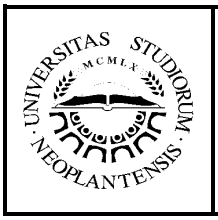


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: **Милош Прањкић**
Број индекса: **RA76/2013**

Тема рада: **Коришћење PCIe сабирнице за дистрибуирану обраду видео сигнала**

Ментор рада: **доц. др Небојша Пјевалица**

Нови Сад, Јул, 2017

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

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Завршни (Bachelor) рад
Аутор, **АУ**:		**Милош Прањкић**
Ментор, **МН**:		**доц. др Небојша Пјевалица**
Наслов рада, **НР**:		**Коришћење PCIe сабирнице за дистрибуирану обраду видео сигнала**
Језик публикације, **ЈП**:		Српски / ћирилица
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2017.
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/29/7/3/17/0/0
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника и рачунарске комуникације
Предметна одредница/Кqучне речи, **ПО**:		Системи за помоћ возачу, Балансирање оптерећења, рачунарски систем у оквиру интегрисаног кола
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У оквиру овог рада представљено је решење за све присутнији проблем недостатка процесорске моћи у алгоритмима за помоћ возачу. Концепт решења јесте да се уместо једног рачунарског система у оквиру интегрисаног кола користи више њих и да се део обраде података премести на неки други рачунарски систем у оквиру интегрисаног кола.
Датум прихватања теме, **ДП**:		**30.06.2017.**
Датум одбране, **ДО**:		07.07.2017.
Чланови комисије, **КО**:	Председник:	доц. др Иван Каштелан
	Члан:	доц. др Немања Лукић — Потпис ментора
	Члан, ментор:	доц. др Небојша Пјевалица

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

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Bachelor Thesis	
Author, **AU**:		**Miloš Pranjkić**	
Mentor, **MN**:		**Nebojša Pjevalica, Phd**	
Title, **TI**:		**Distributed video signal processing over PCIe bus**	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2017	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/29/7/3/17/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Advanced Driver Assistance System, Load Balancing, System on Chip,	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper shows one of the solutions for more and more growing problem of need for processing power. The base concept is to use multiple SoCs instead of one and to offload some processing to the other SoC.	
Accepted by the Scientific Board on, **ASB**:		**30.06.2017.**	
Defended on, **DE**:		07.07.2017.	
Defended Board, **DB**:	President:	Ivan Kaštelan, PhD	
	Member:	Nemanja Lukić, PhD	Menthor's sign
	Member, Mentor:	Nebojša Pjevalica, PhD	

Захвалност

Захваљујем се др Жељку Лукачу и Милени Милошевић на стручној помоћи и саветима приликом израде овог рада.

Посебно се захваљујем члановима своје породице на пруженој подршци током целог школовања.

На крају се захваљујем институту RT-RK, као и свима осталима који су омогућили и на било који начин допринели реализацији овог рада.

САДРЖАЈ

1. Увод 1
2. Теоријске основе 3
2.1 Напредни системи за помоћ возачу – ADAS 3
2.2 Опис ADAS платформе 5
2.3 Основни концепти балансирања оптерећења 6
2.3.1 PCIe и балансирање оптерећења помоћу PCIe 7
2.4 Архитектура окружења за развој програма 10
3. Концепт решења 12
3.1 Подршка за коришћење PCIe сабирнице на различитим нивоима развојног окружења 12
3.2 Карактеристике система за балансирање оптерећења 14
3.3 Комуникација између SoC–ова коришћењем PCIe –а 15
3.3.1 Синхронизација брзине 15
3.3.2 Редослед порука приликом комуникације 16
4. Програмско решење 18
4.1 Опис окружења за развој програмске подршке 18
4.1.1 Архитектура веза и ланаца 18
4.1.2 Развој алгоритама у оквиру развојног окружења 20
4.2 Представљање решења за балансирање оптерећења коришћењем PCIe сабирнице 21
4.2.1 Алгоритми за слање и пријем података коришћењем PCIe -а 21
4.2.2 Креирање ланца за тестирање алгоритама за балансирање оптерећења 21
4.3 Прилагођавање окружења за развој програмске подршке да би се омогућио рад са PCIe сабирницом 23
5. Резултати 25
6. Закључак 28
7. Литература 29

Списак слика

Слика 1 : Приказ ADAS алгоритама у модерним возилима 4
Слика 2 : Alpha ADAS платформа 5
Слика 3 : Блок дијаграм Alpha платформе 6
Слика 4 : Топологија PCIe сабирнице 8
Слика 5 : PCIe комуникација између два уређаја на физичком нивоу 8
Слика 6 : PCIe подсистем на Tda2x SoC-у 9
Слика 7 : Архитектура програмске подршке развојног окружења 11
Слика 8 : Архитектура окружења за развој софтвера која се добија инсталацијом окружења (лево), Архитектура окружења након интеграције PCIe сабирнице до нивоа везе (десно) 13
Слика 9 : Ток података система који користи баланисрање оптерећења 14
Слика 10 : Провера везе пре почетка комуникације 16
Слика 11 : Слање једног оквира 17
Слика 12 : Пример трансфера када су компоненте раздвојене 17
Слика 13 : Ток података за ланац у коме се подаци са камере копирају и приказују на екрану 19
Слика 14 : Ланац који податке са камере шаље преко PCIe сабирнице и приказује на екрану 22
Слика 15 : Ланац који податке примљене преко PCIe сабирнице приказује на екрану 23
Слика 16 : Приказ кашњења приликом слања видеа преко PCIe сабирнице 26
Слика 17 : Приказ кашњења приликом слања само Y компоненте коришћењем PCIe сабирнице 26

Списак табела

Табела 1 : Време трансфера различитих резолуција и формата коришћењем PCIe сабирнице на основу брзине дате у спецификацији 10

Табела 2 : Време трансфера различитих резолуција и формата коришћењем PCIe сабирнице на основу брзине утврђене мерењима 25

Скраћенице

ADAS - **A**dvanced ***D**river **A**ssistance **S**ystems* – Напредни системи за помоћ возачу

AVB - **A**udio **V**ideo **B**ridging – Синхронизација аудио и видео сигнала

Bios - ***B**asic **I**nput-**O**utput **S**ystem* **–** Управљачка програмска подршка која се извршава прва приликом покретања рачунара

BSP - **B**oard **S**upport **P**ackage – Програмски пакети за подршку рада на плочи

DMA - ***D**irect **M**emory **A**ccess* – Директан приступ меморији

DSP - ***D**igital **S**ignal **P**rocessor* – Дигитални Сигнал Процесор

EVE - ***E**mbedded **V**ideo **E**ngine* – Јединица за обраду видео садржаја на ниском и средњем нивоу

GPU - ***G**raphics **P**rocessing **U**nit* – Графичка процесорска јединица

IPC - ***I**nter **P**rocess **C**ommunication* – Комуникација међу процесима

NDK - *Network **D**evelopment **K**it* – Алати за развој мрежне програмске подршке

NSP - *Network **S**upport **P**ackage* – Развојни пакети који подржавају развој мрежне програмске подршке

PCI - ***P**eripheral **C**omponent **I**nterconnect* - периферна комуникациона веза

PCIe - ***P**eripheral **C**omponent **I**nterconnect **E**xpress* - периферна комуникациона брза веза

SBL - *The **S**econdary **B**oot**l**oader* – Иницијални пуњач другог степена

SDK - ***S**oftware **D**evelopment **K**it* - Окружење за развој програмске подршке

SoC - ***S**ystem **o**n **C**hip* – Рачунарски систем у оквиру интегрисаног кола

VIN - ***V**ideo **In**put* – Видео улаз

VLIB - ***V**ideo Analytics and **V**ision **Lib**rary* – Библиотека програмске подршке за анализу и обраду слике

VSDK - ***V****ision* ***S****oftware* ***D****evelopment* ***K****it* – Окружење за развој програма

1. Увод

Почетком развоја аутомобилске индустрије аутомобили су били машине којима је човек у потпуности управљао. Пошто су временом увиђане несавршености човека почела је да се развија идеја да би аутомобил могао да исправља грешке возача и на тај начин доприноси безбеднијој вожњи.

Први системи за помоћ возачу су били врло једноставни и нису захтевали превише обраде података. Данас имамо изузетно комплексне алгоритме од којих неки имају могућност да у потпуности преузму контролу над возилом. За такве алгоритме је веома битно да имају довољно процесорске моћи за рад у реалном времену јер људски животи сада зависе од њих.

Да би комплексни алгоритми који се користе у апликацијама напредних система за помоћ возачу при вожњи (енг. ADAS, ***A**dvanced **D**river **A**ssistance **S**ystems*) [1] могли да раде у реалном времену потребно је да имају одговарајућу физичку архитектуру на којој ће се извршавати. Свака наредна генерација аутомобила доноси нове ADAS алгоритме. Као што је већ познато брзина развоја алгоритама је много већа од брзине развоја физичке архитектуре, тако да се може догодити да следећа генерација алгоритама захтева већу процесорску моћ него претходна и да не може да се извршава на физичкој архитектури на којој се претходна генерација алгоритама извршавала. Оваква чињеница значајно успорава развој крајњег производа – аутомобила. Затим, када се развије физичка архитектура која подржава извршавање нових алгоритама, реализују се новији алгоритми и сав процес се враћа на почетак.

У овом раду је приказано једно од решења за проблем недостатка процесорске моћи система са једним системом на чипу (енг. SoC, ***S**ystem **o**n **C**hip*) чији ресурси нису

довољни за покретање одређених алгоритама. Решење је реализовано у склопу развојног окружења (енг. VSDK, *Vision Software Development Kit*) компаније Texas Instruments, [2] у програмском језику C.

