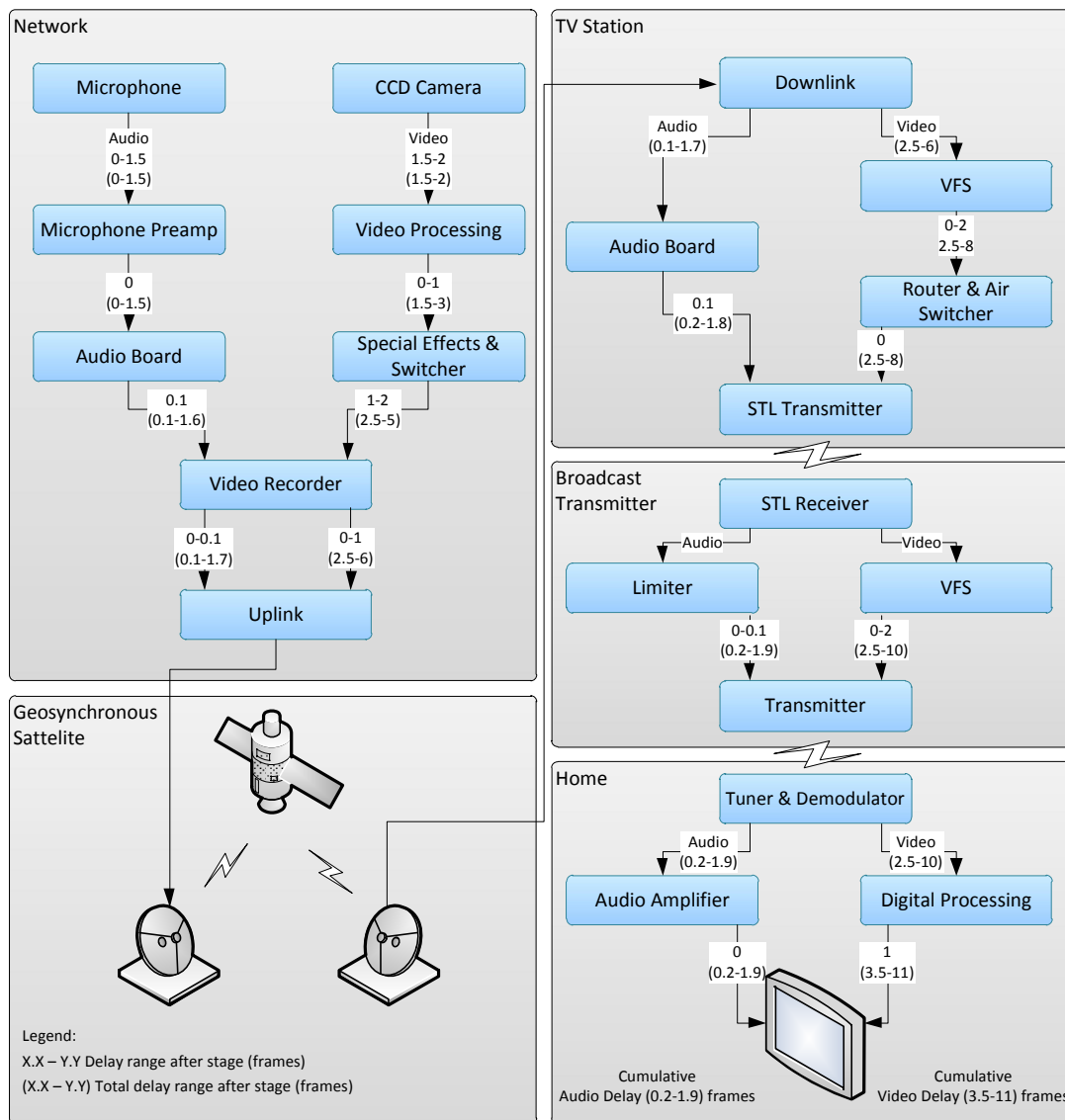
Povoljna okolnost je što u najvećem broju slučajeva, gledalac ne doživljava kašnjenje nastalo udaljenošću mikrofona kao problem. Kada je gledalac prisutan u istom okruženju gde se nalazi mikrofona, on dobija slična kašnjenja zvuka u odnosu na sliku kao posledicu različite brzine prostiranja talasa. Posmatrač sportske priredbe će svesno ili podsvesno osetiti kašnjenje zvuka za slikom, a kako je ovo prirodna pojava, gledalac televizije će i kašnjenje audio signala podsvesno vezati za pređašnje iskustvo. Međutim dodatak čak i male količine kašnjenja će učiniti normalno kašnjenje zvuka prevelikim i time napornim za gledaoca. Obrnut slučaj, to jest kada slika kasni za zvukom, je neprirodan i samim tim izaziva neprijatan za gledaoca [2]. Nakon pretvaranja zvuka u električni signal, obrada audio signala je relativno jednostavna te unosi malo kašnjenje u poređenju sa obradom video signala. Prenosni kanal za prenos zvuka, zavisno od složenosti i korišćene tehnologije, unosi dodatno kašnjenje od 1 ms do 2 ms po uređaju [2]. Ipak, ukupno kašnjenje zvuka retko prelazi par desetina milisekundi.

Zbog daleko veće brzine prostiranja svetlosti u odnosu na zvuk kašnjenje slike prouzrokovano udaljenošću sočiva se može zanemariti. Svetlost prelazi jedan metar za približno 3.3 ns, pa čak i u najekstremnijim slučajevima ne utiče bitno na krajnji rezultat.

Kašnjenje slike i zvuka najčešće nije posledica ograničene brzine prostiranja talasa. Najveći deo kašnjenja se unosi u audio-video lancu koji se prostire od studija do korisnika.

Svaka komponenta ovog lanca unosi kašnjenje, što je naročito izraženo kod zahtevnih obrada video signala.

Na slici **Error! Reference source not found.** je dat prikaz unetog kašnjenja za svaku komponentu tipičnog audio-video lanca. U zagradi je dat kumulativni zbir kašnjenja od početka do date komponente.



Slika 1 – Uneto kašnjenje tipičnih komponenti u AV lancu [2]

Bitno je primetiti da ukupno kašnjenje videa može biti čak do 10 slika veće od kašnjenja audia, kao posledica većeg kašnjenja koje unosi video oprema. Zbog toga se vrše korekcije vremena reprodukcije zvuka kako bi se video i audio isporučili u pravo vreme. Takođe je značajno što se deo kašnjenja javlja nakon produkcije, zbog čega inženjeri u studiju ne mogu da ga primete i koriguju. Preporuka ITU BT.1377 se odnosi na obeležavanje uređaja koji se koriste u audio-video lancu. Preporučuje se jasno isticanje kašnjenja u milisekundama koje uređaj unosi, a ukoliko je kašnjenje promenljivo, preporučuje se navođenje opsega [5][4].

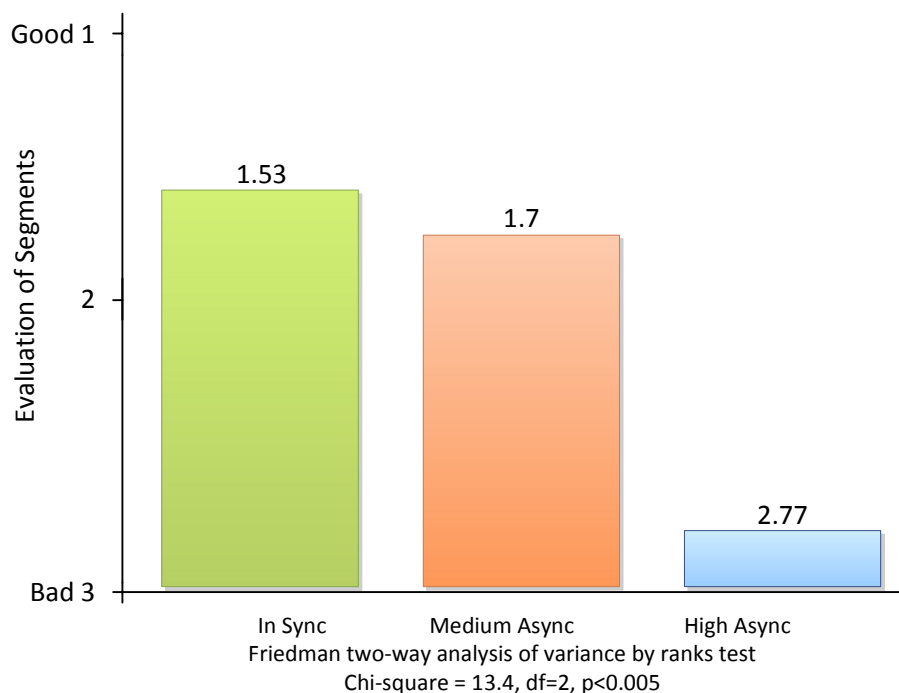
2.2 Značaj pravilne sinhronizacije

Pogrešna audio-video sinhronizacija dovodi do toga da pokreti usana i govor nisu sinhronizovani (eng. *lip sync*). Ova pojava naročito loše utiče na razumevanje govora kod osoba sa oštećenim sluhom. Čak i kod osoba koje nisu u stanju da “čitaju sa usana” loša sinhronizacija izaziva nelagodu i smanjuje razumljivost govora posredstvom Mekgurkovog efekta (eng. *McGurk effect*, po škotskom psihologu Hariju Mekgurku). Pored očiglednog negativnog uticaja na zadovoljstvo gledaoca, greška sinhronizacije uzrokuje i podsvestan efekat kod gledalaca [3].

Istraživanja sprovedena na Američkom univerzitetu Stenford se bavi efektima audio-video sinhronizacije na subjektivan doživljaj gledalaca [3]. Gledaocima su predstavljani televizijski isečci gde:

- su slika i zvuk sinhronizovani
- slika kasni za zvukom 2.5 polu-slika (eng. *fields*)
- slika kasni za zvukom 5 polu-slika

Istraživanje je pokazalo da u slučaju kad slika kasni za zvukom 5 polu-slika gledaoci doživljavaju ljude na video snimcima kao manje interesantne, manje uspešne, manje prijatne...



Slika 2 – Uticaj AV sinhronizacije na subjektivne ocene gledalaca[3]

Gledaoci su u stanju da procene kada su slika i zvuk sinhronizovani kao i kada video kasni za zvukom 5 poluslika. Gledaoci ne primećuju grešku sinhronizacije kada video kasni 2.5 polu-slike. Ipak, istraživanje je pokazalo da su ocene učesnika o ljudima na video snimcima negativnije čak i kada je greška sinhronizacije 2.5 polu-slike [3].

2.3 Međunarodna regulativa

Mnoge međunarodne organizacije izdale su veliki broj preporuka koji se tiču sinhronizacije audio-video signala kao i kvaliteta videa uopšte. Najznačajnije su preporuke organizacija ITU, EBU i ATSC.

ITU (eng. *International Telecommunication Union*) je specijalizovana agencija Ujedinjenih nacija odgovorna za informacione i komunikacione tehnologije. ITU raspoređuje globalni radio spektra, satelitske orbite i razvija međunarodne standarde u oblasti telekomunikacija.

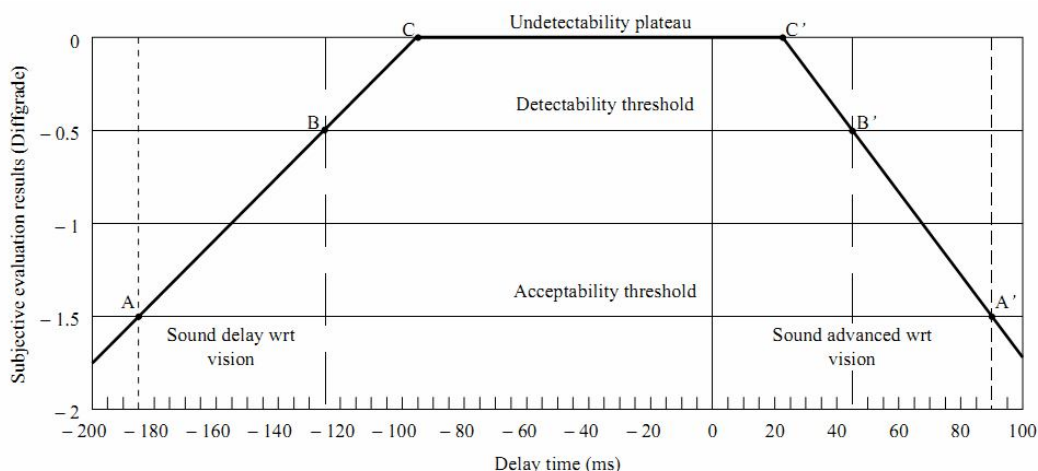
EBU (eng. *European Broadcasting Union*) je udruženje 85 radio difuznih organizacija iz 56 zemalja Evrope i Sredozemlja. Članice EBU su radio i televizijske kompanije od kojih su većina radiodifuzne organizacije u državnom vlasništvu ili u privatnom vlasništvu sa javnom koncesijom.

ATSC (eng. *Advanced Television Systems Committee*) je internacionalna neprofitna organizacija koja razvija standarde za digitalnu televiziju. ATSC je razvila istoimeni standard za digitalnu televiziju u Sjedinjenim Američkim Državama, koji su prihvatili i Kanada, Meksiko, Južna Koreja i Honduras.

Međunarodne regulativne agencije izdale su veliki broj preporuka koje se tiču audio-video sinhronizacije. Najznačajnije preporuke iz ove oblasti su ITU BT.1359 [4], ITU BT.1377 [5], ATSC IS-191 [6] i EBU R37-2007 [7] iz 1998. godine.

2.3.1 Preporuke

U preporuci ITU BT.1359-1 iz 1998. objavljeni su rezultati istraživanja osetljivosti gledalaca na grešku audio-video sinhronizacije sprovedenih u Japanu, Švajcarskoj i Austriji u periodu od 1993. do 1995. Istraživanje je pokazalo da gledaoci nisu u stanju da pouzdano otkriju grešku sinhronizacije kada video kasni 45ms ili kada audio kasni 125ms. Isti dokument kao granicu prihvatljivosti navodi opseg od +90ms do -125ms. Negativna vrednost znači da zvuk kasni za slikom, a pozitivna da slika kasni za zvukom. Pragovi detekcije i prihvatljivosti dati su na slici Slika 3.



Slika 3 – Nalazi istraživanja objavljeni u ITU BT.1359-1 [3]

Zbog neizvesnosti u pogledu sinhronizacije izvornog signala ITU preporučuje da se kašnjenje od tačke odabira izvora signala do emisionog predajnika bude u opsegu od +22.5 ms do -30ms. Treba napomenuti da su istraživanja na kojima se bazira ova preporuka rađena početkom 90-ih godina prošlog veka sa CRT monitorima i televizijskom slikom SD kvaliteta. Zbog napretka tehnologije opseg kašnjenja dat u BT.1359 može biti neadekvatan za primenu u digitalnoj televiziji [6] zbog prevelikog opsega dozvoljene greške sinhronizacije.

Dokument EBU R37-2007 preporučuje organizacijama članicama da, kada god je to moguće, rade na smanjenju greške audio-video sinhronizacije televizijskog signala. Preporučuje se upotreba tehnika za korekciju u svakoj tački audio-video lanca gde je greška sinhronizacije izvesna. Takođe se navodi da greška sinhronizacije u svakoj tački lanca treba da bude u granicama od +5ms do -15ms. Pored živog televizijskog signala zadate granice važe i kod emitovanja sa medija kao i u slučaju snimanja na medijum. Preporučuje se da kod signala namenjenog za emitovanje greška sinhronizacije na ulazu emisionog predajnika bude u granicama od +40ms do -60ms.

Dokument ATSC IS-191 ističe da pod svim operacionim okolnostima na ulazu DTV uređaja zvuk i slika bi trebalo da budu sinhronizovani. Zahteva se greška sinhronizacije u granici od +15ms do -45ms na ulazu prijemnika digitalne televizije. Dokument takođe preporučuje da se pri izgradnji audio-video lanca teži nultoj greški sinhronizacije između čvorova. Implementacioni komitet ATSC nalazi preporuku ITU B.1359 neadekvatnom za primenu u digitalnoj televiziji [6].

Document	Year of publication	Interstage delay	Delay at broadcast transmitter	Delay at viewer side
ITU BT.1359-1	1998	n/a	+22.5ms / -30ms	+90ms/-185ms ¹
EBU R37-2007	2007	+5ms/-15ms	+40ms / -60 ms	n/a
ATSC IS-191	2003	0ms ²	+15 ms / -45 ms	n/a

Tabela 1 – Preporuke međunarodnih organizacija

2.4 Ciljna platforma

Za vreme istraživanja dostupni personalni računari zasnovani na x86 platformi bili su uglavnom zasnovani na dvojezgarnim procesorima. Računari tog tipa nemaju dovoljne performanse za obradu dva audio-video toka HD (eng. *HD - High Definition*) kvaliteta. Zbog toga je izabrana arhitektura zasnovana na IBM Cell BE procesoru koja nudi dovoljne performanse.

Cell BE (eng. *Cell Broadband Engine, Cell/BE*) je mikroprocesorska arhitektura zajednički razvijena od strane STI alijanse koju čine Sony, Toshiba i IBM. Cell kombinuje PowerPC arhitekturu opšte namene skromnih performansi sa protočnim koprosorskim elementima što značajno ubrzava multimedijalne, aplikacije sa vektorskom obradom kao i mnoge druge forme namenskog računanja.

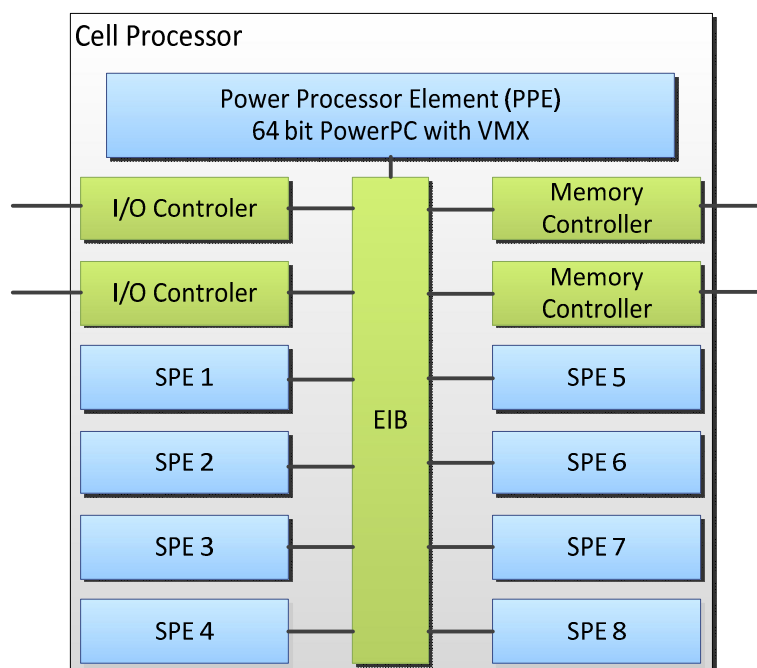
¹ acceptability threshold

² designers should strive for 0ms

Cell procesor je prvi put komercijalno predstavljen u *Sony Playstation 3* igračkoj konzoli 2006 godine. Ovaj procesor se takođe koristi u IBM-ovoj *QS* familiji servera, Sony *ZEGO* serverima, a dostupan je i kao akceleratora *PCI Express* kartica. Arhitektura se pokazala pogodnom za digitalne distribucione sisteme, sisteme za prikaz i snimanje HDTV i računarske sisteme za zabavu. Zbog svoje jedinstvene arhitekture pogodan je za upotrebu u digitalnoj obradi signala, fizičkim simulacijama, naučne, medicinske svrhe i slično [8].