Рад је организован у следеће целине:

- Теоријске основе – У оквиру овог поглавља дат је преглед ADAS алгоритама, основне карактеристике физичке архитекруре на којима се извршавају ADAS апликације. Описан је концепт уравнотежења оптерећења процесорских компонената као и основне карактеристике PCIe сабирнице која ће се користити за уравнотежење;
- Концепт решења – Представљене су основне карактеристике реализованаг система за уравнотежење оптерећења;
- Програмско решење – Кратак опис развојног окружења и начина на које је само решење уграђено у развојно окружење;
- Тестирање и резултати – Приказ резултата и заузеће процесора приликом коришћења PCIe сабирнице;
- Закључак – Кратак осврт на резултате и начини на које се може унапредити добијено решење.

2. Теоријске основе

У овом поглављу дат је приказ основних напредних система за помоћ возачу (енг. ***A**dvanced **D**river **A**ssistance **S**ystems*) као и опис типичне платформе на којој раде ADAS апликације. Поред тога биће представљен концепт уравнотежења оптерећења (енг. *Load balancing*) процесорских компонената, у даљем тексту балансирања, као и периферна комуникациона брза веза (енг. PCIe, ***P**eripheral **C**omponent **I**nterconnect **E**xpress*) [3] преко које ће балансирање бити реализовано.

2.1 Напредни системи за помоћ возачу – ADAS

Напредни системи за помоћ возачу-ADAS развијани су како би вожњу учинили удобнијом, опуштенијом и сигурнијом како за путнике у возилу тако и за возача. Познато је да човек није савршен и да је подложан грешкама. Чак и приликом хода човеку се може догодити да запне за неки предмет у својој околини услед недостатка пажње. Последице су неупоредиво горе у потпуно неприродном окружењу попут аутомобила који се креће у просеку 10 до 30 пута брже него човек у ходу. Због тога је од велике важности да аутомобил има могућност да заштити све путнике у возилу од потенцијалног пропуста возача.

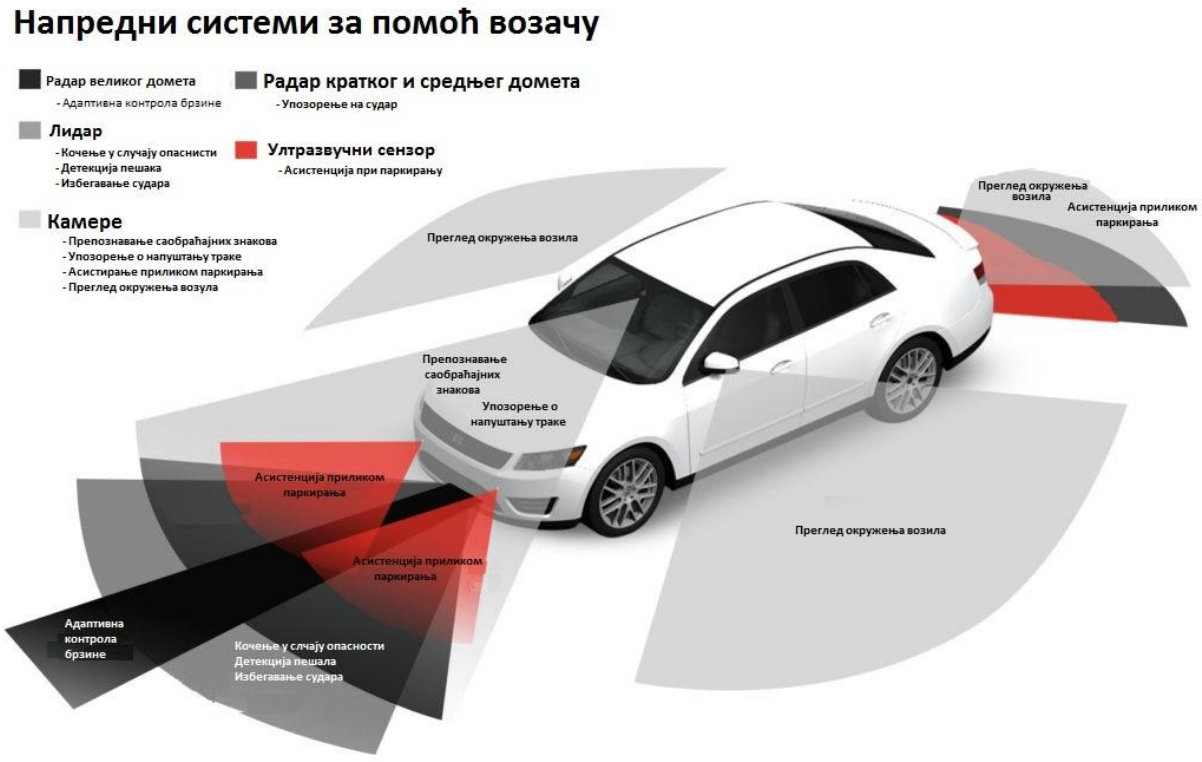
У модерним аутомобилима постоји мноштво сигурносних система који прате услове на путу и благовремено реагују како би обавестили возача или, уколико је то потребно, преузели контролу над возилом како би отклонили могућност нежељеног исхода.

Данас постоји мноштво напредних система за помоћ возачу при вожњи. Системи су углавно независни једни у односу на друге али се њихови резултати комбинују како би се добила целокупна представа стања у возилу и његовом окружењу.

Неки од система су:

- Препознавање саобраћајне траке којом се возило креће,
- Препознавање саобраћајних знакова,
- Препознавање пешака и других препрека на путу,
- Аутоматска контрола брзине (са могућношћу праћења брзине возила испред),
- Аутоматско кочење у случају препреке на путу,
- Приказ окружења возила,
- Препознавање возила која се приближавају из позадине,
- Систем за процену пажње возача.

Ови системи сакупљају информације о окружењу са различитих сензора карактеристичних за сваки појединачни систем или групу система. Тако на пример систем за помоћ приликом паркирања комбинује податке прикупљене са камера како би возачу приказао окружење возила у комбинацији са ултразвучним сензорима који пружају информацију о удаљености препрека од возила. Још један пример представља систем за препознавање пешака који поред камере која снима простор испред возила, прикупља податке и са лидара. На слици испод може се видети који системи за помоћ возачу користе које изворе информација.



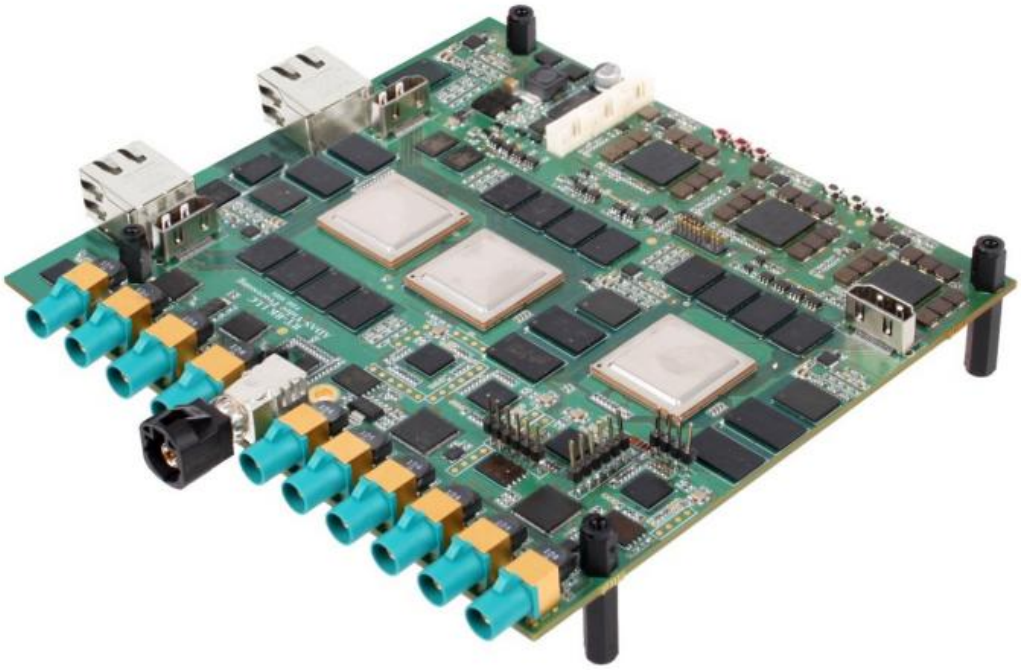
Слика 1 : Приказ ADAS алгоритама у модерним возилима

2.2 Опис ADAS платформе

Када је реч о платформама које се користе за ADAS апликације, њихова физичка архитектура је углавном слична. Централна компонента ADAS платформе је рачунарски систем у оквиру интегрисаног кола (енг. SoC, ***S**ystem **o**n **C**hip*), који се углавном састоји из више различитих процесора. Присутан је најчешће један процесор опште намене који је обично задужен за управљање целим SoC-ом. Поред њега, зависно од намене, могу се наћи дигитални сигнал процесори (енг. DSP, ***D**igital **S**ignal **P**rocessor*), графичка процесорска јединица (енг. *GPU, **G**raphics **P**rocessing **U**nit*) као и још неки процесори специјализовани за одређене обраде података које захтевају ADAS апликације.

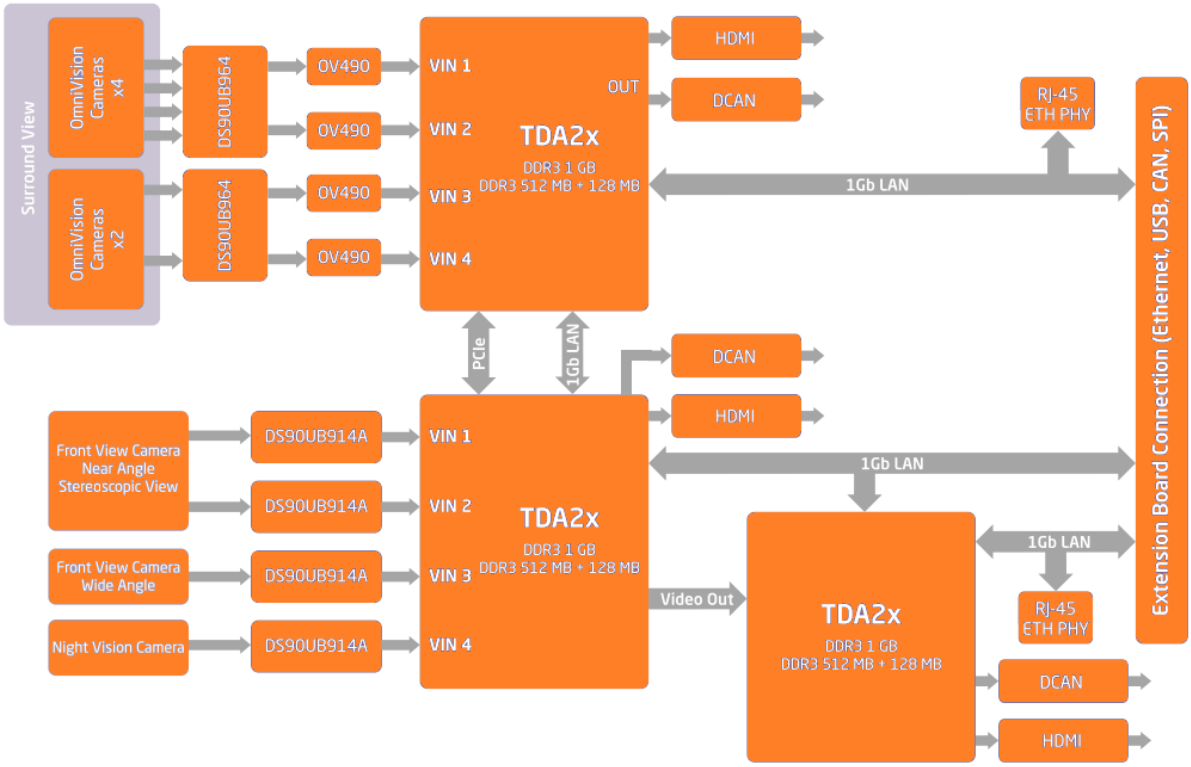
Када је реч о периферијама које се користе на платформама, сама примена као главне намеће камере и остале сензоре. Поред тога постоји могућност да платформа комуницира са околином. Зависно од захтеване брзине и поузданости, за комуникацију са околином може се користити све од Етернета па до комуникационих протокола специфичних за аутомобил.

За реализацију коришћена је Alpha [4] развојна платформа. Платформа је развијена на РТ-РК институту за потребе ADAS-а.



Слика 2 : Alpha ADAS платформа

Платформа садржи три TDA2xx [5] SoC-а произвођача Texas Instruments. Први SoC назива се још и SCV (енг. ***S**urround **C**amera **V**iew*). Повезан је на шест канала видео улаза (енг. VIN, ***V**ideo **IN**put*) и као што му и само име каже користи се за приказ околине возила. Други SoC се назива FFN (енг. ***F**ront View Camera Near Angle Stereoscopic View, **F**ront View Camera Wide Angle, **N**ight Vision Camera*). Користи се за снимање пута испред возила. Трећи TDA2xx SoC назива се FUS (енг. ***FUS**ion*) и користи се да сакупља и приказује возачу резултате рада са прва два SoC-а.



Слика 3 : Блок дијаграм Alpha платформе

2.3 Основни концепти балансирања оптерећења

Балансирање оптерећења представља једно од решења за проблем недовољне процесорске моћи једног SoC-а. Друго решење коме се прибегава јесте да се са сваком захтевнијом генерацијом ADAS система користи SoC веће процесорске моћи и бољих перформанси. Тај приступ је сличан ономе који је користила целокупна рачунарска индустрија док није дошла до зида снаге (енг. Power Wall). Код оваквог приступа може се појавити неколико проблема. Када је потребно у постојећи алгоритам унети извесну измену, може се догодити да SoC на коме се до датог тренутка извршавао алгоритам не поседује довољну процесорску моћ да би се извршавала унапређена верзија у реалном времену. Решење за то може бити коришћење неког другог SoC-а који поседује

довољну процесорску моћ. Овакав подухват углавном није једноставан, јер изискује упознатост са новим SoC-ом, процесорима које SoC поседује, начинима за оптимизацију кода и свим другим особеностима, као и за стари SoC. Наредна потенцијално лоша околност јесте то да процесори веће снаге обично користе и више енергије јер раде на вишем такту, а познато је да се у уграђеним системима потрошња покушава умањити што је више могуће.

Код балансирања оптерећења приступ је другачији. Користи се више SoC-ова уместо једног који поседује велику процесорску моћ. На овај начин добија се проширив систем који може да реши проблем мањка перформанси. За овакав приступ обично се користе SoC-ови исте врсте, тако да није потребно упознавање са новим. Овај начин са собом може донети и неке скривене проблеме. Врло је битно прецизно испланирати балансирање, јер уколико се то не уради само ће се повећавати комплексност система, док се са становишта перформанси неће остварити велики добитак. У неким екстремним случајевима када се у систем интегрише више SoC-ова може чак доћи и до смањења перформанси.

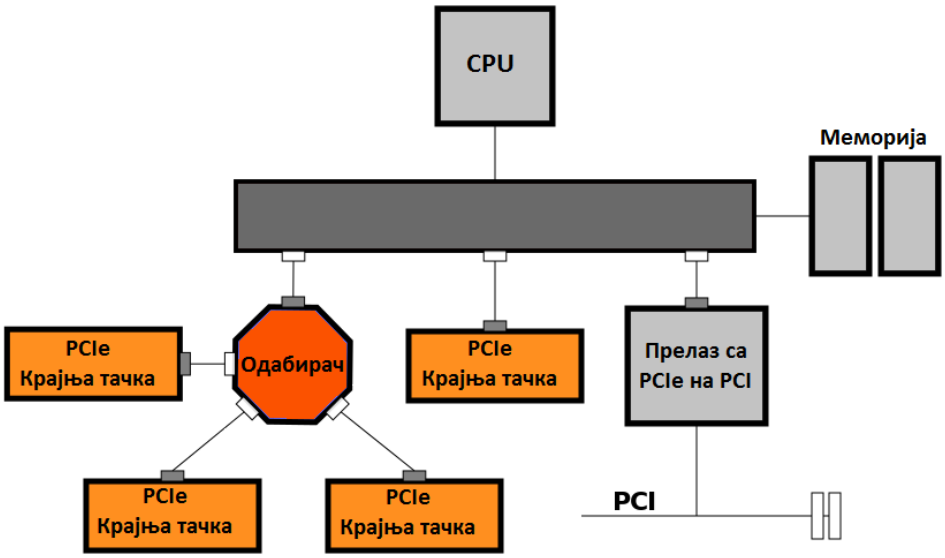
По природи концепта балансирања оптерећења најважније је одлучити се како ће се радити сама комуникација и пренос података ради растерећивања између SoC-ова. За ову примену могу се развити потпуно нови комуникациони протоколи специјализовани за SoC који се корити или се могу користити неки већ постојећи који користе добро познате комуникационе протоколе. Обично се одлучује да се користе већ постојећи како би се уштедело време потребно за развој нових и како би било могуће у систем укључити и неке друге уређаје који поседују те спреге. Уколико су инжењери већ упознати са протоколом, та чињеница такође убрзава развој.

Када се одлучује који ће се протокол користити најбитнији критеријум је пропусни опсег који је потребан за реализацију. Најпре је потребно имати на уму количину података за коју треба растеретити SoC. Затим је важно увидети да ли систем треба да ради у реалном времену или не. На пример, коришћење Етернета за балансирање оптерећења система који раде у реалном времену није добар избор уколико не постоји могућност контроле саобраћаја на мрежи. За системе који раде у реалном времену најбоље је користити протоколе који не могу да се преоптерете саобраћајем.

2.3.1 PCIe и балансирање оптерећења помоћу PCIe

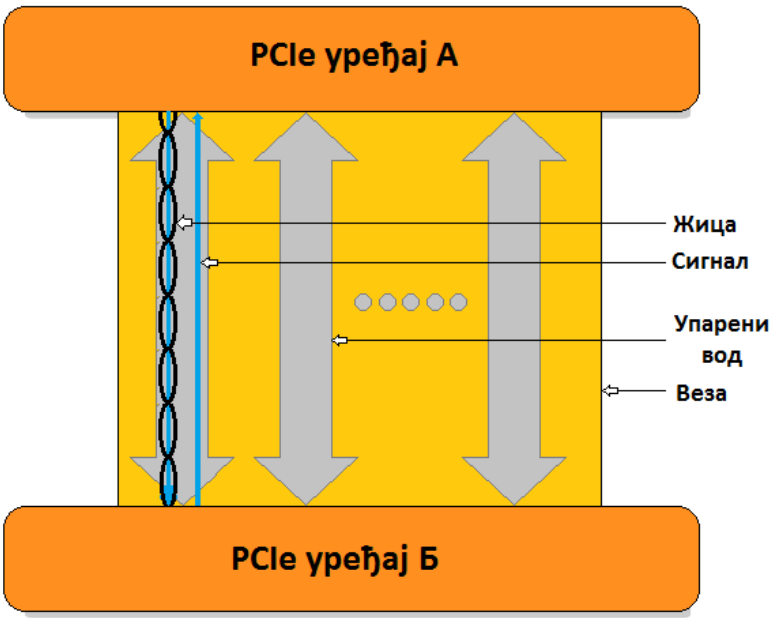
PCIe (***P**eripheral **C**omponent **I**nterconnect **E**xpress*) је сабирница велике брзине која наслеђује старију, PCI (енг. ***P**eripheral **C**omponent **I**nterconnect)* сабирницу. Главна разлика је у томе што PCI користи сабирницу са паралелним водовима и сви уређаји су

прикључени на исту сабирницу, док је PCIe базиран на тачка-тачка топологији и сваки од уређаја на сабирници је повезан засебном серијском везом. Пример PCIe топологије је приказан на слици испод.