Uprošćeno, arhitektura Cell procesora se može podeliti na četiri komponente:

- glavni procesor PPE (eng. *Power Processing Element*)
- 8 SPE (eng. *Synergistic Processing Elements*) koprocesora
- ulazno-izlazne strukture
- specijalizovanu cirkularnu magistralu EIB (eng. *Element Interconnect Bus*) visoke propusne moći koja povezuje sve ostale elemente

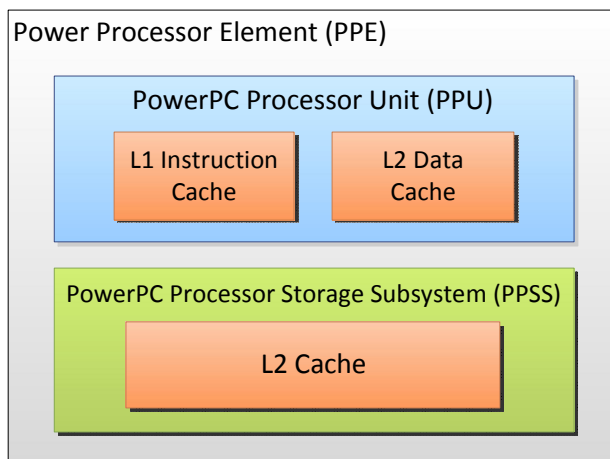


Slika 4 – Unutrašnja arhitektura Cell/BE mikroprocesora

2.4.1 PPE

PPE je kompatibilan sa 64-bitnom implementacijom *PowerPC* arhitekture. U stanju je da izvršava dve niti istovremeno, sadrži jedinicu za rad sa celim brojevima, jedinicu za rad sa pokretnim zarezom, jedinicu za rad sa vektorskim multimedijalnim ekstenzijama (eng. *Vector Multimedia Unit - VMX*) i jedinicu za upravljanje memorijom (eng. *Memory Management Unit - MMU*). Posедуje 64KB keš memorije prvog nivoa (eng. *L1 cache*) po 32KB za instrukcije i podatke, 512KB keš memorije drugog nivoa (eng. *L2 cache*). Širina keš linije je 128 bita. PPE takođe sadrži i AltiVec protočnu jedinicu za rad sa pokretnim zarezom u jednostrukoj preciznosti.

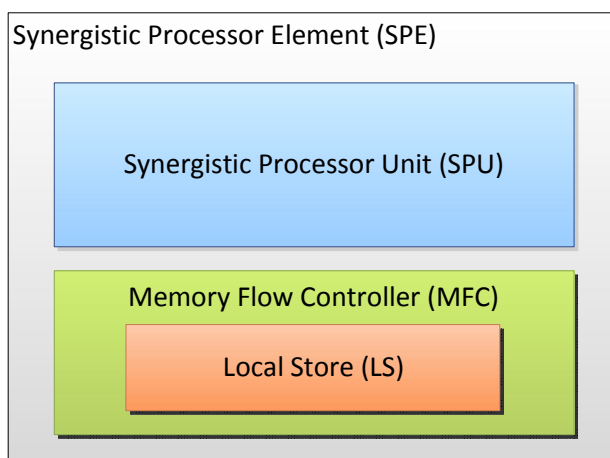
Bitan deo PPE je i kontroler memorijskog toka (eng. *MFC - Memory Flow Controller*) koji omogućava direktan memorijski pristup (eng. *DMA - direct memory access*) SPE procesorima kao i ulazu/izlazu (eng. *I/O - Input/Output*). Komunikacija sa SPE je moguća putem DMA i sistema poštanskih sandučića (eng. *Mailboxes*).



Slika 5 – Komponente PPE procesora

2.4.2 SPE

SPE koprocesor se sastoji od SPU (eng. *Synergistic Processing Unit*) i MFC (eng. *Memory Flow Controller*) memorijskog kontrolera koji sadrži DMA, MMU i magistralu. SPE je procesor sa smanjenim setom instrukcija (eng. *RISC - Reduced instruction set computing*) i 128-bitnom SIMD organizacijom za instrukcije u jednostrukoj i dvostrukoj preciznosti. Svaki SPE poseduje i 256KB ugrađene SRAM memorije (eng. *Static random access memory*) za instrukcije i podatke koja se naziva lokalno skladište (eng. *LS - Local Storage*). Bitno je napomenuti da LS nije uobičajena keš memorija jer njena upotreba nije transparentna za softver, niti sadrži hardverske strukture koje omogućavaju predikciju učitavanja. SPE sadrži niz od 128 128-bitnih registara u koji može da stane 16 8-bitnih celih brojeva, 8 16-bitnih celih brojeva, 4 32-bitna cela broja ili 4 broja u pokretnom zarezu jednostruke preciznosti (po jednom registru). Treba napomenuti da SPU nema direktan pristup sistemskoj memoriji već se 64-bitne memorijske adrese u SPU moraju proslediti SPE memorijskom kontroleru (MFC) da izvrši DMA operaciju u sistemskom adresnom prostoru [8][9]. MFC radi nezavisno od ostatka SPU pa se DMA prenos obavlja bez zaustavljanja izvršenja SPE programa [9].

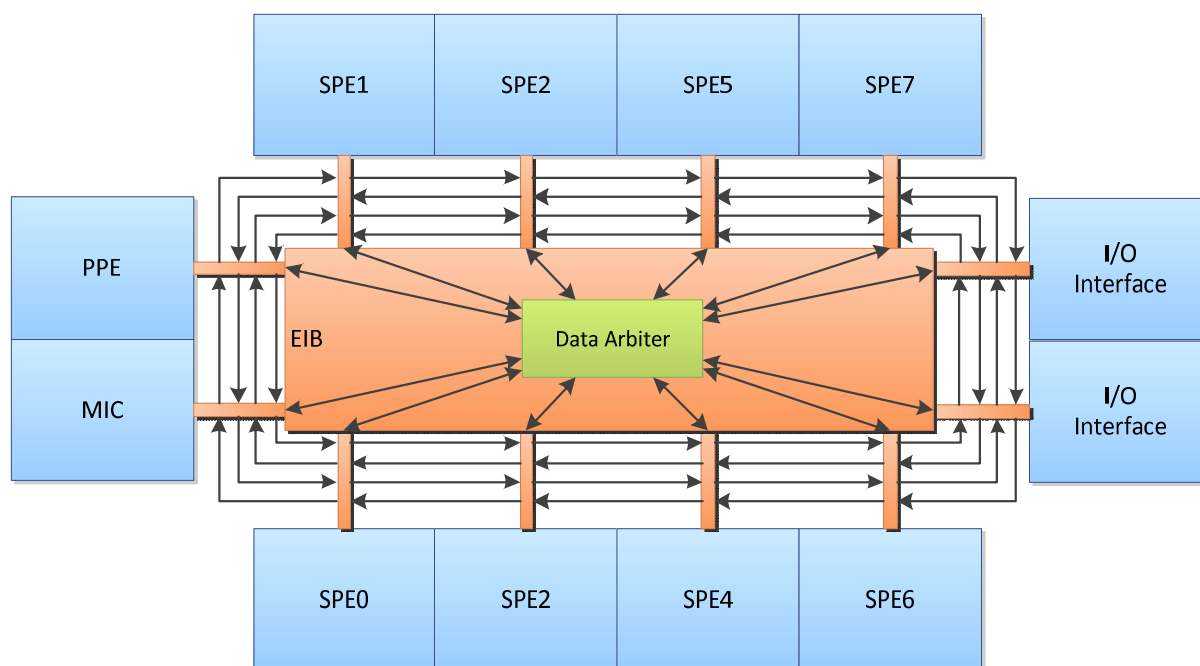


Slika 6 – Komponente SPE procesora

2.4.3 EIB magistrala

EIB je komunikaciona magistrala Cell procesora koja povezuje različite elemente: PPU, memorijski kontroler, 8 SPU jezgara kao i dva ulazno-izlazna sprežna sistema van procesora. EIB sadrži jedinicu za arbitražu koja funkcioniše kao skup semafora. Magistrala je izvedena kao kružni prsten koji se sastoji od 4 jednosmernih kanala, širine 16 bajta, koji se rotiraju u paru. Svaki učesnik na EIB magistrali ima po jedan port za čitanje i pisanje širine 16 bajta. Ograničenje za svakog učesnika je čitanje i pisanje 16 bajta na svaka dva takta, odnosno 8 bajta po taktu. Svaki SPU sadrži i namenski DMA red sposoban da zakazuje sekvence prenosa na različite tačke bez prekida računanja.

Tok podataka na EIB kanalu je koračni oko prstena. Obzirom da postoji dvanaest učesnika, ukupan broj koraka oko kanala je dvanaest. Šest koraka je najduža distanca između bilo koja dva učesnika to jest nije moguće izvesti prenos podataka koji zahteva više od šest koraka zato što postoji kraća ruta. Dužina distance između učesnika određuje ukupne performanse EIB magistrale jer smanjuje konkurentnost. Prsten može započeti novu operaciju na svaka tri ciklusa, svaki transfer traje 8 taktova. Ova osobina ga čini podesnim za prenos velikih količina podataka, u slučaju da se vrši veliki broj malih transfera, EIB performanse opadaju. U praksi je redak slučaj da je propusnost EIB magistrale ograničavajući faktor performansi Cell procesora. U najgorem slučaju programer mora planirati obrasce komunikacije tako da se EIB koristi sa visokim nivoom konkurentnosti [8]. EIB magistrala radi na taktu upola manjem od sistemskog.



Slika 7 – Element Interconnect Bus

2.4.4 Memorijski i U/I kontroleri

Cell sadrži dvokanalni Rambus XIO interfejs ka Rambus XDR memoriji. Memorijski kontroler MIC (eng. *Memory Interface Controller*) je odvojen od XIO. XIO-XDR veza ima

propusnost od 3.2Gbit/s po pinu. Dva 32-bitna kanala omogućavaju teoretski maksimum od 25.6 GB/s.

Ulazno izlazni sprežni sistem se naziva FlexIO. FlexIO sprežni sistem je organizovan u 12 linija, gde je svaka linija jednosmerna putanja od tačke do tačke širine 8 bita. Pet 8-bitnih putanja su ulazne ka Cell-u, dok je preostalih 7 izlaznih. Ovo omogućuje teoretski maksimum od 62.4 GB/s (36.4 GB/s izlazno i 26 GB/s ulazno) pri podrazumevanom taktu od 2.6 Ghz. FlexIO sprežni sistem ima takt nezavisan od ostaka procesora, tipično 3.2Ghz [8].

2.4.5 Performanse

Da bi se razumele mogućnosti i načini optimizacije kod Cell procesora, neophodno je razmotriti jedinstvene osobine ove arhitekture naročito u sferama gde se one razlikuju od konvencionalnih procesora. Za razliku od homogenih višejezgarnih procesora gde se procesor sastoji od više kopija istovetnih jezgara, Cell arhitektura je heterogena. Heterogene arhitekture se sastoje od jezgara različitog tipa, u slučaju Cell-a to su PPE i 8 SPE jezgara. Cell se od konvencionalnih procesora razlikuje i po nameni: projektovan je za protočna opterećenja (eng. *streaming workload*). To u praksi znači da SPE jezgra rade najveći deo posla paralelnom ili protočnom obradom dok PPE služi za pripremu podataka, sinhronizaciju zadataka i izvršenje delova programskog koda koje nije moguće paralelizovati. Zbog specifičnog dizajna, performanse Cell procesora sa zadacima opšteg tipa, koji nisu prilagođeni da koriste SPE jezgra, su slične performansama PowerPC jezgra koje radi na istom taktu kao PPE.

Umesto keš memorije (eng. *cache memory*) koja se koristi kod konvencionalnih procesora, SPE jezgra sadrže 256KB lokalne memorije (eng. LS - local store). Direktni pristup sistemskoj memoriji (eng. *RAM – Random Access Memory*) nije moguć. Umesto toga SPE koristi eksplicitne asinhronne DMA transfere iz RAM-a u LS. Pošto SPE nema pristup nijednom drugom tipu memorije osim lokalne, LS mora sadržati i programski kod koji izvršava i podatke. SPE jezgra imaju smanjen set instrukcija sličan konvencionalnim RISC (eng. *Reduced Instruction Set Computing*) procesorima. Svaka SPE instrukcija je širine 32 bita sa fiksnim vremenom izvršenja od 2 do 7 ciklusa (ovde nisu uzete u obzir instrukcije za rad sa pokretnim zarezom u dvostruko preciznosti). U poređenju sa standardnim procesorima SPU ima veliko registarsko polje od čak 128 128-bitnih SIMD registara (eng. *Single Instruction Multiple Data*). Većina SPU instrukcija su SIMD, namenjene za izvršenje multimedijalnih i zadataka sa pokretnim zarezom. SPU instrukcijski set je fleksibilniji od npr. Intel-ovog SSE/SSE2/SSE3 instrukcijskog seta i uključuje SIMD dodaj-pomnoži instrukcije sa 3 operanda u pokretnom zrezu koje se izvršavaju u jednom ciklusu i omogućavaju teorijski maksimum od 25.6 GFlops (eng. Giga Floating-point operations per second) na SPU koji radi na 3.2Ghz. Svaki SPU ima dve protočne strukture (eng. pipeline) specijalizovane za različit tip instrukcija: učitaj/snimi i aritmetičke. Ovakav pristup omogućava izvršenje dve (nezavisne) instrukcije u paraleli sa teoretskim maksimumom od 6,4 milijarde SIMD instrukcija u sekundi po jednom SPU jezgru. Pored posebnog skupa instrukcija SPE nema predviđanje grananja (eng. branch prediction) poput standardnih procesora. Umesto toga dostupna je posebna instrukcija za nagoveštaj grananja koja pomaže SPE da predvidi da li će određena grana programskog koda biti izvršena.

PPE i SPE izvršavaju programski kod *po redosledu* (eng. in-order execution) nasuprot izvršenju *van redosleda* (eng. out-of-order execution) kod savremenih konvencionalnih procesora. Kako 256KB lokalne memorije kod SPE ima fiksno kašnjenje (eng. *latency*) od sedam ciklusa, *in-order execution* je zgodno pojednostavljenje jer prevodilac (eng. *compiler*)

može da predvidi pristup memoriji i generiše instrukcije tako da maksimizuje performanse. Sa druge strane *in-order execution* PPE jezgra sa samo 32 KB keš memorije prvog i 512KB drugog nivoa usporava izvršenje koda sa čestim pristupanjem memoriji. To ga čini sporijim u odnosu na sličan procesor sa *out-of-order* izvršenjem instrukcija. Da bi se smanjio uticaj keš promašaja (eng. *cache misses*) PPE koristi tehnologiju za izvršenje dve niti na jednom jezgru sličan Intel *Hyperthreading* tehnologiji [11].

Za prenos podataka i efikasnu komunikaciju između SPE jezgara koristi se brza EIB magistrala. Uređaji na EIB-u mogu istovremeno primati i slati 16 bajta podataka po jednom taktu magistrale. Dakle, 8 SPE procesora sa četiri konkurentna transfera na svakom od četiri komunikaciona prstena mogu preneti 128 bajta podataka. Takt EIB magistrale je upola manji od glavnog takta procesora. To znači da je kod Cell procesora koji radi na 3.2GHz najveća teoretska širina propusnog opsega magistrale $128B \times 1.6GHz = 204.6 \text{ GB/s}$. Održiva brzina prenosa podataka je niža od teoretskog maksimuma i zavisi od položaja izvora i odredišta.

Smanjena brzina prenosa se dešava u slučaju kada:

- se pristupa istom odredištu u isto vreme
- su svi transferi u istom smeru, pa su dva kanala četvorokanalnog prstena u stanju mirovanja jer se prenos uvek vrši najkraćom putanjom
- se zahteva veliki broj penosa malih paketa podataka pa se umanjuje iskorišćenost magistrale
- višekoračni prenosi onemogućavaju istovremene prenose (na istom prstenu) sa lokacija koji se nalaze na njihovom putu.

Iz nabrojanih razloga očigledno je da na maksimalne održive brzine prenosa utiče ne samo međusobni položaj odredišta već i položaj ostalih prenosa na magistrali. U IBM-ovim laboratorijama izmerena je ukupna brzina četiri simultana prenosa podataka putem EIB magistrale [9]. Korišćen je Cell BE procesor sa taktom od 3.2GHz, rezultati merenja dati su u Tabela 2.

Test configuration	EIB @ 3.2 GHz
SPE1 <-> SPE3, SPE5 <-> SPE7, SPE0 <-> SPE2, SPE4 <-> SPE6	186 GB/s
SPE0 <-> SPE4, SPE1 <-> SPE5, SPE2 <-> SPE6, SPE3 <-> SPE7	197 GB/s
SPE0 <-> SPE1, SPE2 <-> SPE3, SPE4 <-> SPE5, SPE6 <-> SPE7	197 GB/s
SPE0 <-> SPE3, SPE1 <-> SPE2, SPE4 <-> SPE7, SPE5 <-> SPE6	197 GB/s
SPE0 <-> SPE7, SPE1 <-> SPE6, SPE2 <-> SPE5, SPE3 <-> SPE4	78 GB/s
SPE0 <-> SPE5, SPE1 <-> SPE4, SPE2 <-> SPE7, SPE3 <-> SPE6	95 GB/s
SPE0 <-> SPE6, SPE1 <-> SPE7, SPE2 <-> SPE4, SPE3 <-> SPE5	197 GB/s

Tabela 2 – Realna brzina prenosa podataka između SPE jezgara

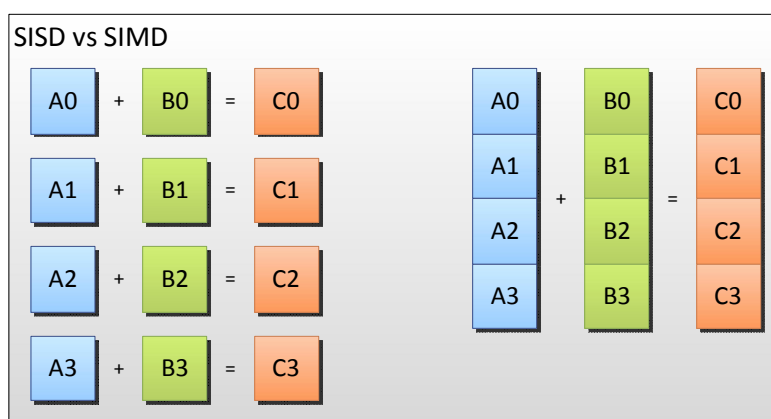
Minimalistički dizajn SPE jezgara, koji podrazumeva odsustvo keš memorije i predviđanja grananja kao i izvršavanje po redosledu, omogućava rad na višem taktu u odnosu na konvencionalne procesore [11].

2.4.6 Optimizacija SPE koda

Svoje visoke performanse Cell procesor najviše duguje procesorskoj snazi SPE jezgara. Da bi se obezbedile visoke performanse bitno je na najbolji način iskoristiti

osobnosti fizičke arhitekture SPE jezgara. SPE programiranje podrazumeva više od pukog prevođenja koda za SPE. SPE prevodioci nisu u stanju da iskoriste sve pogodnosti vektorskog programiranja pri optimizaciji pa je potreban dodatan napor programera za postizanje vrhunskih performansi.