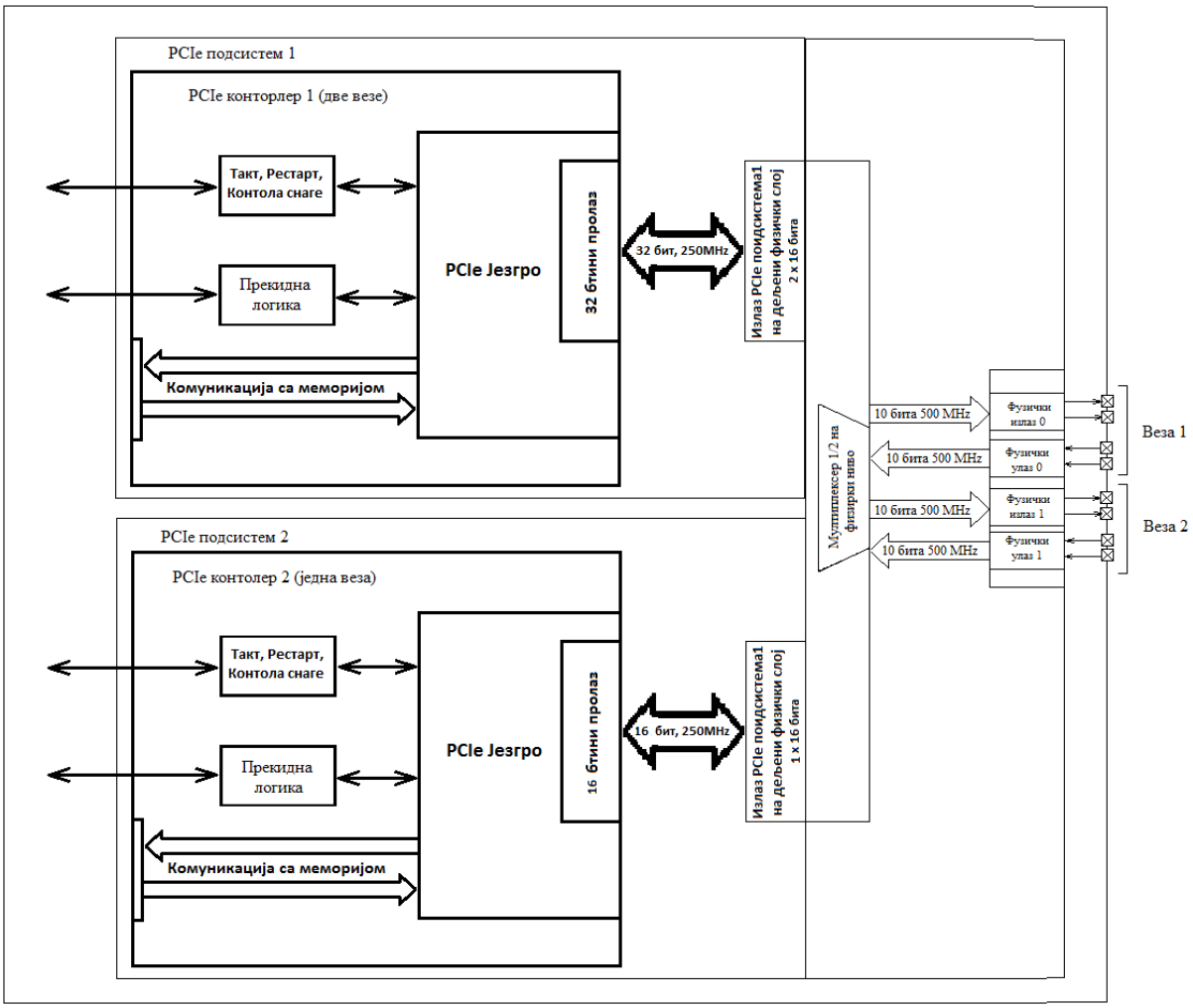
Слика 4 : Топологија PCIe сабирнице

Свака PCIe веза на физичком нивоу се састоји из више упарених водова. Број водова зависи од генерације PCIe. Унутар упареног вода налазе се сигнали за комуникацију. У сваком воду налазе се по два сигнала за двосмерну комуникацију. Водови су диференцијални, што значи да се састоје из 2 физичке везе – жице.



Слика 5 : PCIe комуникација између два уређаја на физичком нивоу

TDA2x SoC поседује 2 PCIe подсистема. Први поседује 2 упарена вода и може бити конфигурисан у режим једносмерне или двосмерне комуникације. Други подсистем поседује један упарени вод. Пошто деле исте физичке ресурсе, уколико се први подсистем конфигурише да ради са два упарена мода, други подсистем се не може користити.



Слика 6 : PCIe подсистем на TDA2x SoC-у

Као што је познато, коришћењем PCIe сабирнице могуће је пренети велике количине података, што је чини погодном за коришћење у системима за рад у реалном времену. Теоријска максимална брзина преноса која се може остварити коришћењем једног упареног вода PCIe сабирнице је око 600 MB/s.

У оквиру ADAS апликација обично се обрада ради над видео садржајем, тако да је управо видео садржај оно што треба да се пребацује коришћењем PCIe-а. Пошто камере на ALPHA платформи имају могућност бележења видео садржаја у више

формата, размотрено је колики пропусни опсег је потребан да би се обавио пренос видео слика између два TDA2x SoC-а, као и да ли је пропусни опсег који може да се оствари на PCIe подсистему TDA2x SoC-а довољан да се задовоље потребе реалног времена.

Формат видеа	*Резолуција [пиксел]*	*Оквири у секунди*	*Величина оквира*	*Величина 1 с видеа*	*Могућност трансфера у два правца у реалном времену*
YUV422	1280x720	30	1,757 МБ	52,71 МБ	Да
YUV422	1280x720	60	1,757 МБ	105,42 МБ	Да
YUV422	1920x1080	30	4,078 МБ	122,34 МБ	Да
YUV422	1920x1080	60	4,078 МБ	244,68 МБ	Да
YUV420	1280x720	30	1,318 МБ	39,54 МБ	Да
YUV420	1280x720	60	1,318 МБ	79,08 МБ	Да
YUV420	1920x1080	30	2.966 МБ	88,98 МБ	Да
YUV420	1920x1080	60	2.966 МБ	177,96 МБ	Да
RGB444	1280x720	30	2,636 МБ	79,08 МБ	Да
RGB444	1280x720	60	2,636 МБ	158,16 МБ	Да
RGB444	1920x1080	30	5,932 МБ	177,96 МБ	Да
RGB444	1920x1080	60	5,932 МБ	355,92 МБ	Да

Табела 1 : Време трансфера различитих резолуција и формата коришћењем PCIe сабирнице на основу брзине дате у спецификацији

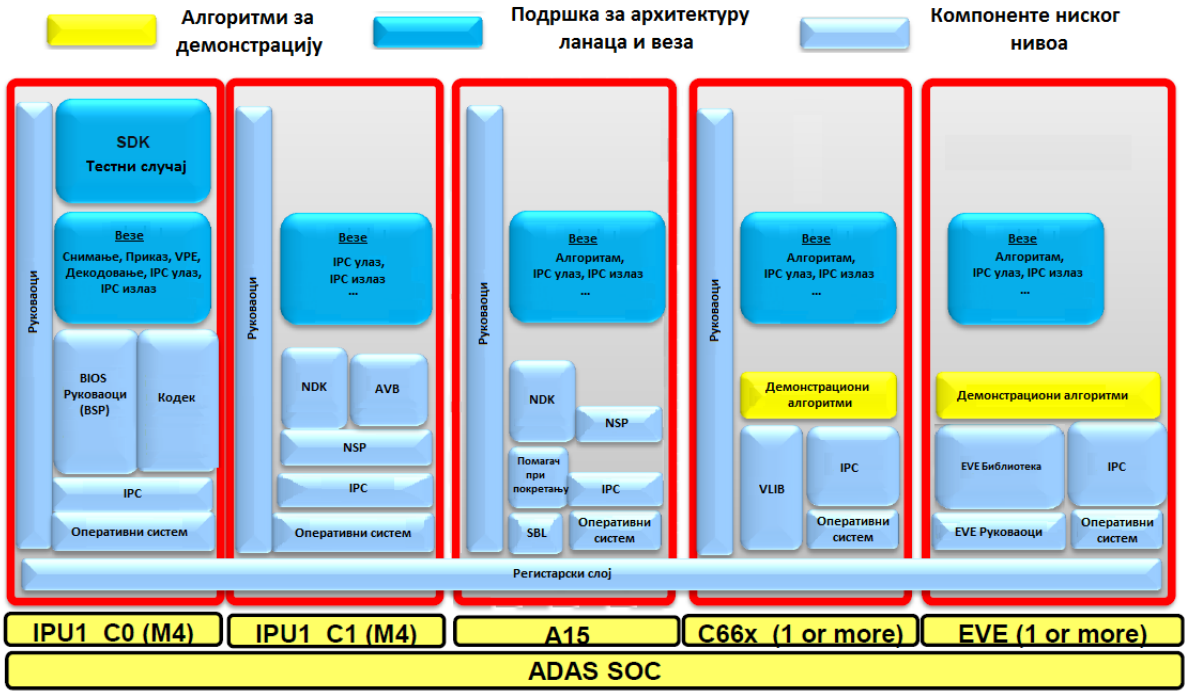
2.4 Архитектура окружења за развој програма

Окружење за развој програма које је коришћено за реализацију овог рада је VSDK произвођача Texas Instruments. У оквиру овог окружења присутан је читав ланац програмских компонената (енг. *toolchain*) за генерисање извршног кода за све процесоре који су присутни на TDA2x SoC-у. Архитектура окружења се састоји из две велике целине.

Прва целина обухвата програмску подршку нижег нивоа која је у тесној вези са конкретном платформом. Ту се налазе преводиоци програмског кода као и сви остали делови програмске подршке за успешно генерисање извршивих датотека, као и компоненте неопходне за олакшавање отклањања програмских грешака (енг. *debugging*) током извршавања. Поред тога, присутни су и руковаоци (енг. *drivers*) за

управљање периферијама на ниском нивоу, почевши од пакета за развој програмске подршке без коришћења оперативног система, па све до пакета за подршку физичке архитектуре.

Друга целина се назива vision_sdk. У оквиру ове целине налази се све оно што је потребно за успешан развој алгоритама, почевши од алата који генерише кориснички случај, ондосно пут података кроз задате алгоритме, подршку за коришћење комуникације међу процесима који се налазе на различитим процесорима, подршку за развој и састављање нових алгоритама, као и мноштво других ствари које унапређују и олакшавају развој.



Слика 7 : Архитектура програмске подршке развојног окружења

На слици изнад налази се схема архитектуре програмске подршке развојног окружења. Она је битна због лакшег схватања корака учињених ради реализације балансирања садржаја.

3. Концепт решења

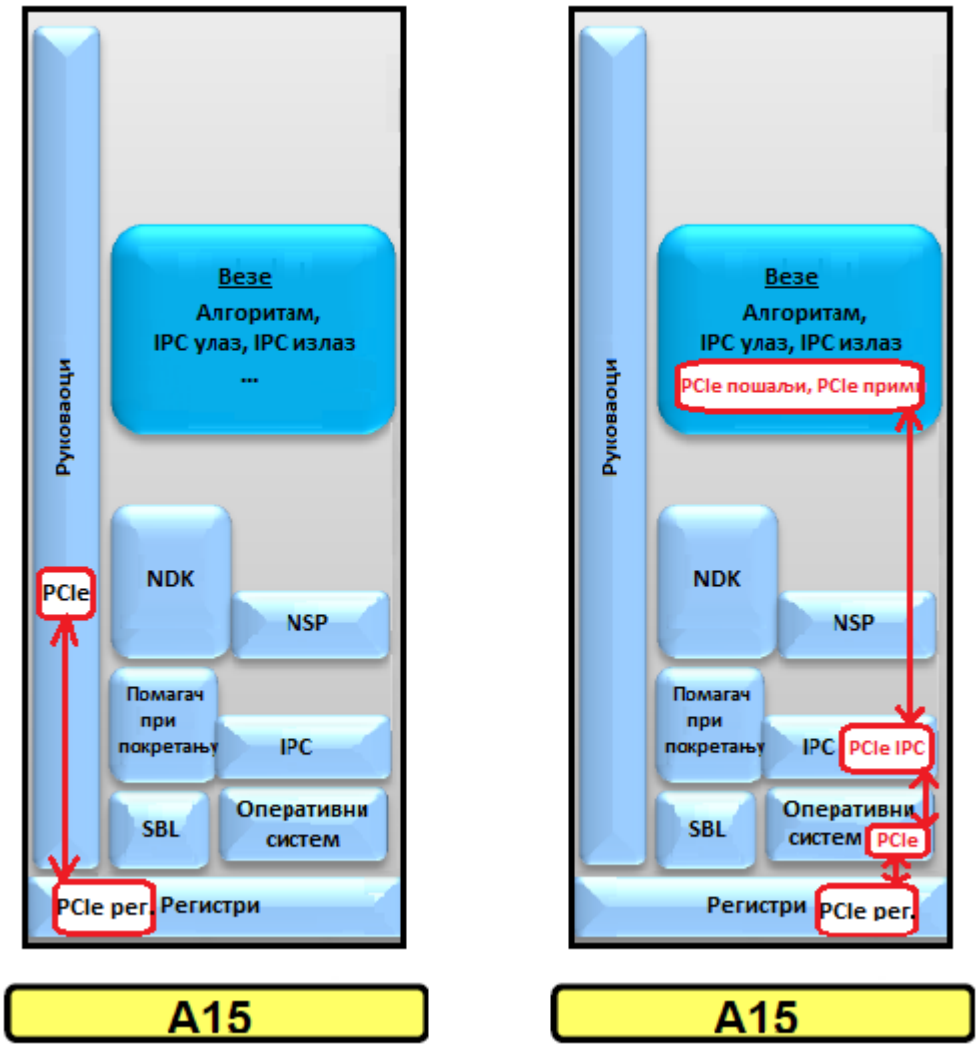
У овом поглављу приказана је идеја за реализацију размене података између два SoC-а ради балансирања оптерећења и начин на који би се инжењеру олакшао развој апликација које користе могућност балансирања оптерећења ради превазилажења проблема недостатка процесорске моћи.

3.1 Подршка за коришћење PCIe сабирнице на различитим нивоима развојног окружења

У оквиру развојног окружења постоје везе које се користе за развој случаја коришћења (енг. *use case*). То су на пример веза (енг. *link*) за снимање садржаја, веза за приказ на екран, веза која представља неки алгоритам, итд. Те везе се налазе на највишем нивоу у хијерархији програмске подршке, што значи да се ослањају на све нивое испод њих и у мањој или већој мери и зависе од њих.

Као што на највишем нивоу у програмској хијерархији постоје везе које се користе за састављање тока података у алгоритмима као и демонстрациони примери у којима се показује како се те везе користе и који је резултат њиховог рада, тако и на осталим нивоима постоје примери за коришћење неких компоненти у систему. Комплексност тих примера зависи од њихове позиције у архитектури програмске подршке. Примери на нивоу везе су најкомплекснији док су они на нивоу руковалаца најједноставнији. Што се тиче PCIe сабирнице, пример за њено коришћење у оквиру програмског окружења се налази на најнижем нивоу, нивоу руковалаца који раде без посредства оперативног система. Пример се састоји од две врло једноставне апликације од којих једна шаље извесне податке другој, друга их прима и шаље назад, првој.

Употреба PCIe сабирнице на овом ниоу није од велике користи, пошто се сви алгоритми развијају и тестирају на нивоу везе и велика количина програмске подршке води рачуна о њиховом извршавању, комуникацији, синхронизацији и свему осталом што на нивоу руковалаца није подржано.



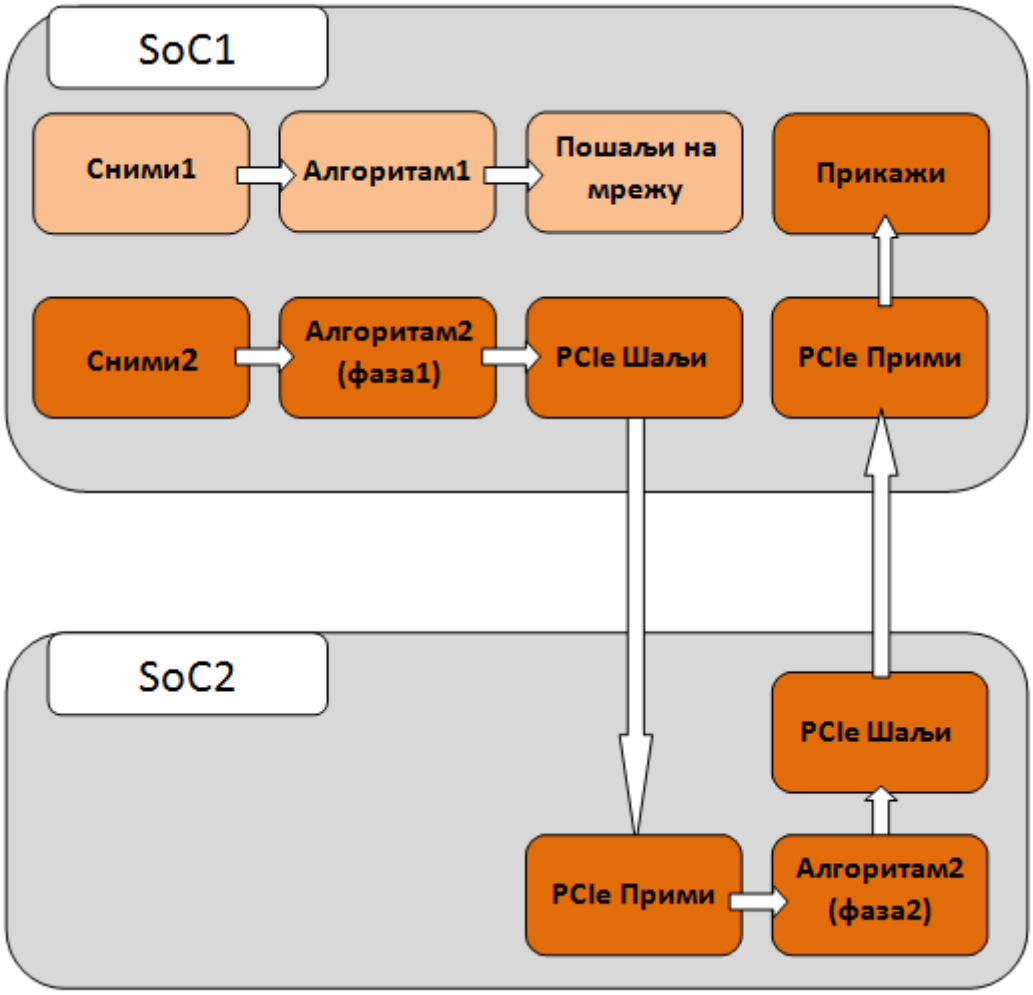
Слика 8 : Архитектура окружења за развој софтвера која се добија инсталацијом окружења (лево), Архитектура окружења након интеграције PCIe сабирнице до нивоа везе (десно)

Ова интеграција представља најсложенији део имплементације балансирања оптерећења иако није повезана са самим концептом. Тек када се управљање PCIe сабирницом реализује на сваком од нивоа програмске архитектуре за развој софтвера, добија се потпуно решење које се уклапа у развојно окружење и при томе се у потпуности задржава модуларност и проширивост развојног окружења.