Najočiglednija optimizacija je vektorizacija koda koja se postiže upotrebom SIMD (eng. *Single Instruction Multiple Data* - jedna instrukcija, više podataka). Na SPU procesorima SIMD instrukcije se mogu izvršavati nad registrima koji sadrže vektore podataka. Svaki 128-bitni registar može sadržati 16 nezavisnih bajtova, odnosno 8 polureči, 4 reči i tako dalje. Ova osobina je naročito značajna u digitalnoj obradi signala jer su odbirci najčešće veličine jednog bajta u slučaju slike i dva bajta u slučaju zvuka. Vektorizacijom je dakle moguće izvršiti instrukciju nad 16 video odbiraka odjednom.



Slika 8 – SISD i SIMD instrukcije

Ipak SIMD instrukcije izvršavaju se nad celim vektorom pa pojava grananja zavisnog od vrednosti podataka predstavlja problem. U tom slučaju bi se morale odvojiti putanje izvršavanja za podatke koji ispunjavaju uslov grananja od onih koji ne ispunjavaju uslov. Ovaj slučaj je teško vektorizovati i zato se pristupa eliminaciji grananja. Eliminacija grananja predstavlja tehniku u kojoj se isti tok instrukcija koristi za podatke bez obzira da li ispunjavaju uslov grananja ili ne. To se postiže upotrebom instrukcija *selbm*, *shufb* i bitskih operatora. Ovakvim pristupom programski kod računa vrednosti za obe grane i korišćenjem nabrojanih instrukcija bira koja će vrednost biti dodeljena. Značaj eliminacije grananja je u tome što omogućava vektorizaciju koda jer nakon nje svi podaci iz vektora prođu kroz isti niz instrukcija.

Treba imati u vidu da se DSP (eng. *Digital Signal Processing*) programski kod često izvršava u programskim petljama. Programska petlja se sastoji od uslova i niza instrukcija koji se izvršava sve dok je uslov ispunjen. Provera uslova se može smatrati grananjem i zbog toga se nastoji smanjiti njihov broj u cilju postizanja boljih performansi. Tehnika koja se koristi za tu namenu naziva se razmatavanje petlje (eng. *loop unrolling*, *loop unwinding*) i zasniva se na povećanju niza korisnih instrukcija uz istovremeno smanjenje broja uslova.

Neke instrukcije zahtevaju više od jednog procesorskog ciklusa za izvršenje, zbog toga je moguće da redosled instrukcija utiče na brzinu izvršavanja. Upotrebom protočne strukture (eng. *pipeline*) procesora se postiže da sledeća instrukcija počne da se izvršava pre nego što prethodna završi. Tehnika kojom se instrukcije postavljaju tako da se u potpunosti iskoristi procesorska protočna struktura naziva se "zakazivanje instrukcija" (eng. *instruction scheduling*). Sledeći termini su bitni za razumevanje ove tehnike:

1. kašnjenje (eng. *latency*) - broj ciklusa potreban se instrukcija u potpunosti izvrši. Jednak je broju nivoa koje instrukcija zauzima u protočnoj strukturi.

2. zastoj (eng. *stall*) - ciklus u kom procesor ne započinje izvršavanje nove instrukcije
3. uslovni zastoj (eng. *dependency stall*) - zastoj koji nastaje kad operand sledeće instrukcije zahteva vrednost iz instrukcije koja još nije u potpunosti završena.

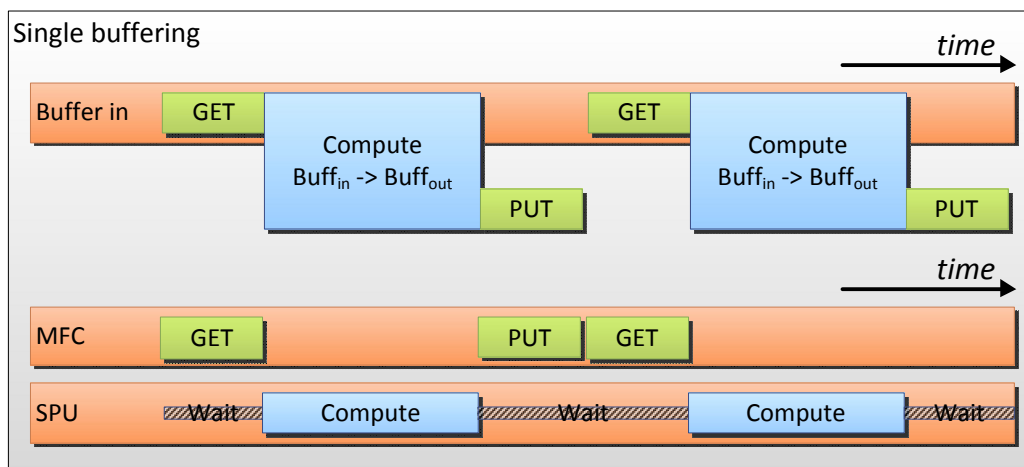
Optimizacija SPU koda uglavnom podrazumeva eliminisanje uslovnih zastoja. SPU poseduje dve protočne strukture: protočnu strukturu *pipeline0* (parnu ili izvršnu) i *pipeline1* (neparnu ili učitaj/snimi strukturu). Protočne strukture se razlikuju po tipu instrukcija koje izvršavaju, što je prikazano u **Tabela 3**. SPU učitava po dve instrukcije odjednom i ukoliko jedna pripada parnoj a druga neparnoj protočnoj strukturi moguće ih je istovremeno započeti. To praktično znači da ne samo da SPU može izvršavati više instrukcija u protočnoj strukturi već može i započeti dve nove u istom ciklusu. Ukoliko nisu zadovoljeni uslovi za istovremeno pokretanje dve instrukcije ili ukoliko su instrukcije međsobno zavisne (postoji uslovni zastoj) instrukcije se započinju u odvojenim ciklusima procesora. Programeru su na raspolaganju dve instrukcije za poravnavanje instrukcija kako *nop* (eng. *no instruction on even pipeline* - bez instrukcije na parnoj protočnoj strukturi) i *lnop* (eng. *no instruction on odd pipeline* - bez instrukcije na neparnoj protočnoj strukturi).

Pipeline0 Instructions	Stall (clocks)	Latency (clocks)	Pipeline1 Instructions	Stall (clocks)	Latency (clocks)
Single precision floating-point operations	0	6	Suffle bytes, quadword rotates and shifts	0	4
Single precision, convert between float/integer, interpolate	0	7	Gather, mask, generate insertion control	0	4
Immediate loads, logical ops, integer add/subtract, signed extend, count leading zeros, select bits, carry/borrow generate	0	2	Estimate	0	4
Double precision floating-point operations	6	7	Loads/stores, branch hints	0	6
Element rotates and shifts	0	4	Branches	0	4
Byte operations, count ones, abs difference, average, sum	0	4	Channel operations, move to/from SPRs	0	6

Tabela 3 – Kašnjenje i instrukcija u protočnoj strukturi [22]

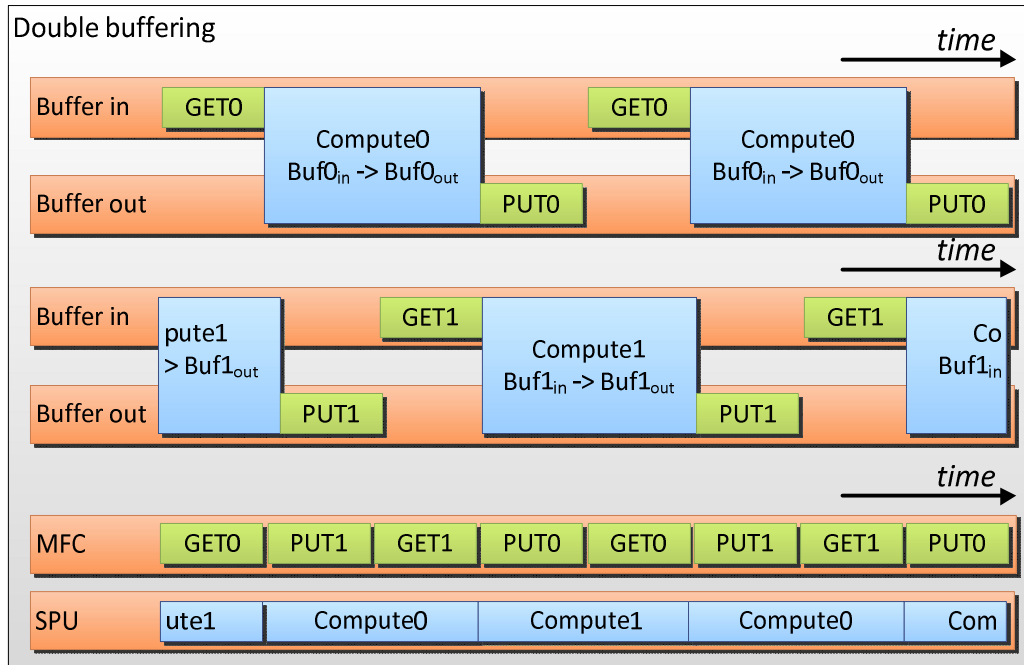
Kao što je ranije napomenuto SPU nema fizičku podršku za predikciju i nagoveštavanje (eng. *branch prediction*) grananja ali to nadoknađuje dobrom programskom podrškom. Nagoveštavanje grananja je potrebno SPU procesorima zbog velikog kašnjenja u slučaju odabira pogrešne grane programskog koda. Potrebno je čak 18 ciklusa za povratak u slučaju odabira pogrešne grane programskog koda. Dodatno zbog nepostojanja predikcije SPU procesor podrazumeva (ukoliko nije instruisan drugačije) da se skok pri grananju neće izvršiti uključujući pri tom i naredbe bezuslovnog skoka. Nagoveštavanje grananja govori procesoru verovatnu adresu (eng. *hint-trigger address*) instrukcije odakle će se izvršavanje

nastaviti. Ovaj mehanizam omogućava procesoru da se pripremi za nastavak izvršavanja nakon skoka dopremanjem (eng. *prefetch*) instrukcija. Nagoveštavanje grananja ni na koji način ne utiče na logičku interpretaciju koda već samo na broj ciklusa potreban za izvršenje.



Slika 9 – Ilustracija zauzeća procesora kod jednostruko spregnutog punjenja

Upotrebom liste DMA poziva moguće je zakazati više DMA transfera. Kako su DMA pozivi asinhroni moguće je posredstvom dvostrukog baferovanja (eng. *double buffering*) koristiti SPE za računanje dok se podaci prenose. Na ovaj način se mogu obezbediti bolje performanse jer se ne troši procesorsko vreme na čekanje da se završi kopiranje memorije. Ilustracije potrošnje procesorskog vremena za jednostruko i dvostruko spregnuto punjenje date su na slikama Slika 9 i Slika 10.



Slika 10 – Ilustracija zazeća procesora kod dvostruko spregnutog punjenja

2.4.7 Sony Playstation 3

Najjeftinija i najčešća upotreba Cell/BE procesora je u Sony *PlayStation 3* (skraćeno PS3) igračkoj konzoli. PS3 koristi Cell procesor na taktu od 3.2GHz koji se sastoji od jednog PPE jezgra i šest (od ukupno osam) dostupnih SPE jezgara. Sedmo SPE jezgro je rezervirano za *hypervisor* sloj za enkripciju dok je osmo jezgro isključeno da bi se smanjio procenat neispravnih procesora. Osim Cell procesora PS3 sadrži 256MB Rambus XDR memorije, NVIDIA RSX grafički procesor sa 256MB DDR3 memorije, *ethernet* mrežni adapter i 802.11b/g bežični (eng. *wireless*, *wifi*) adapter.

Upotreba PS3 u naučne svrhe je moguća jer je, pored redovnog GameOS operativnog sistema, moguće instalirati i neku od GNU/Linux distribucija kao “gostujući” operativni sistem. Na taj način se PS3 transformiše u GNU/Linux super-računar. Linux kernel je odvojen od fizičke arhitekture Hypervisor slojem za virtuelizaciju. Uređaji i sistemski resursi su virtuelizovani pa Linux drajveri rade sa virtuelnim uređajima.

Cell procesor u PS3 je identičan onim u IBM serverima, baziranim na Cell-u, osim što su dva SPE jezgra nedostupna. IBM Cell serveri, poput *QS20*, sadrže Cell čipove sa svih 8 ispravnih SPE jezgara, dok se u PS3 konzolu ugrađuju i čipovi sa jednim neispravnim SPE jezgrom koje se zatim isključuje.

PS3 sadrži NVIDIA RSX grafički procesor sa 256 MB sopstvene memorije. *hypervision* sistem ne dozvoljava pristup grafičkom procesoru tako da je njegova primena za numerička izračunavanja isključena.

Za razliku od PC arhitekture, PS3 Ethernet adapter nije povezan na sistemsku magistralu već direktno na procesorski čip. Linux koristi *hypervisor* pozive da pristupi Ethernet adapteru pomoću *glibc_net* rukovaoca (eng. *driver*). Mrežna sprega ima ugrađenu DMA jedinicu koja omogućava prenos podataka bez angažovanja (time i opterećenja) PPE jezgra.

2.4.8 Virtuelizacija, GuestOS

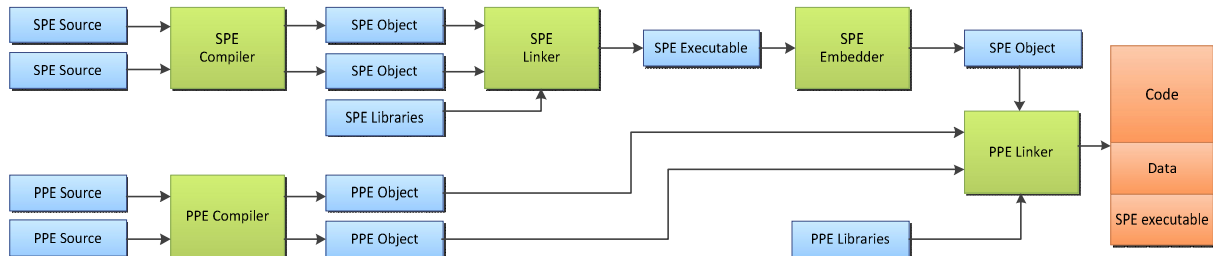
Linux se na PS3 pokreće iznad sloja za Hypervisor virtuelizaciju GameOS operativnog sistema. To znači da je fizička arhitektura dostupna samo kroz Hypervison pozive. Hardver komunicira sa kernelom pomoću sistema virtuelnih prekida (eng. *interrupts*). Virtuelni prekidi se koriste i za implementaciju *callback* funkcija za neblokirajuće systemske pozive. GameOS zauzima jedno SPE jezgro i kontroliše sav pristup fizičkoj arhitekturi poput grafičkog podsistema i mrežne kartice. Posledica toga su povećana kašnjenja u pristupu fizičkoj arhitekturi kao i potpuna nepristupačnost grafičkog čipa korisniku iz GNU/Linux OS.

2.4.9 Programski alati

Sve GNU/Linux distribucije za PS3 dolaze sa standardnim skupom GNU alata za prevođenje programskih jezika: gcc za C, g++ za C++ i gfortran za Fortran95. IBM SDK (eng. *Software Development Kit*) za Cell poseduje sličan set alata sa IBM XLC prevodicem za C/C++ i XLF prevodiocem za Fortran95. SDK se može instalirati direktno na računar baziran na Cell procesoru ili koristiti na x86 kompatibilnom računaru a prevoditi za Cell (eng. *Cross-compilation*).

Osnovni GNU set alata (eng. *toolchain*) za programiranje na Cell procesoru obuhvata:

- gcc – GNU prevodilac za Cell procesor, generiše PPC Linux binarne fajlove za PPE
- spu-gcc – GNU prevodilac za SPE procesore
- embedspu – Poseban alat za prevođenje SPE izvornog koda u objektne fajlove koji se mogu povezati (eng. *link*) sa PPE izvršnim fajlovima. Takođe generiše globalne promenljive koje se odnose na SPE procesore kako bi PPE mogao da učita izvršni kod na SPE procesore i pokrene ga.



Slika 11 – Prevođenje Cell izvornog koda

Pre pokretanja SPE programskog koda neophodno je učitati kod u lokalnu memoriju SPE jezgra. Ova operacija se vrši iz niti PPE programa funkcijom *spe_program_load()*. Pored ugrađivanja u PPE kod, SPE program može biti i nezavisna binarna datoteka u ELF formatu. U ovom slučaju potrebno je prvo učitati SPE program u sistemsku memoriju pozivom funkcije *spe_image_open()*.

Da bi se omogućilo istovremeno izvršavanje PPE i SPE programa SPE programski kod se poziva u programskoj niti PPE programa. U nastavku je dat primer programa koji ispisuje poruku iz SPE programskog koda. Program se sastoji od datoteka *ppe_example.c* sa PPE programskim kodom i *spe_example.c* sa SPE programskim kodom.