3.2 Карактеристике система за балансирање оптерећења

Основна идеја система за балансирање оптерећења јесте омогућавање извршавања захтевних алгоритама и апликација на више SoC-ова. Реализацију оваквог система потребно је добро осмислити. Циљ целокупног поступка је да инжењер који треба да развија алгоритме на платформи која има могућност балансирања оптерећења не губи превише времена на само балансирање и правилно коришћење система за балансирање већ да стекне утисак да развија програм код кога се све одвија на једном SoC-у. Због тога мора постојати добра програмска подршка која би маскирала само балансирање.

У идеалном случају било би могуће једноставно написати изворни код и означити који делови алгоритма се извршавају на ком SoC-у. На тај начин се не би допринело сложености поступка самог развоја и сва пажња би могла да буде усмерена на развој самих алгоритама.



Слика 9 : Ток података система који користи балансирање оптерећења

Псеудо код дат у наставку, заједно са Сликом 9, описује како би у идеалном случају балансирање оптерећења требало да буде поједностављено.

```
Сними1 (SoC1) → Алгоритам1 (SoC1)  → Пошаљи на мрежу (SoC1)

Сними2 (SoC1) → Алгоритам2 прва фаза (SoC1)  → Пошањи путем PCIe (SoC1) → Прими преко PCIe (SoC2) → Алгоритам2 друга фаза (SoC2) → Пошањи путем PCIe (SoC2) → Прими преко PCIe (SoC1) → Прикажи (SoC1)
```

Пошто је утврђено да компонента SoC1 не поседује довољу процесорску моћ за покретање два алгоритма, као решење се намеће балансирање оптерећења додавањем компоненте SoC2. У том случају се на компоненти SoC1 извршава један цео алгоритам који преузима податке са камера, врши обраду и обрађене податке прослеђује на мрежу, а такође и половина другог алгоритма, која такође добавља улазне податке са камера, извршава прву фазу обраде, обрађене податке прослеђује на SoC2, који ради другу фазу обраде, прима податке од SoC2 и приказује резултате обраде. Други SoC задужен је да прихвати податке од првог, одради другу фазу обраде и да врати потпуно обрађене податке првом SoC-у.

Са оваквим приступом инжењер не би имао осећај да пише програмски код који ће се извршавати на више језгара јер је сам механизам балансирања реализован за њега. Самим тим, сву своју пажњу може да посвети развијању алгоритама.

3.3 Комуникација између SoC–ова коришћењем PCIe –а

Приликом реализације система за балансирање оптерећења мора се водити рачуна о многим ставкама, а једна од најважнијих јесте синхронизација и начин комуникације пријемне и предајне стране.

3.3.1 Синхронизација брзине

Да би систем за балансирање оптерећења могао исправно да функционише пре почетка комуникације, пријемна и предајна страна морају бити конфигурисане да тако да користе исту брзину.

На TDA2x SoC-у PCIe подсистем подржава две брзине преноса. Прва брзина је GEN 1 која износи 2,5 Gb/s и користи једну физичку траку за пренос података. Подсистем 2 може бити конфигурисан само за рад у овом моду. Друга брзина, GEN 2, има могућност размене података брзином од 5 Gb/s и само подсистем 1 може да је

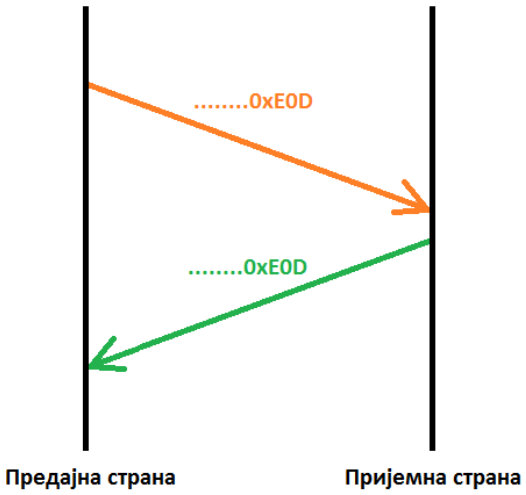
користи јер само он има могућност да комуницира користећи обе траке за пренос података.

3.3.2 Редослед порука приликом комуникације

Следећи проблем који мора бити решен је добро познат проблем произвођача и потрошача. Овај проблем настаје када брзина обраде није усклађена. На пример, ако предајна страна може да врши слање података брже него што пријемна може да их преузима и обрађује, може се догодити да се зависно од реализације послати подаци или препишу новим подацима или да дође до презасићења пријемне стране подацима. Други случај је да пријемна страна очекује нове податке у сувише кратким временским интервалима, унутар којих предајна страна не може да их обезбеди,што може довести до читања старих података или чак и до отказивања читавог система.

У апликацијама које поможу возачу приликом вожње је од изузетног значаја да не дође ни до једног од ових проблема и због тога посебна пажња мора да се обрати приликом дизајнирања и реализације комуникације.

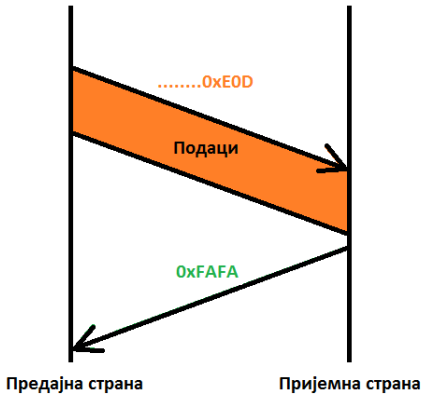
У реализација која је овде представљена редослед порука приликом комуникације је дефинисан на следећи начин. Пре почетка слања било каквих података приликом иницијализације система прво се обавља провера комуникације слањем конторлних порука. Најпре предајна страна шаље пријемној поруку, пријемна након што добије поруку исту поново прослеђује предајној страни и тек када предајна страна добије поруку сматра се да је комуникација успешна и може да се настави са даљом иницијлизацијом система.



Слика 10 : Провера везе пре почетка комуникације

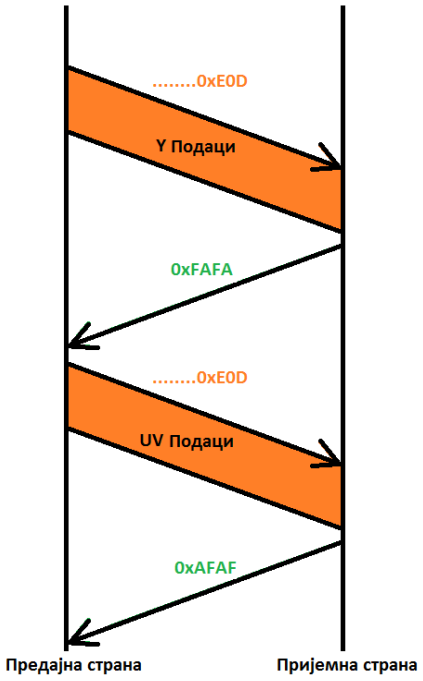
Након успешне иницијализације потврђено је да су предајна и пријемна страна добро конфигурисане и да пренос може да почне.

Слање се одвија тако што предајна страна шаље оквир по оквир и након сваког послатог оквира мора да чека потврду о пријему истог од пријемне стране. На овај начин не може се десити да се пријемна страна преоптерети нити да се подаци препишу новим подацима пре него што су прочитани.



Слика 11 : Слање једног оквира

Да би решење било што је могуће општије и да би могло да се користи за све формате слика, уколико се све компоненте слике не налазе заједно свака компонента се шаље посебно. Пример за такав случај је формат YUV420 SP, Код кога је Y компонента раздвојена од U и V.



Слика 12 : Пример трансфера када су компоненте раздвојене

4. Програмско решење

У овом поглављу биће представљено окружење за развој програмске подршке за Tda2x SoC (енг. VSDK, *Vision Software Development Kit*) како би се што лакше разумело програмско решење које ће након тога бити описано.

4.1 Опис окружења за развој програмске подршке

Окружење за развој програмске подршке које је коршћено за реализацију балансирања оптерећења помоћу PCIe сабирнице се заснива на принципу веза и ланаца (енг. links and chains).

4.1.1 Архитектура веза и ланаца

Архитектура ланаца и веза се користи за развој апликација у оквиру развојног окружења. Везе представљају неку обраду над подацима док се повезивањем веза добија ланац који представља ток података.

У оквиру окружења постоји развијен механизам за врло једноставно креирање оваквих ланаца. Могуће је направити ланац чији се поједини делови извршавају на различитим процесорима и прилком тога не мора превише пажње да се посвећује комуникацији међу процесима (енг. IPC, *Inter Process Communication*) већ тај део посла окружење обавља уместо нас. У наставку следи пример креирања једног ланца.