Sadržaj datoteke *ppe_example.c*:

```

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h> // For threads
#include <libspe2.h> // For SPE management (spe_*)
void *spe_test_function(void *data); // pthread callback
extern spe_program_handle_t test_handle; // Embedded SPE program

int main()
{
    pthread_t my_thread;
    // Create thread
    pthread_create(&my_thread, NULL, spe_test_function, NULL);
    // Join thread
    pthread_join(my_thread, NULL);
    return 0;
}

void *spe_test_function(void *data)
{
    int retval;
    unsigned int entry_point = SPE_DEFAULT_ENTRY;
    spe_context_ptr_t my_context;
    /* Create the SPE context */
    my_context = spe_context_create(SPE_EVENTS_ENABLE | SPE_MAP_PS, NULL);
    /* Load the embedded code into this context */
    spe_program_load(my_context, &test_handle);
    /* Run the SPE program until completion */
    do {
        retval = spe_context_run(my_context, &entry_point, 0, NULL, NULL, NULL);
    } while( retval > 0 );
    /* Destroy the context */
    spe_context_destroy(my_context);
}

```

```
pthread_exit(NULL);
}
```

Sadržaj datoteke *spe_example.c*:

```
#include <stdio.h>
int main(unsigned long long spe_id, unsigned long long pdata)
{
    printf("Hello, Cell!");
    return 0;
}
```

Pri prevođenju izvornog koda potrebno je prevesti SPU kod komandom `spu-gcc`:

```
spu-gcc -o spe_example spe_example.c
```

Zatim se SPU objektna datoteka prevodi u PPU format komandom `ppu-embedspu`:

```
ppu-embedspu test handle spe example spe example csf.o
```

Na kraju se prevodi PPU programski kod i povezuje sa objektom i bibliotekom `libspe2` komandom `ppu-gcc`:

```
ppu-gcc -o ppe example ppe example.c spe example csf.o -lspe2
```

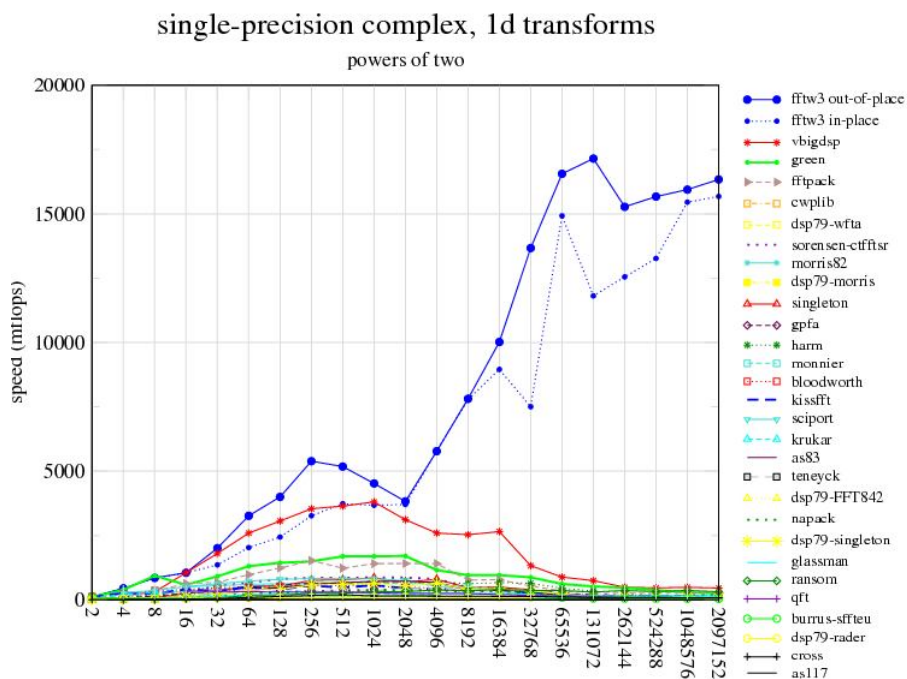
Rezultat rada programa nakon izvršenja izvršne datoteke:

```
./ppe_example
>> Hello, Cell!
```

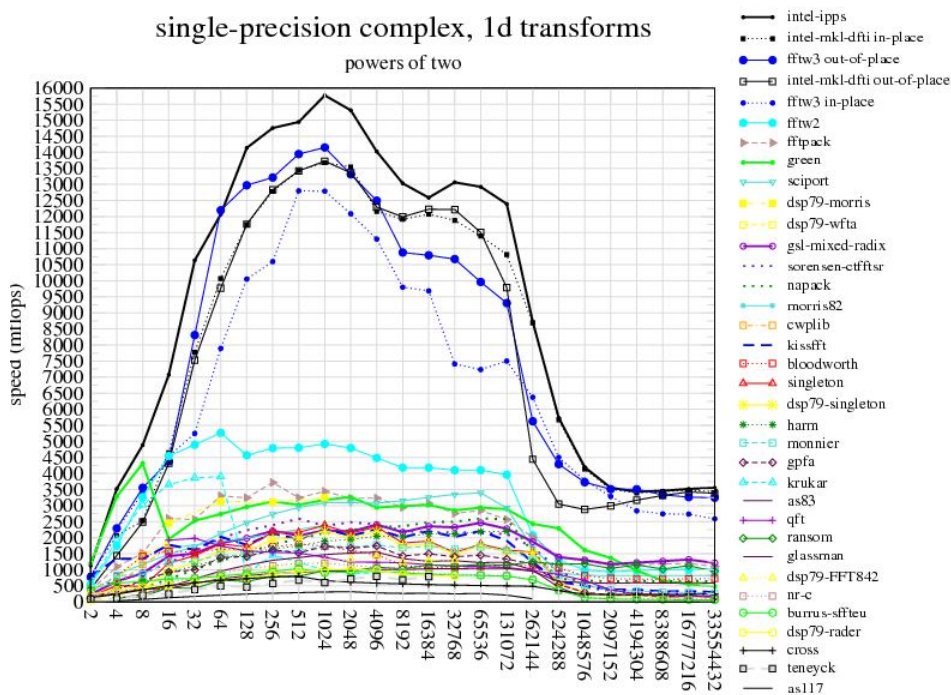
2.4.10 Primena

Zahvaljujući dostupnosti, visokim performansama i niskoj ceni PlayStation 3 se može smatrati najjeftinijim super-računarom. Specifična arhitektura SPE jezgara je posebno podesna za pojedine tipove namenskih izračunavanja u nauci. Distribuirani računarski sistemi koji se sastoje od velikog broja umreženih PS3 konzola su našli primenu u raznim naučnim disciplinama. Veliku zaslugu za ukupne performanse PS3 imaju brza XDR memorija i SPE jezgra. Njegova arhitektura je naročito pogodna za primenu paralelnog izvršavanja memorijski zahtevnog koda kao što je primena u obradi signala visoke definicije.

Na slici Slika 12 i slici Slika 13 dat je prikaz performansi PS3 i Intel Xeon Core Duo 3.0 GHz procesora pri računanju brze Fourierove transformacije. Performanse *fftw3* biblioteke na Cell procesoru su čak 4 puta bolje od Intel Xeon procesora kod transformacije nad velikim brojem odbiraka (više od 262144 odbrika) [23].



Slika 12 – FFT performanse PS3 3.2 GHz Cell B/E, 6 SPE jezgara [23]



Slika 13 – FFT performanse 3.0 GHz Intel Xeon Core Duo 64-bit [23]

3. Koncept rešenja

Ciljna platforma Sony Playstation 3 (PS3) igračka konzola zasnovana je na Cell/BE procesoru koji je zbog svoje arhitekture podesan za zahtevne DSP aplikacije. PlayStation3 poseduje 3 tipa ulazno-izlaznih uređaja USB 2.0, 802.11b/g Wireless adapter i 1000BASE-T mrežni adapter. Maksimalna teoretska brzina ovih uređaja je data u tabeli Tabela 4:

Device	Bitrate
IEEE 802.11g wireless device	54 Mbps
USB 2.0 port	480 Mbps
1000BASE-T Ethernet adapter	1000 Mbps

Tabela 4 – *Raspoloživi ulazno-izlazni uređaji*

Ulazno/izlazna funkcionalnost se ne može proširiti dodavanjem dodatnih uređaja. Gigabitni mrežni adapter, kao najbrži dostupni uređaj, nameće se kao najbolji izbor za prenos zahtevnog materijala poput multimedije.

Prenos zvuka i slike u digitalnom formatu je jako zahtevan u pogledu širine propusnog opsega. Potrebne širine propusnog opsega za najčešće video i audio formate u nekompresovanom obliku dati su u tabelama Tabela 5 i Tabela 6:

Format	Resolution	Frame rate	Frame size in pixels	Color Format	Bits per pixel	Bitrate
576i/50	720 x 576	25	414720	YUV422	16	159 Mbps
480i/60	720 x 480	30	345600	YUV422	16	159 Mbps
576p/50	720 x 576	50	414720	YUV422	16	317 Mbps
480p/60	720 x 480	60	345600	YUV422	16	317 Mbps
720p/50	1280 x 720	50	921600	YUV422	16	704 Mbps
720p/60	1280 x 720	60	921600	YUV422	16	844 Mbps
1080p/25	1920 x 1080	25	2073600	YUV422	16	792 Mbps
1080p/30	1920 x 1080	30	2073600	YUV422	16	950 Mbps
1080i/50	1920 x 1080	25	2073600	YUV422	16	792 Mbps
1080i/60	1920 x 1080	30	2073600	YUV422	16	950 Mbps

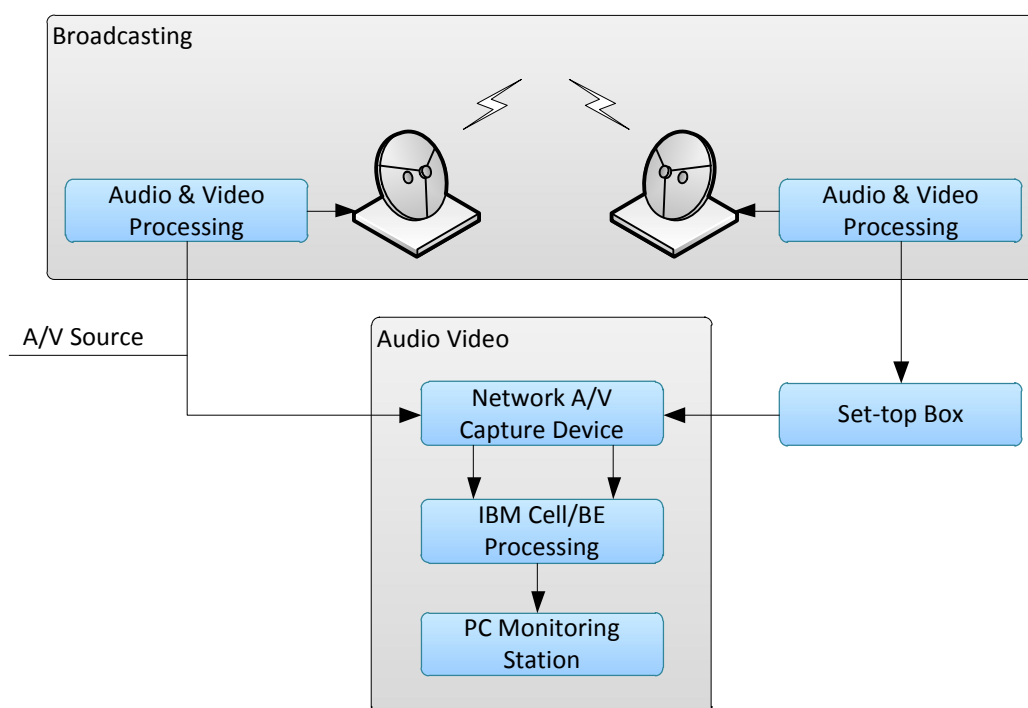
Tabela 5 – Širina propusnog opsega za video formate

Sample rate	Sample size	No Channels	Bitrate
44100 Hz	16bit	2	1411.2 Kbps
48000 Hz	16bit	2	1536 Kbps
96000 Hz	24bit	2	4608 Kbps

Tabela 6 – Širina propusnog opsega za audio formate

Na osnovu tabela očigledno je da je širina propusnog opsega od 1Gbps dovoljna za prenos dva audio-video toka SD kvaliteta slike bez posebne kompresije. Ipak dva audio-video toka HD kvaliteta premašuju propusnost ugrađenog mrežnog adaptera na PS3 pa je kompresija neophodna.

Koncept rešenja dat je na sledećoj slici:



Slika 14 – Koncept rešenja

Tipična primena opisanog sistema je otkrivanje i merenje greške audio-video sinhronizacije koja nastaje između televizijskog studija i gledaoca. Predloženo rešenje koristi pristup sa punom referencom što podrazumeva prisustvo dva AV signala:

- referentnog signala
- signala čiji se stepen sinhronizacije meri u odnosu na referentni

U opisanom slučaju primene referentni signal bi bio signal u televizijskom studiju pre slanja, dok bi se merila sinhronizacija signala na strani TV prijemnika, nakon prenosne mreže.

Fizički sistem se sastoji iz tri celine:

1. uređaja za prijem, prilagođavanje formata, kompresiju i slanje audio video signala mrežom
2. uređaja zasnovanog na Cell procesoru zaduženog za mrežni prijem signala, dekompresiju, merenje i slanje rezultata

3. udaljena radna stanica za prijem i interpretaciju rezultata merenja

Predviđeno je da se ciljnoj platformi simultano prosleđuju dva audio-video toka posredstvom mrežnog adaptera. Na ciljnoj platformi se audio odbirci i slike smeštaju u prijemne bafere. Kada se prijemni baferi popune vrši se upoređivanje video i audio odbiraka referentnog i signala čije se kašnjenje meri. Razlika kašnjenja između audio i video signala se smatra greškom sinhronizacije. Najveća merljiva greška sinhronizacije zavisi od veličine prihvatnih bafera. Rezultat merenja se posredstvom mreže šalje na radnu stanicu za nadgledanje. Nakon slanja rezultata baferi se prazne i počinje novi ciklus merenja. Mala količina sistemske memorije na ciljnoj platformi uslovljava mali prijemni bafer što znači i manji opseg merenja.

U narednim poglavljima će biti detaljno objašnjene fizičke komponente sistema.

3.1 Mrežni prijem i slanje

Za predloženo rešenje neophodno je obezbediti prenos slike i zvuka u digitalnom domenu posredstvom računarske mreže. Za konverziju televizijskog signala u zahtevani oblik koriste se mrežni uređaji za prijem signala (eng. *Network Attached Video Capture*).

Uređaji ovog tipa u realnom vremenu vrše prijem signala sa nekog od audio-video ulaza, digitalizuju signal, prilagođavaju format, vrše kompresiju a zatim slanje putem lokalne računarske mreže. RT-RK RT-AV100 je predstavnik ove klase uređaja.

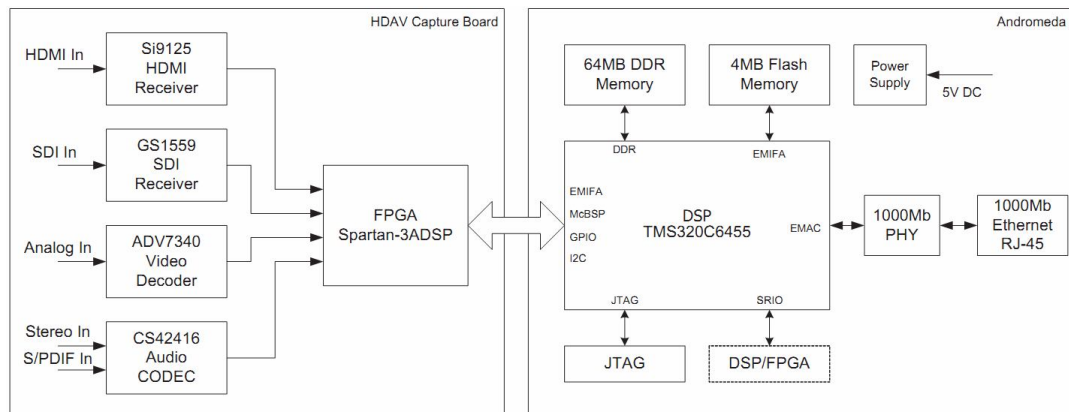
3.1.1 RT-RK RT-AV100

RT-AV100 je samostalni uređaj za prijem, obradu i slanje audio-video signala u realnom vremenu. Uređaj podržava velik broj ulaznih priključaka što je prikazano na slici Slika 15.



Slika 15 – Ulazni priključci RT-AV 100

Uređaj takođe omogućava kompresiju bez gubitaka kvaliteta (eng. *lossless*) u realnom vremenu u razmeri 3:1. Visok stepen kompresije je neophodan da bi dva audio-video toka mogla simultano da se šalju kroz mrežu od 1Gbps. Prijem audio-video signala obavlja FPGA (eng. *Field-programmable gate array*) procesor. Za kompresiju i prilagođenje formata zadužen je DSP (eng. *Digital Signal Processor*) procesor. Unutrašnja arhitektura uređaja data je na Slika 16.



Slika 16 – Unutrašnja arhitektura RT-RK RT-AV100

3.1.2 Mrežna oprema

Mrežni *svič* (srp. *komutator*, eng. *switch*) je uređaj koji upravlja protokom podataka na lokalnoj računarskoj mreži (eng. *Local Area Network - LAN*). Da bi se obezbedila potrebna brzina prenosa podataka od 1Gbps neophodno je i da mrežna oprema podržava 1000BASE-T standard. Sistem za otkrivanje i merenje greške sinhronizacije sastoji se od četiri mrežna uređaja: dva RT-AV100, PS3 i PC radne stanice za praćenje rezultata. U najjednostavnijem slučaju potreban je gigabitni *svič* sa najmanje 4 priključka (eng. *port*).



Slika 17 – Gigabitni mrežni svič sa pet priključaka

3.2 PlayStation 3

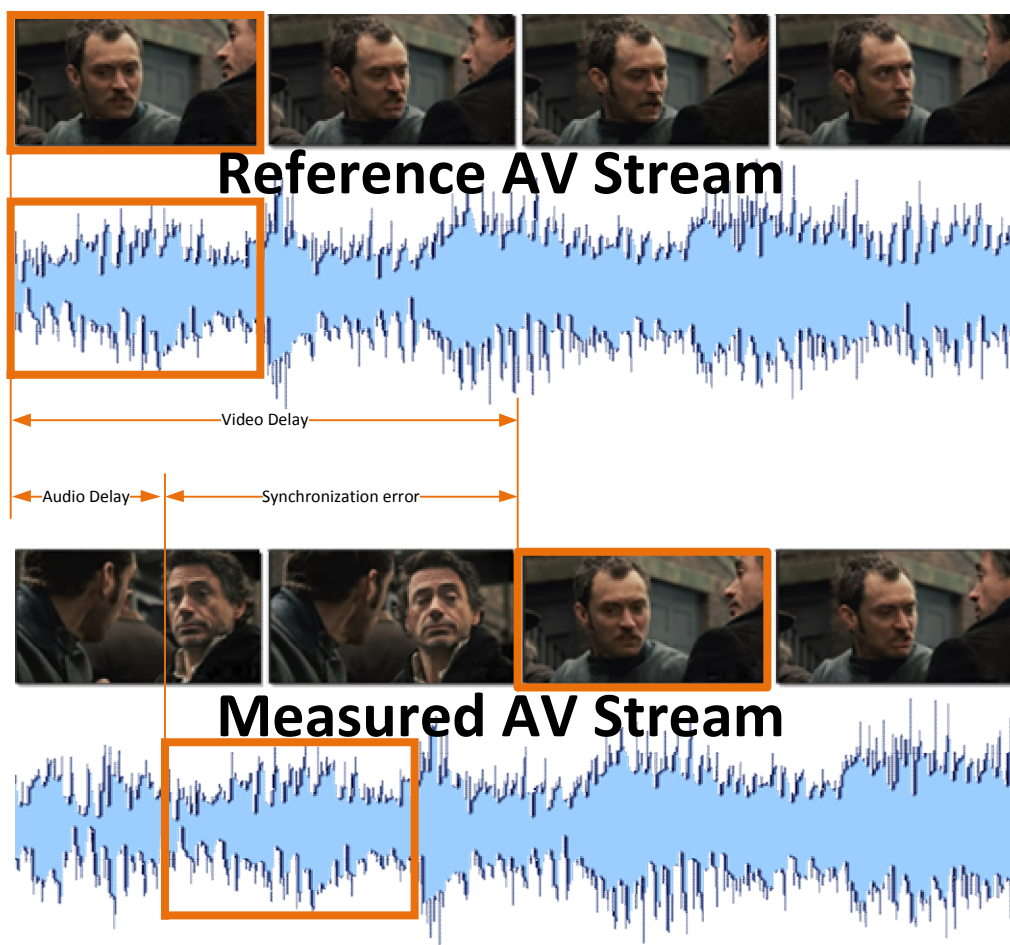
Uloga PS3 u opisanom rešenju je višestruka, uređaj istovremeno vrši:

1. prijem audio i video paketa posredstvom mreže
2. smeštanje sadržaja primljenih audio video paketa u memoriju
3. merenje kašnjenja audio i video signala

4. slanje rezultata merenja radnoj stanici za praćenje

Proces prijema audio i video paketa se razlikuje najviše zbog samog tipa signala. Za razliku od audio signala koji predstavlja neprekidan tok podataka video signal je po svojoj prirodi podeljen na slike. Stoga je potrebno po prijemu video paketa rastumačiti njihov sadržaj i rekonstruisati celu sliku pre nego što se ona uskladišti u memoriju i prosledi na dalju obradu. Audio paketi nemaju tu vrstu obrade već ih je dovoljno pravilnim redoslednom smestiti u memoriju. Bitno je napomenuti da je potrebno rastumačiti i kontrolne poruke mrežnog uređaja za slanje jer se podaci o prisustvu i tipu kompresije nalaze u njima. Uređaj RT-AV100 ne vrši kompresiju audio signala, dok je kod video signala nakon prenosa celokupne slike potrebno je dekompresovati kako bi se dobila slika u YUV422 formatu. Zbog uštede memorijskog prostora i procesorskog vremena, pohranjivanje i obrada se vrši samo na luminansi slike (eng. *luminance*, *luma*). Luminansa (Y) je komponenta YUV formata koja nosi informaciju o osvetljenosti dok se hrominansa (UV) tj. komponente koje nose informacije o boji najčešće odbacuju kod algoritama za obradu slike.

Zbog ograničenog kapaciteta sistemske memorije ciljne platforme ograničen je i broj slika i odgovarajućih audio odbiraka koji se može pohraniti. Zbog prirode algoritama ovo ograničenje direktno utiče i na merni opseg. Zbog toga je potrebno na početku rada na osnovu formata audio-video toka izračunati količinu potrebne memorije za skladištenje jedne slike (i korespondentnog dela audio signala) te koliko ih je moguće smestiti u raspoloživu memoriju. Samim tim je izračunat i merni opseg za dati audio-video materijal. Algoritmi za merenje kašnjenja i greške sinhronizacije se izvršavaju nakon popunjavanja prijemnih *bafera*. Način računanja greške sinhronizacije je predstavljen na slici Slika 18.



Slika 18 – Odnos kašnjenja i greške sinhronizacije

Greška audio-video sinhronizacije je razlika kašnjenja audio i video signala. Stoga je potrebno izmeriti kašnjenje audio i video signala u odnosu na referentni signal. Merenje kašnjenja video signala se meri tako što se među pohranjenim slikama oba signala traži poklapanje. Kašnjenje izraženo u milisekundama zavisi od broja slika u sekundi (eng. *framerate*):

$$\text{delay}[\text{frames}] \times \frac{1}{\text{framerate} \left[\frac{\text{frames}}{s} \right]} = \text{delay}[s]$$

Poređenjem audio signala se na sličan način nalazi i kašnjenje audio signala izraženo u odbircima (eng. *samples*). Kašnjenje izraženo u milisekundama zavisi od broja semplova u sekundi (eng. *samplerate*) i broja kanala:

$$\text{delay}[\text{samples}] \times \frac{1}{\text{channels} \times \text{samplerate} \left[\frac{\text{samples}}{s} \right]} = \text{delay}[s]$$

Ukoliko se usvoji da predznak "+" označava da signal koji se meri kasni za referentnim, "-" obrnut slučaj, greška sinhronizacije je jednaka:

$$E_{\text{sync}} = \text{delay}_{\text{video}} - \text{delay}_{\text{audio}}$$

3.2.1 Algoritmi

Poređenje (traženje poklapanja) slika ili njihovih delova je jedna od najistraženijih oblasti digitalne obrade signala. Najčešće korišćeni algoritmi za poklapanje (eng. *matching*) slika zasnivaju se na srednjem kvadratu greške (eng. *Mean Square Error - MSE*). MSE daje meru koliko su tačke dve slike slične - pri čemu je srednji kvadrat greške jednak nuli kada su slike identične. Sa druge strane ovaj algoritam ne uzima u obzir sam sadržaj slika. Ukoliko se porede dve slike istog sadržaja od kojih je jedna tamnija ili svetlija od druge greška će biti velika što ukazuje da su slike različite.

Pri izboru algoritma bitno je prepoznati projektne zahteve i na osnovu njih izabrati adekvatan algoritam. Osobnosti projektnog zahteva se zasnivaju na dve bitne činjenice:

1. porede se uzastopne slike video toka: zbog toga se može očekivati veliki broj gotovo identičnih slika naročito kod statičnih scena. Algoritam mora biti dovoljno robustan da eliminiše pojavu lažnih poklapanja, a sa druge strane da ne odbaci poklapanje zbog niskog praga tolerancije u slučaju pojave izobličenja. Treba imati u vidu da pogrešno prepoznavanje n-te slike kao n+1 slike u slučaju 24p materijala znači grešku merenja od 40ms što je nedopustivo uzimajući u vidu raspon dopušten međunarodnim standardima.
2. između referentnog i signala koji se testira mogu se pojaviti velike razlike na binarnom nivou čak i kada se slike poklapaju. Analogni artefakti poput šuma i digitalni poput blocking artefakta se očekuju i ne smeju uticati u znatnoj meri na ispravnost rada algoritma.

Pretvaranje formata i naknadna obrada (eng. *postprocessing*) u STB neminovno unose razliku između referentnog i merenog signala. Obzirom da se ne može očekivati savršeno (eng. *bitperfect*) poklapanje referentnog i testiranog signala, a imajući u vidu uticaj spoljnih faktora poput šuma može se zaključiti da algoritmi koji se zasnivaju na oduzimanju sa fiksnim pragom poput srednjeg kvadrata greške nisu podesni za ovu namenu. Eksperimentalni rezultati u toku razvoja idu u prilog ovoj tvrdnji.

U narednim poglavljima će detaljno biti opisani korišćeni algoritmi za traženje poklapanja slike i zvuka.

3.2.2 Video matching algoritam - varijansa

Za slučajnu promenljivu X diskretnog tipa sa raspodelom $P(x)$ i N mogućih vrednosti za x_i , varijansa ili disperzija populacije σ^2 (eng. *variance*) se definiše kao:

$$\sigma^2 = \sum_{i=1}^N P(x_i)(x_i - \mu)^2,$$

gde je μ matematičko očekivanje za X . U slučaju kontinualne promenljive X :

$$\sigma^2 = \int P(x)(x - \mu)^2 dx.$$

Kada distribucija nije poznata, varijansa uzorka se računa kao

$$s_N^2 = \frac{1}{N} \sum_{i=1}^N (x - \bar{x})^2,$$

gde je N broj uzoraka, a srednja vrednost uzoraka se računa kao:

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n x_k$$

Treba primetiti da varijansa uzorka s_N^2 nije centrirana ocena varijanse populacije σ^2 . Da bi se dobila centrirana ocena varijanse za σ^2 potrebno je definisati korigovanu uzoračku varijansu:

$$s_{N-1}^2 = \frac{1}{N-1} \sum_{i=1}^N (x - \bar{x})^2$$

Razlog zbog čega s_N^2 daje necentriranu ocenu varijanse populacije je što su dva parametra μ i σ^2 procenjeni iz samih uzoraka. Da bi se dobila centrirana varijansa populacije od uzoraka sa n elemenata i *a priori* nepoznatom srednjom vrednošću (računa se iz uzorka) potrebna je centrirana $\hat{\sigma}^2$ ocena za σ^2 [12]:

$$\hat{\sigma}^2 = \frac{N}{N-1} s_N^2$$

Varijansa se može izračunati algoritmom sa dva prolaza. U prvom koraku se računa srednja vrednost potrebna u drugom koraku. Pseudokod ovog algoritma je dat u nastavku:

```

def two_pass_variance(data):
    n = 0
    sum1 = 0
    sum2 = 0

    for x in data:
        n = n + 1
        sum1 = sum1 + x

    mean = sum1/n

    for x in data:
        sum2 = sum2 + (x - mean)*(x - mean)

    variance = sum2/(n - 1)
    return variance

```

3.2.3 Audio matching algoritam - fazna korelacija

U obradi signala unakrsna korelacija (eng. *cross correlation*) je mera sličnosti dva signala predstavljena kao funkcija kašnjenja jednog od njih. U praksi se često koristi za merenje kašnjenja signala kao i za prepoznavanje šablona, prepoznavanje poznatog signala kao komponente složenog, merenje pomeraja, kriptanalizu, lokalizaciju izvora zvuka...

Upotreba unakrsne korelacije za prepoznavanje šablona je motivisana merenjem pomeraja tj. euklidske razdaljine:

$$d_{f,t}^2(u) = \sum_x [f(x) - t(x-u)]^2,$$

gde je f signal, t šablon. Dakle:

$$d_{f,t}^2(u) = \sum_x [f^2(x) - 2f(x)t(x-u) + t^2(x-u)]$$

Izraz $t^2(x-u)$ je konstantan, ako je $\sum f^2(x)$ približno konstantno onda je preostali izraz unakrsne korelacije:

$$c(u) = f(x)t(x-u) \tag{1}$$

mera sličnosti signala i šablona.

Ovako definisana korelacija pati od brojnih problema:

- ako se energija signala $\sum f^2(x)$ menja sa vremenom, traženje poklapanja upotrebom izraza (1) može dovesti do lažnog poklapanja. Na primer, korelacija između obeležja (eng. *feature*) i istovetnog dela signala može biti manja od korelacije obeležja i dela signala sa velikom energijom.
- opseg $c(u)$ zavisi od veličine obeležja
- jednačina (1) nije invarijantna na promene u amplitudi signala

Ipak ovi nedostaci se mogu nadoknaditi upotrebom normalizovane unakrsne korelacije. Normalizacija se vrši oduzimanjem srednje vrednosti i deljenjem sa standardnom devijacijom:

$$\gamma(u, v) = \frac{\sum_x [f(x) - \bar{f}] [t(x-u) - \bar{t}]}{\sqrt{\sum_x [f(x) - \bar{f}]^2 \sum_x [t(x-u) - \bar{t}]^2}} \quad (2)$$

gde je $\bar{t}$ srednja vrednost obeležja, a $\bar{f}$ srednja vrednost signala $f(x)$. Imenilac predstavlja standardnu devijaciju $f(x)$ i $t(x-u)$.

Razmatrajući brojilac u (2), sa pretpostavkom da postoje signali $f'(x) \equiv f(x) - \bar{f}$ i $t'(x-u) \equiv t(x-u) - \bar{t}$ gde su srednje vrednosti već oduzete dobija se jednakost:

$$\gamma^{num}(u, v) = \sum_x f'(x) t'(x-u) \quad (3)$$

Za oblast pretrage veličine M odbiraka i veličinu obeležja N jednačina (3) zahteva približno $N(M+1)$ sabiranja i $N(M+1)$ množenja.

Uzevši u obzir da konjugovanje vrši inverziju parametra osobinom Fourierove transformacije:

$$Ff^*(-x) = F^*(\omega) \quad (4)$$

jednačina (3) je konvolucija signala sa obrnutim obeležjem $t'(-x)$ i može se izračunati kao:

$$F^{-1}\{F(f')F^*(t')\} \quad (5)$$

gde je F Fourierova transformacija.

Pošto se (5) može efikasno računati u frekventnom domenu, razvijeno je više metoda aproksimacije energije signala iz (2). Varijacije intenziteta energije signala obeležja mogu se smanjiti upotrebom visokopropusnog (eng. *high-pass filter*) filtra pre računanja unakrsne korelacije. Filtriranje se može zgodno dodati u frekventnom domenu, ali je izbor granične frekvencije problematičan: niska granična frekvencija ostavlja velike varijacije energije signala dok visoka granična frekvencija može odstraniti informacije iz signala bitne za traženje poklapanja.

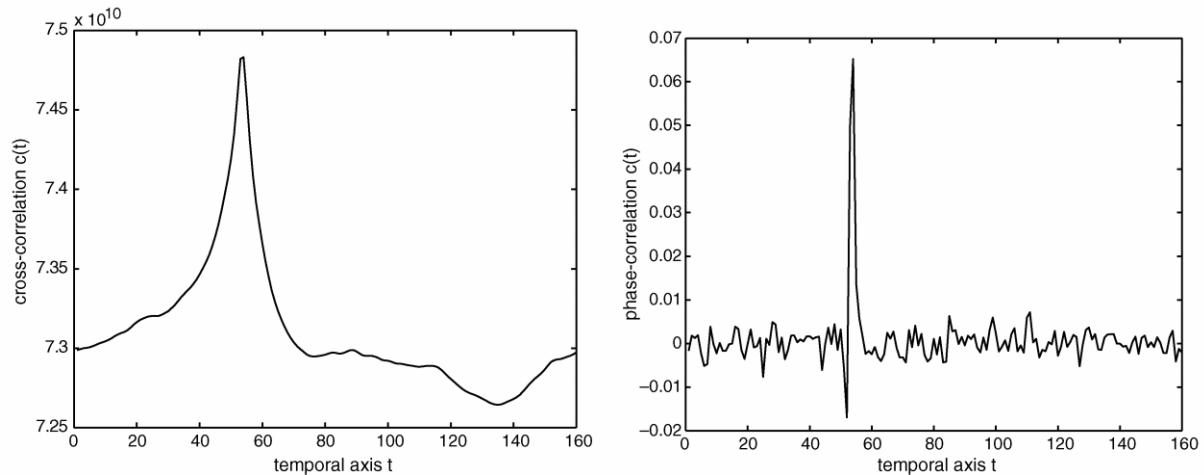
Robustniji pristup je varijanta korelacije iz (3) pod nazivom Fazna korelacija [13]. Fazna korelacija se često koristi za merenje kašnjenja signala kao i merenje translacionog i rotacionog pomeraja kod slika. Kod fazne korelacije se normalizuje u frekventnom domenu pre računanja inverzne Fourierove transformacije:

$$r = F^{-1}\left\{\frac{F(f')F^*(t')}{|F(f')F^*(t')|}\right\} \quad (6)$$

Izlaz fazne korelacije (u idealnom slučaju) je jedinstveni impuls koji se nalazi u tačnoj vrednosti pomeraja u . Dakle vremenski pomeraj u obeležja t u odnosu na signal f se nalazi u argumentu maksimuma r :

$$u = \arg \max \{r\} \quad (6)$$

Pored primene u merenju kašnjenja zvuka fazna korelacija se slično može koristiti i nad višedimenzionalnim vektorima podataka. Česta upotreba ovog algoritma je prepoznavanje odlika (eng. *feature*) slika npr. u registraciji slika (eng. *image registration*) i poravnavanju videa (eng. *video alignment*) gde je brzina algoritma zbog zahtevnijeg skupa podataka još bitnija. Istraživanja [13][15][16] su pokazala preciznije rezultate fazne u odnosu na unakrsnu korelaciju. Rezultat rada unakrsne i fazne korelacije na istom skupu podataka je dat na slici Slika 19 [16]:



Slika 19 – Poređenje rezultata unakrsne i fazne korelacije

Strmiji maksimum funkcije i veći dinamički opseg ukazuju na veću preciznost i manju mogućnost greške.

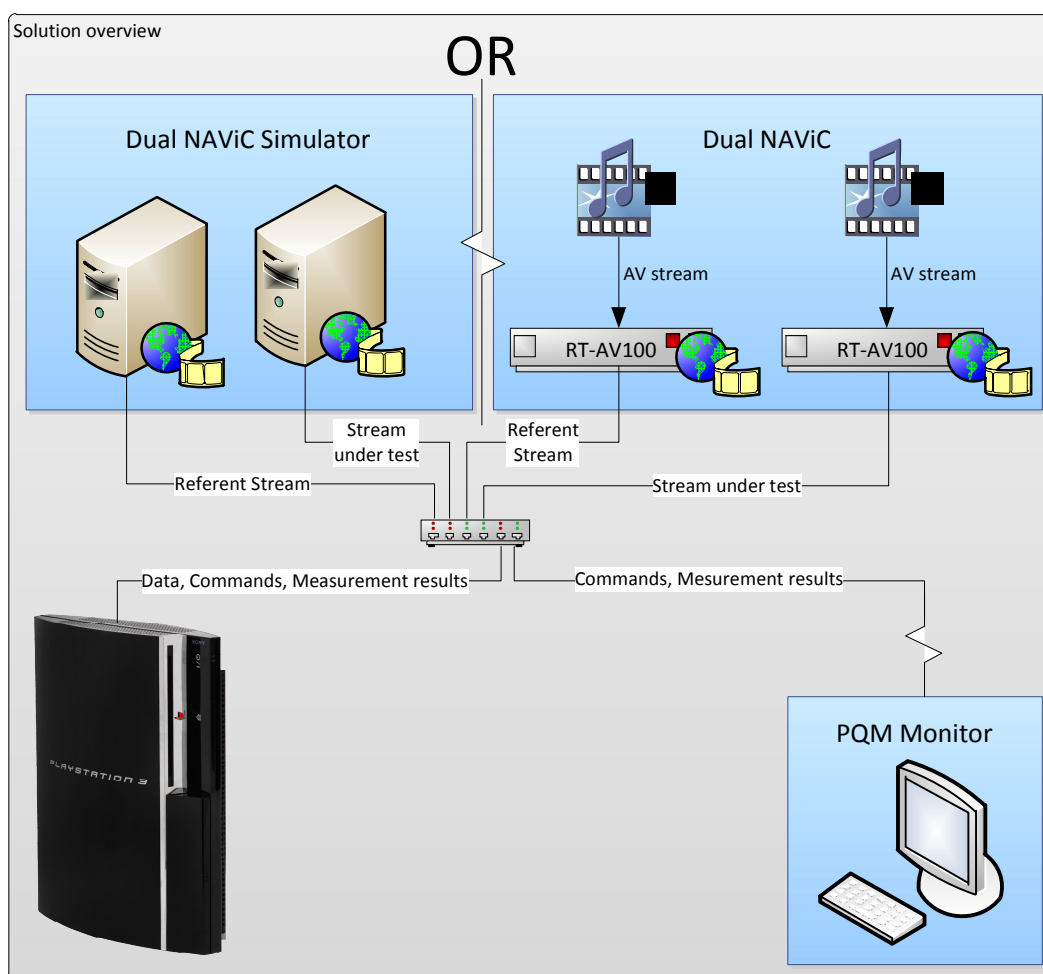
3.3 Radna stanica za praćenje rezultata

Radna stanica za praćenje rezultata prima rezultate merenja posredstvom računarske mreže i grafički ih prikazuje korisniku. Softver za radnu stanicu može se izvesti samostalno ili kao deo većeg sistema za merenje kvaliteta slike. U slučaju integracije u veći automatizovani sistem za merenje kvaliteta (*PQM*) potrebno je omogućiti dvosmernu komunikaciju sa *PS3* odnosno slanje komandi pored prijema rezultata. Za automatsko praćenje sinhronizacije većeg broja TV kanala moguće je primati podatke sa više uređaja istovremeno na udaljenoj lokaciji.

4. Opis rešenja

Ukupno programsko rešenje se može podeliti na tri celine:

1. uređaj za slanje audio video toka posredstvom računarske mreže
2. Cell programsko rešenje za otkrivanje i merenje greške audio video sinhronizacije
3. aplikacija za praćenje rezultata rada.

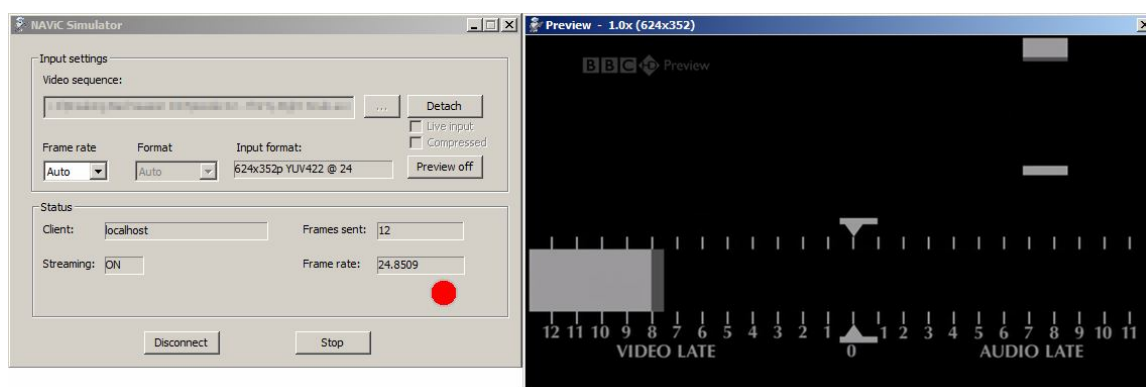


Slika 20 – Pregled fizičkih komponenata rešenja

Pregled fizičkog rasporeda komponenata rešenja prikazan je na slici Slika 20. Za vreme izrade programskog rešenja nisu bila dostupna dva RT-RK AV100 uređaja potrebna za razvoj i testiranje Cell aplikacije. Stoga je za potrebe razvoja korišćen *NAViCSim* simulator koji je promenjen i proširen za potrebe razvoja *Cell* programskog rešenja. U narednom poglavlju biće opisana funkcionalnost *NAViCSim* aplikacije kao i modifikacije za potrebe razvoja i testiranja programskog rešenja.