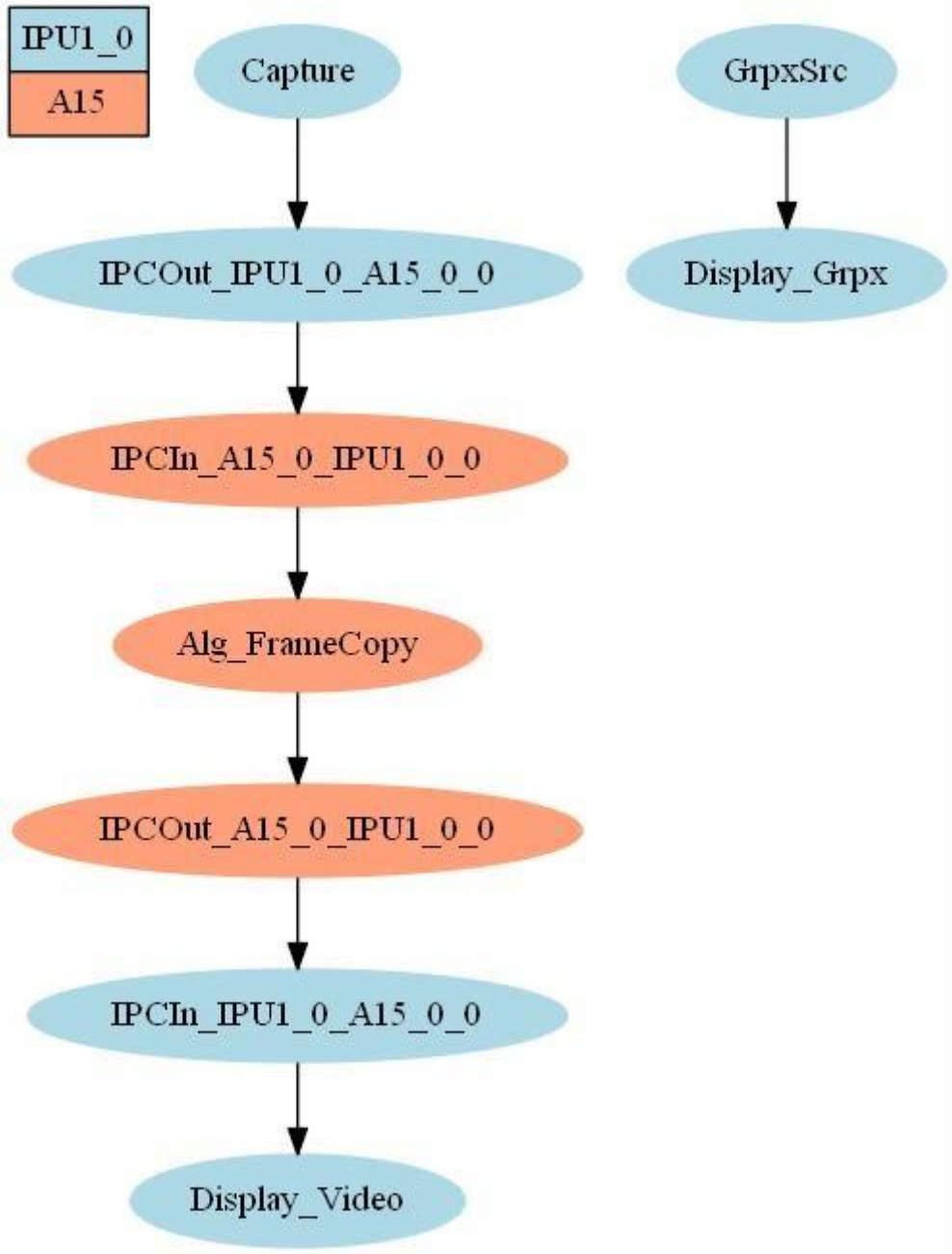
Најпре треба креирати текстуалну датотеку помоћу које нам генератор за прављење ланаца генерише све остало. Текстуална датотека за једноставан ланац који се зове “ chains_vipSingleCameraFrameCopy “, и који податке снимљене са камере копира и прослеђује да се прикажу на екрану изгледа овако :

UseCase: FrameCopy

Capture -> Alg_FrameCopy (A15) -> Display_Video

GrpxSrc -> Display_Grpx

Након креирања ове датотеке позива се генератор ланца који креира све остале фајлове. Генератор између осталог креира и слику на којој се види ланац и на којој се много јасније види сам ток података.



Слика 13 : Ток података за ланац у коме се подаци са камере копирају и приказују на екрану

Након овога још само треба попунити датотеку у којој се налазе функције које се користе за контролу ланца. Следеће функције треба попунити:

Void chains_FrameCopy_SetFrameCopyAlgPrms(
AlgorithmLink_FrameCopyCreateParams *pPrm,
Chains_Ctrl *chainsCfg);

Уколико неки алгоритам може да се извршава на више језгара у овој функцији се на основу генерисаног ланца одлучује који извршиви фајл ће се користити.

Void chains_FrameCopy_SetAppPrms(**chains_FrameCopyObj** *pUcObj, **Void** *appObj);

У оквиру ове функције се подешавају параметри свих веза који се користе у ланцу. Пример нечега што се овде подешава су рецимо параметри за конфигурисање екрана на коме се приказује излаз обраде алгоритма.

Void chains_FrameCopy_StartApp(**Chains_FrameCopyAppObj** *pObj);

Позивом ове функције стартују се све везе и рад целог система почиње.

Void chains_FrameCopy_StopAndDeleteApp(**Chains_CopyAppObj** *pObj);

Позива се када треба да се прекине рад ланца.

Void Chains_FrameCopy(**Chains_Ctrl** *chainsCfg);

У оквиру ове функције позивају се све до сада наведене функције. Она служи за контролу целог ланца. Контрола се врши на основу вредности које се SoC–у пошаљу преко серијске везе по PC-232 стандарду.

4.1.2 Развој алгоритама у оквиру развојног окружења

Као и креирање самих ланаца и развој алгоритама који се користе у оквиру ланаца је такође врло једноставан. Посао инжињера се своди на то да податке добија на улазу,

обради их и на крају обрађене податке упише у излазни низ, док о свему осталом брине развојно окружење.

4.2 Представљање решења за балансирање оптерећења коришћењем PCIe сабирнице

За реализацију балансирања оптерећења било је неопходно направити два алгоритма. Један за слање а други за пријем података. Након тога треба направити ланац у коме се користе ти алгоритми како би могао да се верификује рад самих алгоритама.

4.2.1 Алгоритми за слање и пријем података коришћењем PCIe -а

Реализација слања и пријема реализована је у два алгоритма. То су pcie_send алгоритам и pcie_rcv алгоритам.

У оквиру pcie_send алгоритма, најпре се ради иницијализација PCIe система. У оквиру ње се подешава који подсистем ће се користити, која брзина и све остало неопходно за рад. Након тога се ради тест да се утврди успешна успостава везе са другим SoC–ом. После тога почиње пренос података. Зависно од формата који се шаље дефинисано је прецизно како изгледа ток размене синхронизационих порука и самих података.

Што се тиче pcie_rcv алгоритма код њега је цела процедура доста слична. На почетку се најпре ради иницијализација целог система, подешавање основних параметара комуникације као и тест успоставе везе. Након тога слање података може да почне. На пријемној страни није познато који формат слике да очекује и то мора да се зада као параметар. Након тога комуникација може да почне.

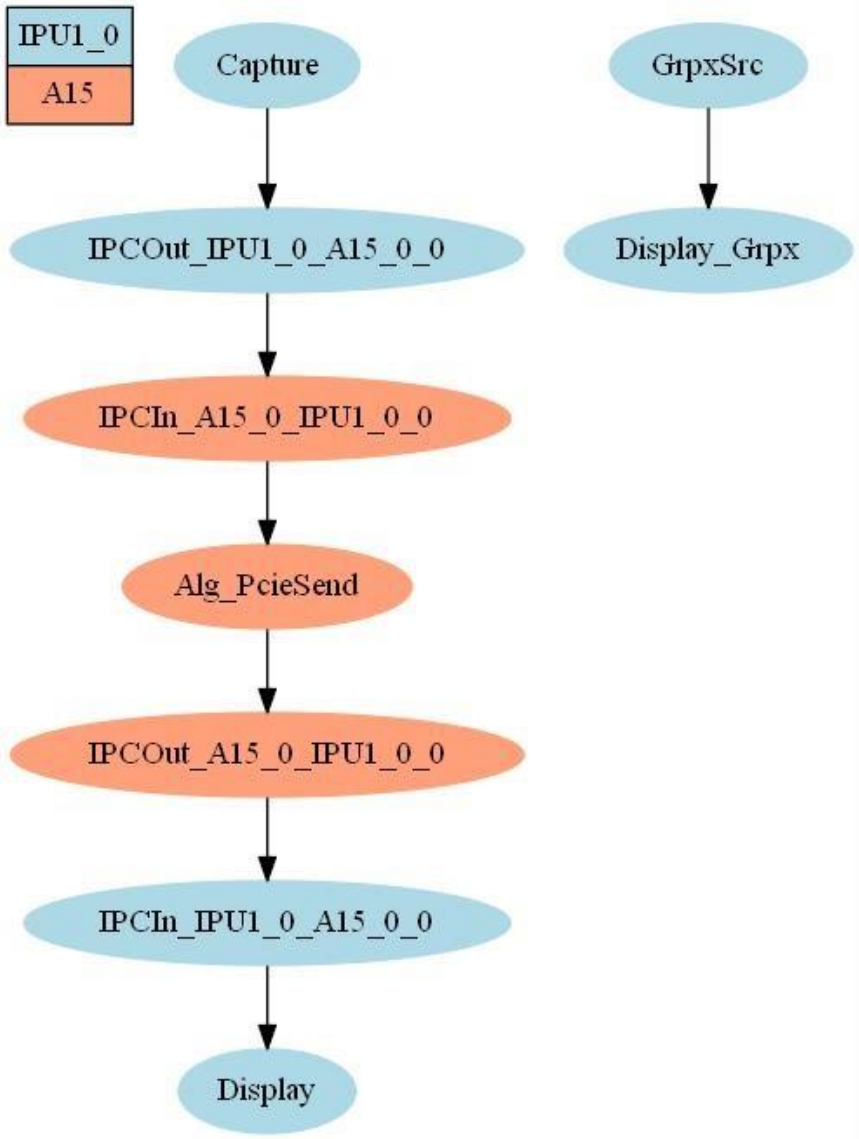
4.2.2 Креирање ланца за тестирање алгоритама за балансирање оптерећења

Да би се написани алгоритми тестирали неопходно је направити ланац у коме се они користе и на тај начин верификовати њихов исправан рад. За верификацију је коришћен врло једноставан пример који на једном SoC–у снима камером, затим преко pcie_send алгоритма снимљене податке шаље на други SoC и у исто време их приказује на екрану како би се упоредили снимци и како би могло да се уочи кашњење уколико постоји.