4.1 NAViCSim

NAViCSim je simulator uređaja RT-RK AV100 razvijen u istraživačko-razvojnom institutu RT-RK. Za razliku od fizičke implementacije RT-AV100 koji posredstvom računarske mreže šalje audio video signal sa nekog od svojih ulaznih priključaka prikazanih na slici Slika 15, *NAViCSim* koristi multimedijalne datoteke sa lokalnog diska računara na kom je pokrenut. Datoteke se zatim dekoduju i šalju računarskom mrežom na isti način kao kod fizičkog uređaja. Na taj način je moguć brz razvoj i testiranje programskih rešenja predviđenih za rad sa RT-AV100.



Slika 21 – Glavni i video prozor NAViC Simulatora

Za početak rada simulatora potrebno je odabrati ulaznu datoteku i uspostaviti vezu sa klijentom. Moguće je odabrati željenu brzinu slanja slike, izraženu u broju slika u sekundi (eng. *frame rate*).

Zadatak aplikacije je da :

- Upostavi vezu sa mrežnim klijentom (u ovom slučaju PS3)
- Učitava i dekodira multimedijalne datoteke
- Opciono prikazuje sliku koja se trenutno šalje
- Opciono komprimuje sadržaj HuffYUV kodekom
- Konverzija slike u UYVY format koji koristi RT AV-100 za slanje
- Šalje sadržaj slike posredstvom računarske mreže

NAViC simulator je Win32 aplikacija i koristi *pthread*, *DirectX*, *HuffYUV* i *FFmpeg* biblioteke.

HuffYUV je brz kodek (eng. *codec* - *coder/decoder*) za kompresiju videa bez gubitaka (eng. *lossless*). Kompresija bez gubitka je kompresija pri kojoj ne dolazi do gubitka podataka i kvaliteta informacija. Postupak je reverzibilan što znači da se kompresovana a zatim dekompresovana datoteka u potpunosti poklapa sa izvornom.

HuffYUV kodek kompresuje sliku predviđajući vrednost tačke na osnovu vrednosti tačaka u njenom okruženju. Vrednost greške se računa oduzimanjem predviđene vrednosti od

stvarne vrednosti tačke. Rezultat oduzimanja se zatim koduje Hafmanovim algoritmom (eng. *Huffman Coding*) koristeći drugačiju tabelu za svaki kanal. Tabele za kodiranje se odabiraju za vreme kodiranja iz ugrađene baze tabela a potom se smeštaju u komprimovanu datoteku i koriste pri dekompresiji.

Dekompresija slike se vrši obrnutim postupkom: hafmanovim dekodovanjem se dobijaju razlike predikcije za svaku tačku a zatim rekonstruiše vrednost tačke sabiranjem sa prediktovanom vrednošću.

FFmpeg je projekat slobodnog softvera koji sadrži brojne biblioteke i programe za manipulaciju multimedijalnim datotekama. Za izradu programskog rešenja najvažniji su:

- *ffmpeg* - konzolna aplikacija za pretvaranje video formata
- *libavcodec* - biblioteka koja sadrži sve FFmpeg audio/video kodere i dekodere
- *libavformat* - biblioteka koja sadrži multipleksere i demultipleksere za audio/video kontejnere
- *libswscale* - biblioteka koja sadrži rutine za skaliranje i konverziju prostora boja i formata tačke
- *libavutil* - pomoćna biblioteka koja sadrži funkcije zajedničke za druge delove FFmpeg. Između ostalog sadrži rutine za *adler32*, *crc*, *md5*, *sha1*, *lzo* kompresiju zatim *des*, *rc4* i *aes* enkripciju itd.

Microsoft *DirectX* je skup aplikacionih programskih sprega (eng. *API - application programming interface*) za rad sa multimedijom. U okviru programskog rešenja se koristi za prikaz video slike koja se trenutno šalje.

4.2 Adaptacija NAViC Simulatora

Adaptacija simulatora je izvršena u dve faze: u prvoj je proširena funkcionalnost dodavanjem podrške za audio i proizvoljan mrežni port dok je u drugoj fazi promenjen ulazni format audio video datoteka.

Zbog potreba projektnog zadatka da se signal prima sa dva uređaja (ili simulatora) proširena je funkcionalnost NAViC simulatora tako da se nakon pokretanja mora izabrati mrežni port. Takođe i prijemna strana (PS3) mora biti prilagođena tako da uspostavlja vezu na dva mrežna porta.

NAViC simulator ne podržava rad sa audio tokom pa je stoga morao biti proširen tom funkcionalnošću. Obzirom da *FFmpeg* podržava brojne video i audio formate implementacija programske podrške za dekodiranje i slanje audio odbiraka je bilo jednostavno.

Kod realizacije zadatka bitno je obezbediti pouzdan mehanizam za testiranje, kontrolu i merenja rezultata rada. Simulator je stoga izmenjen tako da koristi "sirove" audio i video formate: *PCM* i *DVS*. Na taj način je postignuto da se promena sinhronizacije slike i zvuka može jednostavno promeniti dodavanjem ili brisanjem audio odbiraka ili video slika. Na ovaj način je omogućeno jednostavno i pouzdano testiranje ispravnosti rada jer zbog odsustva kompresije, dekompresije i internih algoritama za sinhronizaciju koji su sastavni deo modernih *kodeka*.

4.3 PQM Monitor

PQM Monitor (eng. *Picture Quality Monitor - PQM*) je deo RT-RK *RT-IP030* sistema za merenje kvaliteta multimedijalnog signala u realnom vremenu bez reference. U ovom sistemu PQM Monitor pruža nadgledanje procenu kvaliteta, uzbunjivanje i praćenje mernih vrednosti u realnom vremenu.

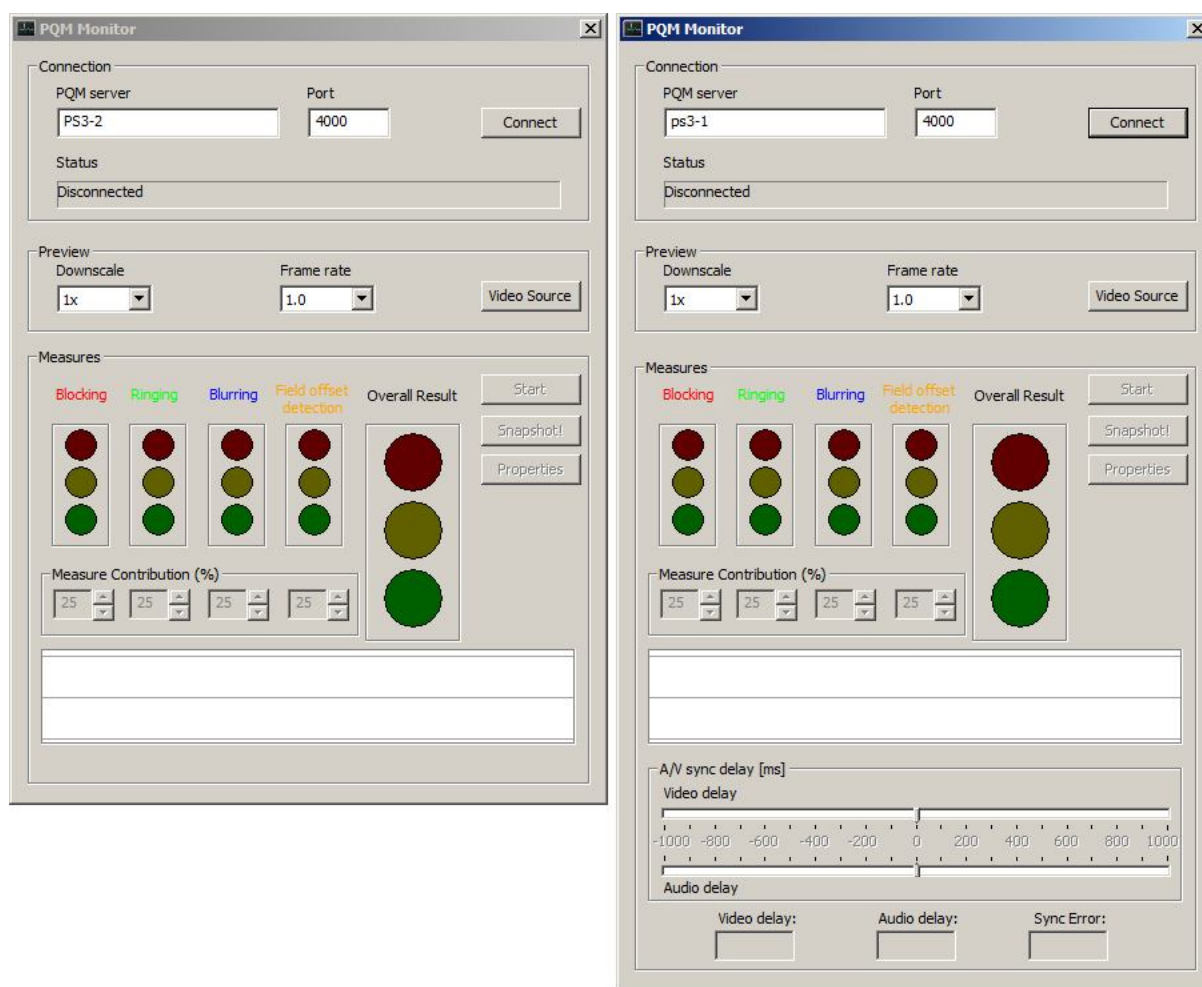
Glavni zadatak PQM Monitora je vizuelizacija i prikaz rezultata merenja dobijenih od PS3 servera za računanje (eng. *Compute Server*). RT-IP030 sistem za ocenu kvaliteta slike meri prisustvo četiri artefakta karakteristična za MPEG kompresiju:

1. Blocking - pojava blokova usled greške kvantizacije
2. Ringing - pojava pseudo ivica uslovljenih visokofrekventnim šumom
3. Blurring - zamućenje slike kao posledica ograničenog propusnog opsega
4. Field offset - efekat češlja

Nakon pokretanja aplikacije PQM Monitor potrebno je uspostaviti vezu sa PS3 posredstvom računarske mreže. Pritiskom na dugme "Start" šalje se mrežna komanda za početak računanja na PS3. Algoritam za merenje kvaliteta se izvršava i šalje podatke nazad PQM Monitoru gde se oni prikazuju u realnom vremenu.

4.4 Adaptacija PQM Monitora

U cilju integracije projektnog zadatka u postojeće programsko rešenje za merenje kvaliteta slike aplikacija PQM Monitor je proširena novim funkcijama. Dodata su polja za prikaz trenutnog kašnjenja audio i video signala, trenutne greške audio-video sinhronizacije u odnosu na referentni tok kao i klizači za njihov vizuelni prikaz. Promene u izvornom kodu se svode na dodavanje dodatnih polja u strukture za prenos rezultata merenja i njihov prikaz na glavnoj formi aplikacije. Izgled početne i izmenjene verzije je dat na slici Slika 22.

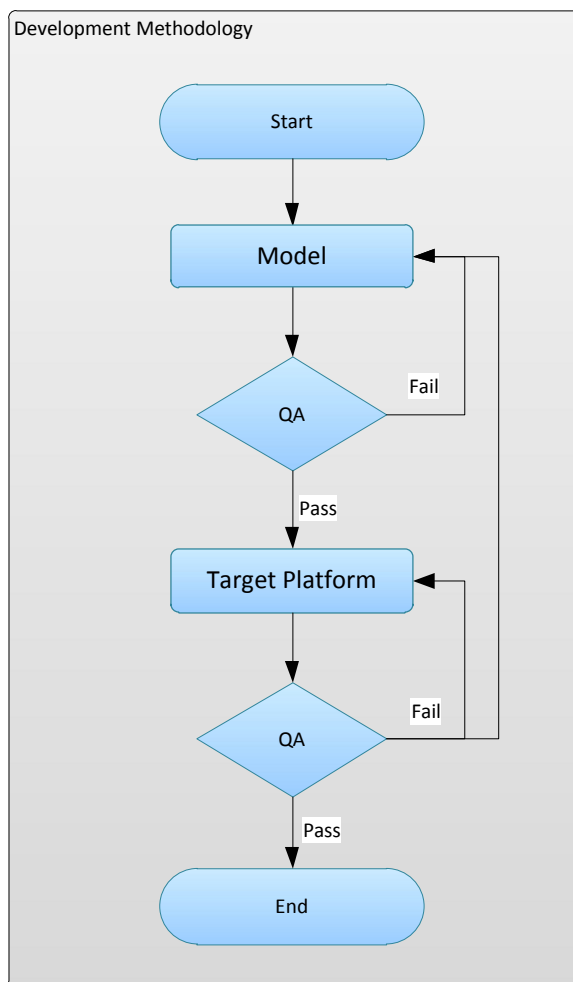


Slika 22 – PQM monitor (levo) i modifikovana verzija (desno)

4.5 Metodologija razvoja

Zbog specifične fizičke arhitekture programiranje za Cell ciljnu platformu je dosta otežano. Zbog toga se razvoj i verifikacija programskog rešenja vrši na PC x86 arhitekturi a zatim prepravlja za Cell gde se ponovo verifikuje ispravnost rada. Ciklus razvoja programskog rešenja dat je na slici Slika 23.

NAViC simulator i PQM monitor, aplikacije za slanje audio video toka i vizuelizaciju rezultata, su Win32 aplikacije za x86 platformu. Za njihov razvoj korišćen je Microsoft *Visual Studio*, integrisano razvojno okruženje (eng. *IDE - Integrated Development Environment*), što omogućava brži razvoj, lakše ispravljanje grešaka i jednostavniju verifikaciju ispravnosti rada. Osim toga ove aplikacije oslanjaju se na postojeća rešenja pa se njihov razvoj svodi na proširenje i modifikaciju postojeće funkcionalnosti. Zbog toga u ovom poglavlju će dublje biti razmotrena samo metodologija razvoja algoritama za traženje poklapanja audio i video signala koji su opisani u poglavljima 3.2.2 i 3.2.3. U toku razvoja ukupnog programskog rešenja bilo je potrebno razviti nekoliko pomoćnih alata, pa će njihova primena biti objašnjena u daljem tekstu.



Slika 23 – Metodologija razvoja i verifikacije

Prvi korak implementacije algoritma je odabir referentnog skupa ulaznih podataka koji će se koristiti za verifikaciju ispravnosti rada algoritma. Referentni skup ulaznih podataka se generiše na osnovu referentnih audio video sekvenci. Za potrebe verifikacije bilo je neophodno razviti mehanizam za precizno unošenje merljivog kašnjenja audio i video signala.

Za potrebe testiranja potrebno je obezbediti skup sekvenci sa različitim i poznatim stepenom kašnjenja kako bi se rezultat merenja mogao uporediti sa poznatom vrednošću. U toku razvoja pokazalo se da moderni kodeci dele audio tok u tzv. audio okvire (eng. *audio frame*) čije se vreme trajanja ne poklapa sa trajanjem video slike (eng. *video frame*). Uz to ponovno kodiranje i dekodiranje unosi određen nivo distorzije u video a kašnjenja (eng. *latency*) u audio signal. Zbog toga je odlučeno da se ulazne sekvence konvertuju u sirov format a NAViCSim adaptira da koristi takve audio-video sekvence. Razvijen je namenski alat "*DVS-PCM dump*" koji koristi *ffmpeg* za dekodovanje sekvenci a za snimanje koristi nekomprimovan format: *DVS* za video i *PCM* za audio. Korišćenjem sirovih formata unošenje tačno određene količine kašnjenja postaje trivijalno: dodavanjem slike za video i dodatnih odbiraka za audio. Na ovaj način generisan je veći broj ulaznih sekvenci sa poznatim kašnjenjem za potrebe verifikacije algoritma. Takođe su generisane i sekvence sa degradiranim nivoom kvaliteta za potrebe testiranja robustnosti.

4.5.1 Matlab simulacija

Početak implementacije algoritma je izrada modela u programskom paketu *Matlab*. Nakon verifikacije modela on se koristi kao referentno rešenje pri izradi C i programskog koda za Cell platformu.

U nastavku je dato Matlab rešenje modela algoritma za računanje fazne korelacije sa smanjenjem uzorkovanja (eng. *subsampling*) devet puta.

```
function [m,n] = phase_correlation(Y1,Y2)

b=[1/9 1/9 1/9 1/9 1/9 1/9 1/9 1/9 1/9];
a=1;

Y1F=filter(b,a,Y1);
Y2F=filter(b,a,Y2);
Y1FS=decimate(Y1F,9);
Y2FS=decimate(Y2F,9);

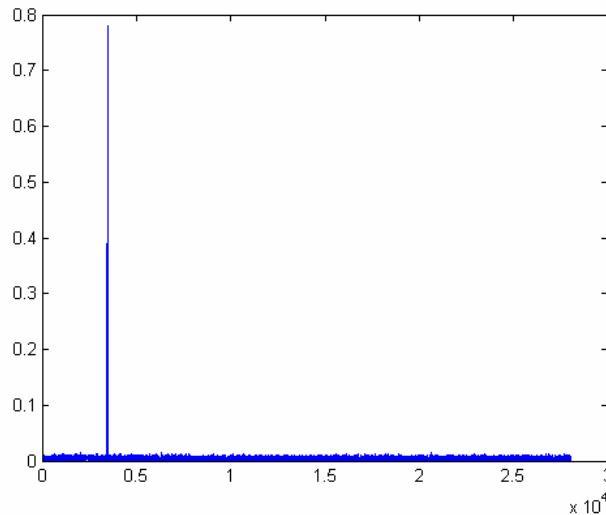
FY1=fft(Y1FS);
FY2=fft(Y2FS);
U=FY1.*conj(FY2);
D=abs(real(U))+abs(imag(U))+0.001;

PC=U./(D);
ssh=ifft(PC);
[m,n]=max(ssh)
figure,plot(ssh)

YT=ifft(FY1)

%pdm = exp(i*(angle(FY1)-angle(FY2)));
%cps = real(ifft2(pdm));
%[m,n]=max(cps)
%figure,plot(cps)
```

Rezultat Matlab simulacije dat je na slici Slika 24. Maksimum funkcije predstavlja pomeraj dva signala. Obzirom da je frekvencija uzorkovanja efektivno smanjena 9 puta, kašnjenje signala izraženo u broju odbiraka je 9 puta veće od argumenta maksimuma funkcije na slici.



Slika 24 – Rezultat rada Matlab simulacije algoritma fazne korelacije

4.5.2 C programski kod za x86 arhitekturu

Druga faza razvoja algoritma je implementacija C koda za x86 procesor. Nakon implementacije algoritma vrši se njegova verifikacija poređenjem rezultata rada sa Matlab simulacijom na osnovu poznatog skupa ulaznih podataka. Deo implementacije algoritma za računanje fazne korelacije u programskom jeziku C dat je u nastavku:

```
void phase_cor(double* f, double* fi, double* g, double* gi, double* s, double* si)
{
    double tmp;
    for(int i=0; i<FFT_SIZE; i++)
    {
        s[i] = ( g[i] * f[i] ) - ( -gi[i] * ( fi[i] ) );
        si[i] = ( g[i] * ( fi[i] ) ) + ( -gi[i] * f[i] );
        tmp=abs(s[i])+abs(si[i])+0.01;
        s[i] = s[i]/tmp;
        si[i] = si[i]/tmp;
    }
}

void findmax(double* s, double* max, int* delay)
{
    double temp;
    max =0;
    for(int i=0; i<FFT_SIZE; i++)
    {
        temp = s[i];
        if(temp>*max)
        {
            *max = temp;
            *delay = i;
        }
    }
}

...
filter(f);
filter(g);

subsample(f);
subsample(g);

FFT(1,17,f,fi);
```

```

FFT(1,17,g,gi);

phase_cor( f, fi, g, gi, s, si);

FFT(-1,17,s,si);

findmax(s,&max,&delay);
...

```

4.5.3 Programski kod sa posebnim funkcijama za x86 arhitekturu

Nakon verifikacije C koda za x86 procesor sledeći korak je adaptacija algoritma za SPU jezgra Cell procesora. Razvoj programskog rešenja za Cell platformu je otežan specifičnom fizičkom arhitekturom i slabom programskom podrškom za ispravljanje grešaka.

U istraživačko-razvojnom institutu RT-RK razvijena je C biblioteka koja sadrži posebne funkcije (eng. *intrinsic function*) za SPU instrukcije. U teoriji prevodioca intrinsik je funkcija programskog jezika koja se prevodi na poseban način: najčešće se poziv funkcije zamenjuje nizom unapred definisanih instrukcija [18]. Korišćenjem ove biblioteke omogućen je razvoj SPU koda u razvojnom okruženju *Visual Studio* na *Windows* operativnom sistemu uz upotrebu svih njegovih pogodnosti za ispravljanje grešaka. Na taj način je znatno olakšan razvoj SPU programskog koda. Uz upotrebu takvih posebnih funkcija, ukoliko se ispoštuju sve paradigme SPU programiranja, kasnija adaptacija na SPU procesor i proširenja programskog jezika za vektorsku obradu podataka [19][20] postaje trivijalna. U nastavku teksta dat je primer x86 implementacije *spu_madd* posebne funkcije za vektorsko sabiranje-množenje:

```

typedef union _vec_int4 {
    int i[4];
    unsigned char uc[16];
} vec_int4;

typedef union _vec_short8 {
    short s[8];
    unsigned char uc[16];
} vec_short8;

/*****
 * SPU INTRINSIC - spu_madd
 * Each element of vector a is multiplied by vector b and added to the
 * corresponding element of vector c and returned to the corresponding
 * element of vector d. For integer multiply-and-adds, the odd elements
 * of vectors a and b are sign extended to 32-bit integers prior to
 * multiplication.
 *****/
vec_int4 spu_madd(vec_short8 a, vec_short8 b, vec_int4 c)
{
    int i;
    vec_int4 d;
    for (i=0; i<4; i++)
    {
        d.i[i] = (int)c.i[i] + (int)a.s[(i<<1)+1] * (int)b.s[(i<<1)+1];
    }
    return d;
}

```

Razvijene su ukupno tri verzije implementacije algoritama sa upotrebom biblioteke sa posebnim funkcijama:

1. Referentni programski kod
2. Particionisan programski kod prilagođen DMA transferima

3. Programski kod sa emulacijom SPU SIMD (eng. *Single Instruction Multiple Data*) instrukcija

Referentni kod je u suštini postojeći x86 programski kod prethodno verifikovan poređenjem rezultata rada sa Matlab verzijom algoritma. Promene se odnose samo na uređenje izvornog koda u dve datoteke SPU.c i SPU.h. Nakon verifikacije ispravnosti rada poređenjem sa Matlab verzijom ova verzija se koristi za verifikaciju ostalih verzija programskog koda.

U prethodnim poglavljima je objašnjeno da SPU jezgra nemaju neposredan pristup sistemskoj memoriji. Zbog toga je audio i video odbirke neophodno učitati u LS memoriju pre izvršavanja algoritma. Treba imati u vidu da podaci koji se prenose moraju biti poravnati (eng. *aligned*) tj. moguće je preneti samo podatke veličine od 1, 2, 4 i 8 bajta kao i veličine koje su umnošci 16 bajta. Najveća količina podataka koju je moguće preneti u jednom transferu je 16KB. Najbolje performanse se dobijaju kada su podaci poravnati na 128 bajta a veličina transfera je umnožak 128 bajta. Da bi se olakšala kasnija adaptacija algoritma za stvarno SPU jezgro u drugoj verziji koda dodata je programska sprema koja simulira DMA prenos memorije iz sistemske RAM u LS memoriju.

Nakon verifikacije prve dve verzije izvornog koda pristupilo se izradi verzije koda koja koristi posebne funkcije za emuliranje SIMD instrukcija za vektorko izračunavanje. Na ovaj način se olakšava dalja adaptacija programskog koda za Cell platformu.

4.6 Adaptacija programskog koda za Cell platformu

Prva faza adaptacije algoritama na SPU jezgra je izrada *Makefile*-ova. Da bi se SPU kod preveo za Cell potrebno je adaptirati postojeći kod da se umesto posebnih funkcija (intrinsika) koriste stvarne SPU instrukcije. U nastavku je dat odlomak koda koji ilustruje kako je simulacija DMA transfera iz sistemske memorije u LS adaptirana u stvaran zahtev MFC memorijskom kontroleru.

Simulacija DMA na x86 upotrebom *memcpy()* funkcije:

```
void GetDataFromPPU(void* dest, void* src, int size, int tag){
    memcpy(dest, src, size);
};

void SendDataToPPU(void* dest, void* src, int size, int tag){
    memcpy(dest, src, size);
};
```

Stvaran SPU zahtev MFC kontroleru:

```
static __inline void GetDataFromPPU(void* dst, unsigned long long src, int len, int tag)
{
    mfc_get(dst, src, len, 1, 0, 0);
    mfc_write_tag_mask(tag);
    mfc_read_tag_status_all();
}

static __inline void SendDataToPPU(unsigned long long dst, void* src, int len, int tag)
{
    mfc_put(src, dst, len, 1, 0, 0);
    mfc_write_tag_mask(tag);
    mfc_read_tag_status_all();
}
```

Nakon modifikacije programskog koda uzimajući u obzir osobenosti SPU procesora, kod se prevodi SPU prevodiocem i dalje nastavlja razvoj koristeći Cell platformu. Nakon integracije u postojeće programsko rešenje, programski kod se optimizuje tehnikama opisanim u poglavlju 2.4.6.

4.7 Opis Cell programskog rešenja

U narednim poglavljima dat je opis programskih modula i najvažnijih komponenti implementiranog Cell programskog rešenja za PPU i SPU jezgra Cell procesora.

4.7.1 PPU

Podela posla u okviru programskog rešenja podrazumeva deo programskog koda namenjen PPU jezgru obavlja funkcije vezane za komunikaciju sa uslužiocima (eng. *server*) za mrežno slanje podataka i radnom stanicom za praćenje (eng. *monitoring*). Pored toga zadatak PPU procesora je i priprema podataka za obradu koju obavljaju SPU jezgra.

U daljem tekstu dato je objašnjenje najbitnijih modula, funkcija i struktura podataka koje se koriste u PPU programskom rešenju.

4.7.1.1 Modul za mrežnu komunikaciju - Comprot

Modul za mrežnu komunikaciju *Comprot* se sastoji od zaglavlja *comprot.h* i implementacije u datoteci *comprot.cpp*. Ovaj modul realizuje protokol za prijem i slanje naredbi i podataka. U opisanom programskom rešenju uloga PS3 je dvojaka: on je mrežni klijent za uređaje za slanje signala (eng. *network streaming server*) i mrežni poslužilac za *PQM* aplikaciju.

Funkcije za mrežnu komunikaciju implementirane su dvostruko, posredstvom TCP i UDP mrežnog protokola. Zbog boljih performansi u okviru programskog rešenja korišćene su UDP funkcije za mrežnu komunikaciju. Sledi opis najvažnijih programskih tipova, struktura podataka i programskih funkcija.

Podaci koji opisuju mrežni uslužilac odnosno klijent čuvaju se u strukturama podataka *cp_server_ctx* i *cp_client_ctx*:

```
struct _cp_server_ctx {
    SOCKET m_ServerSocket;
    void* m_pUserData;
    unsigned short m_usLocalPort;
    unsigned short m_usMaxConnections;
    unsigned short m_usConnectionCount;
    unsigned int m_uiMaxMessageSize;
    cp_client_ctx* m_pConnections;
    cp_event_callback m_pOnClientConnect;
    cp_event_callback m_pOnClientDisconnect;
    cp_message_callback m_pOnClientMessageCallback;
    pthread_t m_ServerThread;
    pthread_mutex_t m_ServerAccess;
};
typedef struct _cp_server_ctx cp_server_ctx;

struct _cp_client_ctx {
    SOCKET m_ClientSocket;
    void* m_pUserData;
    int ID;
    int* myStatus;
    unsigned char* m_pucBuffer;
    unsigned int m_uiMaxMessageSize;
};
```

```

    unsigned int m_uiWritePtr;
    unsigned int m_bLocalClient;
    struct sockaddr_in m_RemoteServerAddr;
    cp_event_callback m_pOnClientDisconnect;
    cp_message_callback m_pOnClientMessageCallback;
    pthread_t m_ClientThread;
    pthread_mutex_t m_ClientAccess;
    pthread_mutex_t m_SendMutex;
    pthread_cond_t* g_hResponseEventCond;
};
typedef struct _cp_client_ctx cp_client_ctx;

```

Najvažnije programske funkcije *comprot* modula su:

- *int cp_init()* - inicijalizuje mrežnu spregu
- *int cp_server_start(...)* - pokreće mrežni uslužilac
- *int cp_server_stop(...)* - zaustavlja mrežni uslužilac
- *cp_client_ctx* cp_client_add(...)* - dodaje PQM mrežni klijent
- *int cp_client_connect(...)* - uspostavlja vezu sa NAViC mrežnim uslužiocem
- *int cp_client_connect(...)* - vrši povezivanje klijenta sa mrežnim uslužiocem
- *int ReceiveMessage(...)* - prijem mrežne poruke
- *int ReadMessage(...)* - interpretira primljenu poruku

Programski kod klijenta i uslužioca izvršava se u programskim nitima:

- *void* client_thread_udp1(...)* - programska nit klijenta za prvi mrežni uslužilac
- *void* client_thread_udp2(...)* - programska nit klijenta za drugi mrežni uslužilac
- *void* server_thread(...)* - programska nit uslužioca

4.7.1.2 Modul za obradu - vse_proc_cap

Modul za prijem *vse_proc_cap* se sastoji od zaglavlja *vse_proc_cap.h* i implementacije u datoteci *vse_proc_cap.cpp*. Ovaj modul realizuje prijem audio video paketa putem računarske mreže, konverziju formata i upravljanje resursima za pohranjivanje slike i zvuka.

Struktura podataka *stClientCtxData_t* opisuje format primljenih audio video podataka:

```

typedef struct
{
    unsigned int m_uiHeight;
    unsigned int m_uiWidth;
    unsigned int m_uiVideoType;
    unsigned int m_uiTopField;
    unsigned int m_uiPrevTopField;
    unsigned int m_uiFrameIdentificator;
    unsigned int m_uiPrevFrameIdentificator;
    unsigned int m_uiInterlaceFactor;
    unsigned int m_uiFrameRate;
    unsigned int m_uiCompressionRatio;
    unsigned int m_uiPreviewWidth;
    unsigned int m_uiPreviewHeight;
    unsigned int m_uiSampleRate;
    unsigned int m_uiSampleSize;
    unsigned int m_uiNoChannels;
    unsigned int m_uiAudioFrameLenght;
} stClientCtxData_t, *hstClientCtxData_t;

```

Dozvoljena stanja programske niti za prijem podataka su definisana sledećim makroima:

```
#define VSEPROCCAP_OK 0
```

```
#define VSEPROCCAP_ERR -1
#define VSEPROCCAP_IDLE 1
#define VSEPROCCAP_INITIALIZED 2
#define VSEPROCCAP_CONNECTED 3
#define VSEPROCCAP_STARTED 4
```

Strukture podata *ucUDPBuffersPool1* i *ucUDPBuffersPool2* služe za skladištenje prispelih podataka sa servera za mrežno slanje. Strukturi *ucUDPBuffersPool1* se pristupa pomoću uvezanih lista *g_lUDPEmptyBuffersPool1* *g_lUDPCliantMsgBuffersPool1* a *ucUDPBuffersPool2* pomoću *g_lUDPEmptyBuffersPool2* i *g_lUDPCliantMsgBuffersPool2* na sledeći način:

- na početku rada lista *g_lUDPCliantMsgBuffersPoolX* je prazna dok *g_lUDPEmptyBuffersPoolX* sadrži niz pokazivača na memorijske lokacije *ucUDPBuffersPoolX*

- primljena poruka se smešta u *ucUDPBuffersPoolX* a pokazivač na njenu adresu se smešta na kraj *g_lUDPCliantMsgBuffersPoolX* list, adresa memorijske lokacije se briše iz *g_lUDPEmptyBuffersPoolX*

- nakon interpretiranja poruke sa početka *ucUDPBuffersPoolX* ona se kopira u odgovarajući bafer, a pokazivač na memorijsku lokaciju se briše iz *g_lUDPCliantMsgBuffersPoolX* i vraća na kraj *g_lUDPEmptyBuffersPoolX*. Deklaracije opisanih struktura podataka date su u nastavku teksta:

```
MEM_ALIGN(static unsigned char ucUDPBuffersPool1[MAX_BUFFS_NUM][MAX_MSG_SIZE], 128);
MEM_ALIGN(static unsigned char ucUDPBuffersPool2[MAX_BUFFS_NUM][MAX_MSG_SIZE], 128);
MEM_ALIGN(static matchingVFrame VideoMatchingBuffersPool1[MAX_MATCHING_RANGE], 128);
MEM_ALIGN(static matchingVFrame VideoMatchingBuffersPool2[MAX_MATCHING_RANGE], 128);
MEM_ALIGN(static matchingAFrame AudioMatchingBuffersPool1[MAX_MATCHING_RANGE], 128);
MEM_ALIGN(static matchingAFrame AudioMatchingBuffersPool2[MAX_MATCHING_RANGE], 128);
```

```
std::list<void*> g_lUDPEmptyBuffersPool1;
std::list<void*> g_lUDPCliantMsgBuffersPool1;
std::list<void*> g_lUDPVideoMsgBuffersPool1;
std::list<void*> g_lUDPAudioMsgBuffersPool1;
```

```
std::list<void*> g_lUDPEmptyBuffersPool2;
std::list<void*> g_lUDPCliantMsgBuffersPool2;
std::list<void*> g_lUDPVideoMsgBuffersPool2;
std::list<void*> g_lUDPAudioMsgBuffersPool2;
```

Prijem i opisana manipulacija bafera obavlja se u programskim nitima:

```
void* video_msg_proc_thread1(void* _pUserData)
void* video_msg_proc_thread2(void* _pUserData)
void* audio_msg_proc_thread1(void* _pUserData)
void* audio_msg_proc_thread2(void* _pUserData)
```

upotrebom sledećih funkcija:

- *void PutVideoMatchingBuffer1(...)* - smešta sliku u prvi video bafer
- *void *GetVideoMatchingBuffer1(...)* - doprema sliku iz prvog video bafera
- *int ClearVideoMatchingBuffer1()* - prazni prvi video bafer
- *void PutVideoMatchingBuffer2(...)* - smešta sliku u drugi video bafer
- *void *GetVideoMatchingBuffer2(...)* - doprema sliku iz drugog video bafera
- *int ClearVideoMatchingBuffer2()* - prazni drugi video bafer
- *void PutAudioMatchingBuffer1(...)* - smešta audio odbirke u prvi audio bafer
- *void *GetAudioMatchingBuffer1(...)* - doprema audio odbirke iz prvog bafera

- *int ClearAudioMatchingBuffer1()* - prazni prvi audio bafer
- *void PutAudioMatchingBuffer2(...)* - smešta audio odbirke u drugi audio bafer
- *void *GetAudioMatchingBuffer2(...)* - doprema audio odbirke iz drugog bafera
- *int ClearAudioMatchingBuffer2()* - prazni drugi audio bafer

U slučaju kada je video komprimovan *Huffyuv* kodekom, pre dalje obrade se dekomprimuje a zatim konvertuje nazad u YUV422 format slike upotrebom funkcija:

```
void RepackBuffer1(...)
void RepackBuffer2(...)
```

Smanjenje frekvencije uzorkovanja audio odbiraka postiže se upotrebom funkcije:

```
void filter(double* f)
```

Nakon popunjavanja prihvatnih bafera započinje slanje podataka SPU jezgrima koja su zadužena za pronalaženje poklapanja. Algoritmi za traženje poklapanja se izvršavaju na dva SPU jezgra:

1. algoritam fazne korelacije za pronalaženje poklapanja audio signala
2. varijansa za pronalaženje poklapanja video signala

Programske niti *video_matching_thread* i *audio_matching_thread* su zadužene za komunikaciju i prenos podataka ka SPU jezgrima.

```
void* video_matching_thread(void* _pUserData)
void* audio_matching_thread(void* _pUserData)
```

4.7.1.3 Modul za komunikaciju sa monitorom - vse_proc_mon

Modul za komunikaciju sa aplikacijom za prikaz rezultata se sastoji od datoteka *vse_proc_mon.h* i *vse_proc_mon.cpp*. Ovaj modul realizuje funkcije za komunikaciju sa aplikacijom za nadgledanje PQM Monitor. Najvažnije funkcije realizovane u ovom modulu su

```
int vseproc_mon_on_client_connect(...) - funkcija se poziva nakon povezivanja
int vseproc_mon_on_client_disconnect(...) - funkcija se poziva nakon prekida veze
int vseproc_mon_on_client_message(...) - funkcija se poziva nakon prijema poruke
int vseproc_mon_cleanup() - deinicijalizacija
int vseproc_mon_get_status() - vraća status PQM monitora
int vseproc_mon_send_results() - šalje rezultate rada PQM monitoru
int vseproc_mon_send_video_info(...) - šalje informacije o formatu slike
```

Dozvoljena stanja programske niti za komunikaciju su definisana sledećim makroima:

```
#define VSEPROC_MON_OK 0
#define VSEPROC_MON_ERR -1
#define VSEPROC_MON_IDLE 1
#define VSEPROC_MON_INITIALIZED 2
#define VSEPROC_MON_CONNECTED 3
#define VSEPROC_MON_STARTED 4
```

4.7.2 SPU

Jedan od zadataka programskog rešenja je integracija u postojeći sistem za merenje kvaliteta slike. Zbog toga deo programskog rešenja namenjenog SPU jezgrima zauzima dva jezgra, dok su ostala jezgra zauzeta preostalim algoritmima *VQA*.

Izvršavaje SPU programskog koda počinje pozivom funkcije *spe_context_run* iz programskih niti *spu_variance_measure_spawn* i *spu_phase_correlation_spawn*:

```
void *spu_variance_measure_spawn(void *arg) {
    unsigned int entry = SPE_DEFAULT_ENTRY;
    spe_context_run(speVM, &entry, 0, &stVarianceCB, NULL, NULL);
    pthread_exit(NULL);
}

void *spu_phase_correlation_spawn(void *arg) {
    unsigned int entry = SPE_DEFAULT_ENTRY;
    spe_context_run(spePC, &entry, 0, &stPhasecorrCB, NULL, NULL);
    pthread_exit(NULL);
}
```

4.7.2.1 Varijansa

Algoritam za računanje varijanse se izvršava u programskoj niti:

```
int SPU_THREAD(variance_measure)(unsigned long long speid, unsigned long long argp,
unsigned long long envp)
```

Pri računanju varijanse se koriste samo vrednosti luminanse slike. Kako je vrednost luminanse u tački opisana osmobarbitnom vrednošću veličina uzorka za računanje varijanse jednaka je broju tačaka u slici. Ta vrednost za uobičajene formate slike višestruko prevazilazi količinu memorije raspoloživu SPU procesoru. Zbog toga se računanje varijanse obavlja nad delovima slike veličine 16KB. Za pronalaženje poklapanja potrebno je da varijansa svih delova bude ispod unapred utvrđenog praga.

4.7.2.2 Fazna korelacija

Algoritam za računanje fazne korelacije se izvršava u programskoj niti

```
int SPU_THREAD(phase_correlation)(unsigned long long speid, unsigned long long argp,
unsigned long long envp)
```

Za vreme izrade programskog rešenja nije bila dostupna odgovarajuća SPU biblioteka za računanje brze Fourierove transformacije (eng. *Fast Fourier Transformation - FFT*). Kako su ostala SPU jezgra zauzeta drugim algoritmima *VQA* odgovarajuća *FFT* biblioteka se mora izvršavati na istom jezgru sa ostatkom algoritma za računanje fazne korelacije i podržati rad sa podacima u pokretnom zarezu jednostruke preciznosti.

Implementacija vektorizovane *FFT* biblioteke prevazilazi okvir ovog rada pa je zbog toga taj deo računanja fazne korelacije realizovan skalarnim operacijama na SPU jezgru. Zbog arhitekture SPU jezgra ovaj deo programskog koda se izvršava izuzetno sporo što utiče na ukupne performanse programskog rešenja.

5. Ispitivanje i rezultati

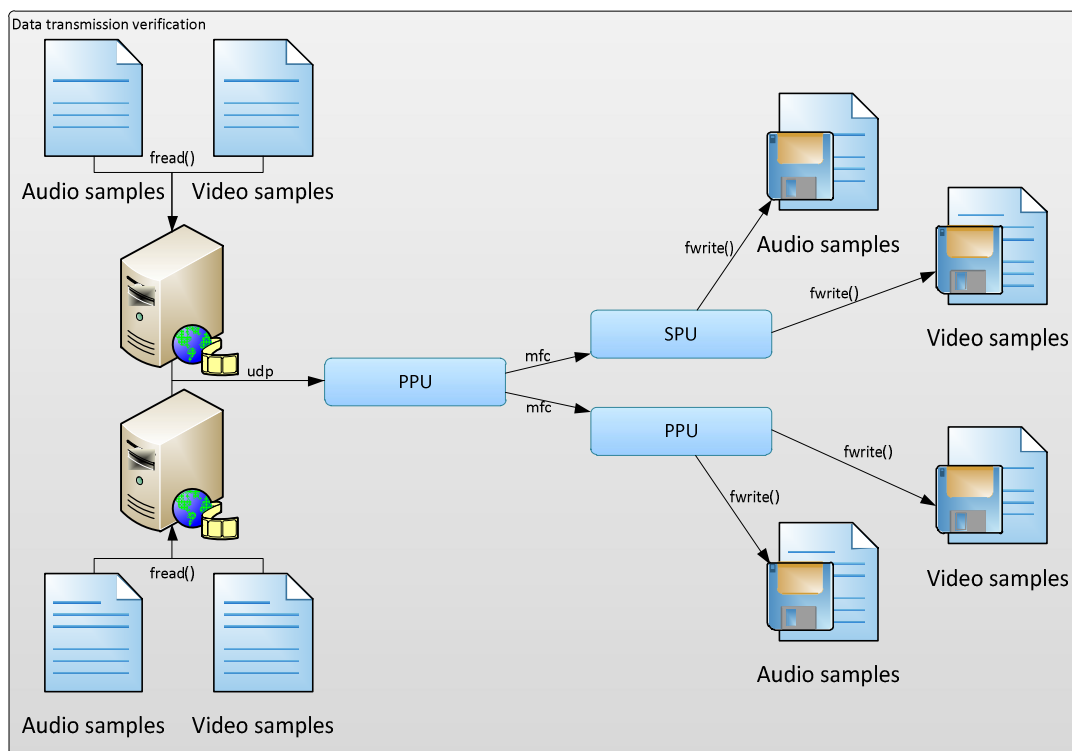
Pored ispitivanja u toku razvoja programskog rešenja opisanog u prethodnim poglavljima, sprovedeno je ispitivanje sa ciljem verifikacije ispravnosti rada programskog rešenja i merenja performansi.

5.1.1 Verifikacija ispravnosti rada

Verifikacija ispravnosti rada vršena je u dva pravca:

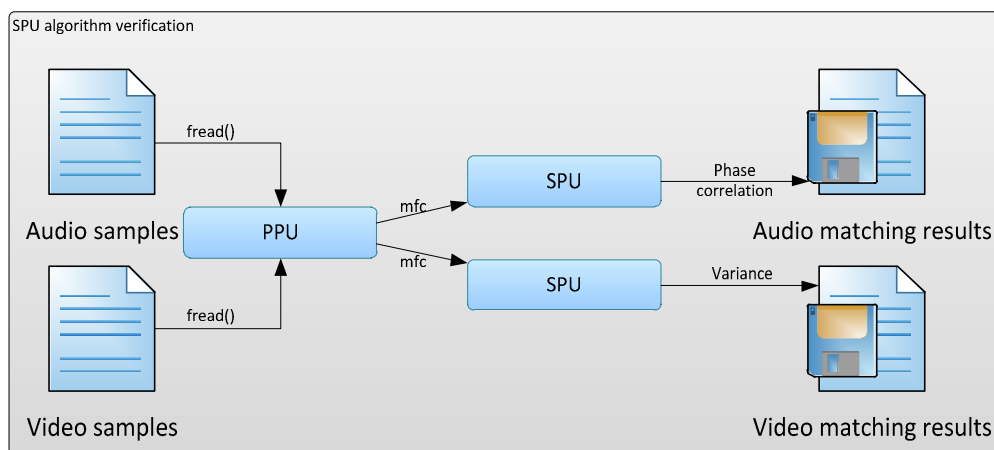
1. verifikacija ispravnog prenosa podataka
2. verifikacija ispravnosti rada algoritma.

Za proveru ispravnosti prenosa podataka programsko rešenje je izmenjeno tako da umesto obrade na SPU procesorima vrši samo upis podataka primljenih od PPU posredstvom DMA. Verifikacija se vrši poređenjem podataka dobijenih na ovaj način sa ulaznim podacima NAViC Simulatora. Rezultat testa je savršeno poklapanje ulaznih i izlaznih podataka što dokazuje ispravan rad programske sprege za prenos podataka. Na slici Slika **25** je prikazana putanja podataka kod ovog testnog slučaja.



Slika 25 – Verifikacija ispravnosti prenosa slike i zvuka

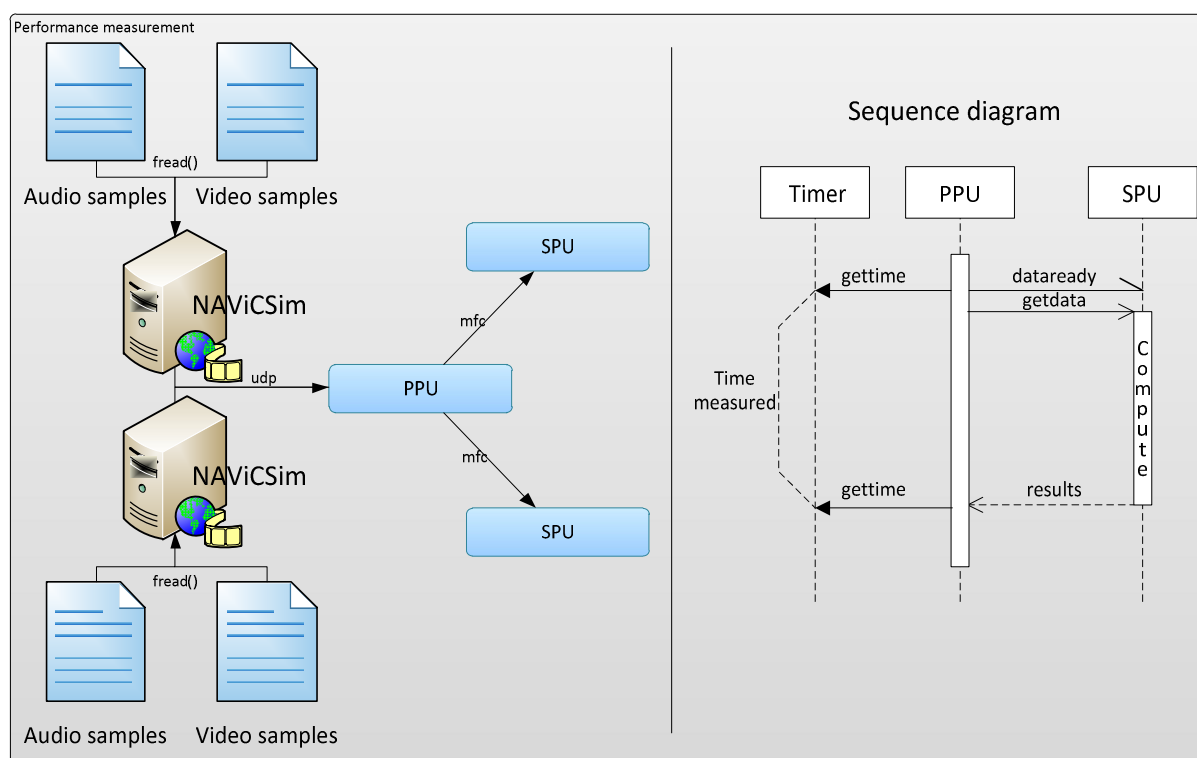
Za proveru ispravnosti rada algoritama napisana je mala PPU aplikacija koja preuzima audio i video odbriske iz datoteke i prosleđuje ih SPU programskom kodu. Rezultati rada SPU programske sprege odnosno koeficijenti fazne korelacije i varijanse se upisuju u datoteku. Poređenje izlaznih podataka sa izlaznim podacima referentnog *Matlab* programskog koda pokazuju neznatno odstupanje uslovljeno greškom zaokruživanja kod operacija u pokretnom zarezu jednostruke preciznosti kod SPU procesora. Naime SPU jedinica za rad sa pokretnim zarezom jednostruke preciznosti nije kompatibilna sa IEEE 754 standardom u smislu da, između ostalog, podržava samo zaokruživanje odsecanjem.



Slika 26 – Verifikacija ispravnog rada algoritama

5.1.2 Merenje performansi

Merenje performansi je vršeno merenjem vremena potrebnog za izvršavanje algoritama. Za merenje performansi rada algoritma korišćena su dva PC kompatibilna računara sa NAViC simulatorima kao izvorima signala. Zbog ograničene procesorske snage nije bilo moguće koristiti sekvence veće od standardne SDTV rezolucije za mrežno slanje signala u realnom vremenu. Ukupno vreme poređenja sadržaja video bafera zavisi od broja poređenja pre nego što se pronađe poklapanje. Zbog toga su za potrebe merenja performansi korišćene potpuno različite sekvence. Na taj način se osigurava da je izmereno vreme obrade u najgorem slučaju tj. kada se se izvrše sva poredjenja bez pronalaska poklapanja. Ilustracija sistema korišćenog za merenje performansi i sekvencni dijagram merenja dati su na slici Slika 27.



Slika 27 – Sistem korišćen za merenje performansi

Zbog nedostatka programske podrške za FFT na SPU jezgrima opisane u poglavlju 4.7.2.2 nije bilo moguće ostvariti rad u realnom vremenu. Vreme obrade za sliku i zvuk dat je tabeli **Tabela 7** – .

Tip signala	Format	Trajanje	Vreme obrade
Video	720x576i, 25fps	400ms	~500ms
Audio	48Khz, 16bit, 2 ch	400ms	~3ms

Tabela 7 – Performanse programskog rešenja

Vreme obrade video signala je veće od trajanja signala pa stoga rad u realnom vremenu nije moguć.

6. Zaključak

U ovom radu je opisana realizacija programskog rešenja za merenje greške sinhronizacije audio-video signala na IBM Cell platformi. Opisan je razvoj algoritama, njihova verifikacija, prilagođenje postojećeg sistema za merenje kvaliteta u cilju testiranja kao i metodologija razvoja SPU programskog koda. Zbog otežanog razvoja i ispravljanja grešaka na Cell platformi razvoj algoritama je vršen u programskom paketu Matlab a zatim je portovan na x86 i Cell arhitekturu. Ispravnost rada svake od verzija programskog koda verifikovan je poređenjem sa referentom Matlab implementacijom koristeći poznat skup ulaznih podataka. Nedostupnost *RT-AV100* uređaja za vreme razvoja uslovalo je adaptaciju *NAViCSim* simulatora za potrebe izvora signala. Upotreba simulatora je umnogome olakšala testiranje jer je moguće izvoditi ponovljive testove sa poznatim skupom ulaznih podataka. Opisani sistem za merenje greške sinhronizacije audio-video je moguće integrisati u postojeće rešenje za ocenu kvaliteta slike *RT-IP030*.

Opisano rešenje je funkcionalno ali ostavlja prostor za ispravku nedostataka. Najveći nedostatak je nemogućnost rada u realnom vremenu. Pored toga nedostatak je i relativno mali merni opseg uslovljen nedostatkom systemske memorije na ciljnoj platformi. PS3 platforma poseduje svega 256MB memorije što ograničava broj odbiraka koje je moguće pohraniti u memoriju, što direktno ograničava merni opseg. Programsko rešenje je ograničeno na 10 slika video materijala i odgovarajuću količinu audio odbiraka. Programsko rešenje je kompatibilno i sa drugim implementacijama Cell procesora koje sadrže veću količinu systemske memorije ali to opisano programsko rešenje ne uzima u obzir. Dinamičko određivanje mernog opsega na osnovu rezolucije ulaznog signala i količine raspoložive memorije bi poboljšalo upotrebljivost postojećeg rešenja.

Vreme računanja FFT na SPU jezgri bez upotrebe vektorizovanog programskog koda je izuzetno dugo. Izmeštanje računanja FTT na PPU jezgro ne donosi dovoljno ubrzanje da bi se ostvario rad u realnom vremenu. Ovaj nedostatak programskog rešenja se može prevazići pisanjem funkcija za brzu Fourierovu transformaciju i inverznu Fourierovu transformaciju prilagođenim arhitekturi SPU.

7. Literatura

- [1] Advanced Television Systems Committee standards, "How to Tackle Lip Sync" The Standard News from ATSC Volume Four, Issue Two, July 2003, http://www.atsc.org/cms/communications/newsletter/2003_july_standard.pdf
- [2] J. C. Cooper, C. Smith, M. Vojnovic, "Audio to video synchronisation errors, sources, measurement and correction", <http://www.pixelinstruments.tv/articles4.htm>, 2011
- [3] Byron Reeves, David Voelker, "*Effects of Audio-Video Asynchrony - Research Report*", 1993, <http://pixelinstruments.tv/articles8.htm>, 2011
- [4] International Telecommunication Union, "*Recommendation ITU-R BT.1359: Relative timing of sound and vision for broadcasting*", 1998
- [5] International Telecommunication Union, "*Recommendation ITU-R BT.1377: Labelling of video and audio apparatus throughput (processing) delay*", 1998
- [6] Advanced Television Systems Committee, "*Document IS-191: ATSC Implementation Subcommittee Finding: Relative Timing of Sound and Vision for Broadcast Operations*", June 2003
- [7] European Broadcasting Union, "*EBU Recommendation R37-2007: The relative timing of the sound and vision components of a television signal*", February 2007
- [8] Cell microprocessor , [http://en.wikipedia.org/wiki/Cell_\(microprocessor\)](http://en.wikipedia.org/wiki/Cell_(microprocessor)), January 2012
- [9] T. Chen, R. Raghavan, J. Dale, E. Iwata, "*Cell Broadband Engine Architecture and its first implementation*", <http://www.ibm.com/developerworks/power/library/pa-cellperf/>, November 2005.
- [10] Dave Krolak, "*Meet the experts: David Krolak on the Cell Broadband Engine EIB bus*", <http://www.ibm.com/developerworks/power/library/pa-expert9/>, December 2005
- [11] Carsten Benthin, Ingo Wald, Michael Scherbaum, Heiko Friedrich "Ray Tracing on the

- CELL Processor”, Proceedings of the IEEE Symposium on Interactive Ray Tracing, pages 15–23, 2006.
- [12] Weisstein, Eric W. "Variance." From MathWorld--A Wolfram Web Resource.
<http://mathworld.wolfram.com/Variance.html>
- [13] C. Kuglin and D. Hines, “The Phase Correlation Image Alignment Method” Proc. Int. Conf. Cybernetics and Society, 1975, pp. 163-165.
- [14] J. P. Lewis, “Fast Normalized Cross-Correlation”, Industrial Light & Magic,
<http://scribblethink.org/Work/nvisionInterface/nip.pdf>
- [15] H. Foroosh (Shekarforoush), J. B. Zerubia, and M. Berthod, “Extension of phase correlation to subpixel registration”, IEEE Transactions on Image Processing, vol. 11, no. 3, pp. 188–200, March 2002.
- [16] C. Dai, Y. Zheng, and X. Li, “Accurate Video Alignment Using Phase Correlation”, IEEE Signal Processing Letters, Vol. 13, No. 12, December 2006
- [17] Roberto Togni, Description of the HuffYUV (HFYU) Codec,
<http://multimedia.cx/huffyuv.txt>
- [18] Wikipedia, Intrinsic function, http://en.wikipedia.org/wiki/Intrinsic_function, May 2013
- [19] IBM, Language Extensions for CBEA, Publication Number v2.6, Revision Date 25.08.2008
<https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/30B3520C93F437AB87257060006FFFE5E>
- [20] IBM, SIMD Math Library Specification for Cell BE, Publication Number: v1.2 , Revision Date: 18.7.2008 ,
<https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/6BFB9899CEA5456800257360001938B3>
- [21] IBM International Technical Support Organization, Programming the Cell Broadband Engine™ Architecture Examples and Best Practices, August 2008,
<http://www.redbooks.ibm.com/redbooks/pdfs/sg247575.pdf>
- [22] Jonathan Bartlett, "Programming high-performance applications on the Cell BE processor", Part 4: Program the SPU for performance ,
<http://www.ibm.com/developerworks/power/library/pa-linuxps3-4/> , Mar 2007
- [23] Matteo Frigo, Steven G. Johnson, "FFTW on the Cell Processor",
<http://www.fftw.org/cell/ps3/>