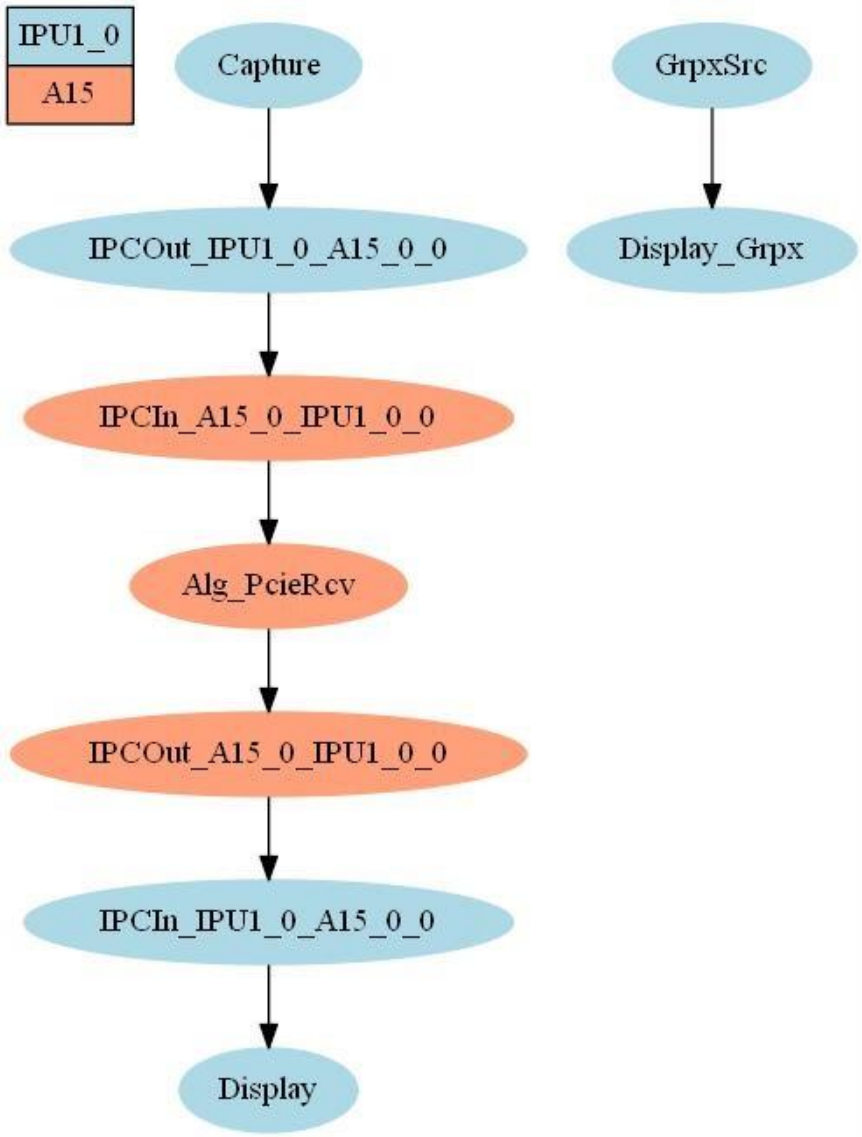
На другом SoC–у ланац изгледа исто као и на првом само се умсто pcie_send алгоритма користи pcie_rcv и након тога се подаци исцртавају на дисплеју. Намеће се логично питање а то је због чега се и на пријемној страни користи ланац који има

камеру када се на екрану приказују подаци пристигли PCIe - ом. Да би се дао одговор на то мора да погледа начин на који раде алгоритми. Функција алгоритма се извршава само када има нове податке на улазу, а то да су нови подаци пристигли алгоритму се јавља преко комуникације између процеса. Пошто PCIe још увек није провучен кроз целу програмску архитектуру и није могуће да се преко унутрашње комуникације процеса дође до информације да су пристигли нови подаци преко PCIe сабирнице. Тако да ће се алгоритам извршавати када му комуникација између процеса јави да је пристигао нови оквир са камере а радиће обраду над подацима пристиглим преко PCIe сабирнице.

На сликама испод могу се погледати схеме ланаца који су коришћени за верификацију рада система за балансирање оптерећења коришћењем PCIe сабирнице.



Слика 14 : Ланац који податке са камере шаље преко PCIe сабирнице и приказује на екрану



Слика 15 : Ланац који податке примљене преко PCIe сабирнице приказује на екрану

4.3 Прилагођавање окружења за развој програмске подршке да би се омогућио рад са PCIe сабирницом

Да би се све оно што је горе описано могло реализовати неопходно је направити пар измена у оквиру самог окружења.

Најпре је потребно исључити опцију коршћења скривене меморије на процесору на коме се извршавају апликације за слање и пријем коричћењем PCIe сабирнице. То се ради због тога што се локације које су учитане у скривену меморију и које су промењене од стране неког екстерног уређаја неће освежити и у скривеној меморији и тиме ће се добијати невалидни подаци.

Затим је потребно преправити адресу на којој је мапиран бафер за пријем података преко PCIe сабирнице јер нису мапиране исто када физичком архитектуром

управља оперативни систем и када не управља, односно када се користи PCIe пример који је дат на нивоу руковалаца.

5. Резултати

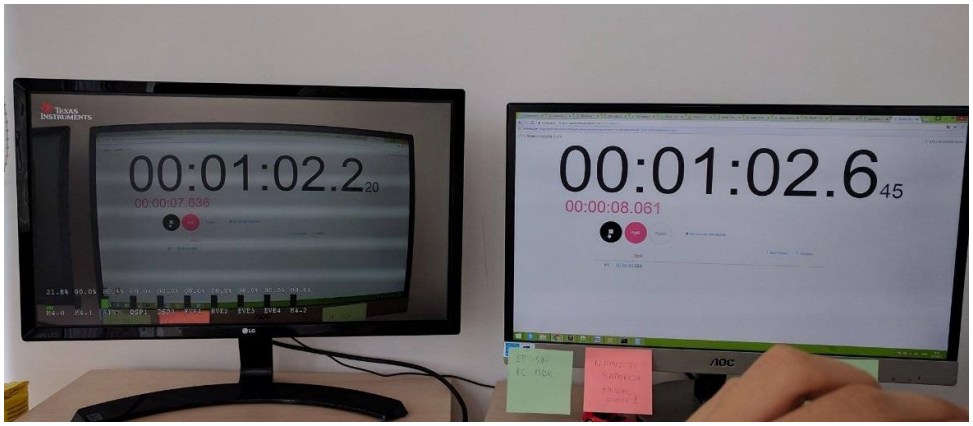
Тестирање и реализација обављени су на већ поменутој Alpha ADAS платформи. Брзина трансфера коју смо успели да постигнемо је знатно мања од максималне теоријске брзине коју омогућава сам протокол. Она износи око 370 МБ/с. Та брзина је и даље сасвим довољна за већину случајева коришћења у ADAS апликацијама на Alpha платформи. Повећање брзине се очекује када буде изашла новија верзија драјвера.

Формат видеа	*Резолуција [пиксел]*	*Оквира у секунди*	*Величина оквира*	*Величина 1 с видеа*	*Могућност трансфера у два правцау реалном времену*
YUV422	1280x720	30	1,757 МБ	52,71 МБ	Да
YUV422	1280x720	60	1,757 МБ	105,42 МБ	Да
YUV422	1920x1080	30	4,078 МБ	122,34 МБ	Да
YUV422	1920x1080	60	4,078 МБ	244,68 МБ	Ne
YUV420	1280x720	30	1,318 МБ	39,54 МБ	Да
YUV420	1280x720	60	1,318 МБ	79,08 МБ	Да
YUV420	1920x1080	30	2.966 МБ	88,98 МБ	Да
YUV420	1920x1080	60	2.966 МБ	177,96 МБ	Да
RGB444	1280x720	30	2,636 МБ	79,08 МБ	Да
RGB444	1280x720	60	2,636 МБ	158,16 МБ	Да
RGB444	1920x1080	30	5,932 МБ	177,96 МБ	Да
RGB444	1920x1080	60	5,932 МБ	355,92 МБ	Не

Табела 2 : Време трансфера различитих резолуција и формата коришћењем PCIe сабирнице на основу брзине утврђене мерењима

Подаци у табели изнад нису утврђени експериментално. Само је брзина конкретно измерена на нивоу руковалаца а до података се дошло рачунски на основу измерене брзине.

Такође је утврђено да постоји кашњење од око 0,4 s између онога што камера тренутно снима и онога што се приказује на екрану након што се пошаље PCIe сабирницом.



Слика 16 : Приказ кашњења приликом слања видеа преко PCIe сабирнице

Даљим испитивањима и тестирањем дошло се до закључка да кашњење не уноси сам трансфер већ функција задужена за копирање пристиглог садржаја ради исписа на екран. То је утврђено тако што се уместо копирања садржаја, вредност показивача на низ који треба да се испише на екран промени тако да сад има исту вредност као и показивач на почетак секције пристигле сабирницом. Када се реализује овако кашњење се спушта на око 0,1 s. У овом случају може се пребацивати само једна компонента слике уколико су компоненте распоређене у различите низове јер би у супротом дошло до преписвања.

Слика 17 : Приказ кашњења приликом слања само Y компоненте коришћењем PCIe сабирнице

Још једна вредност које је битна да се узме у обзир јесте заузетост процесора када се ради балансирање. Јер уколико само балансирање оптерећује процесор више него што би га оптерећивало извршавање целог алгоритма на једном SoC-у, цео концепт губи смисао.

У реализацији која је овде представљена већину посла који је везан за механизам балансирања обавља општенаменски процесор Cortex-A15 [6]. Ради се о општенаменском процесеру високих перформанси ARMv7-A [7] архитектуре.

Заузетост процесора приказана је у табели испод.

	Формат оквира	Резолуција оквира [пиксел]	Заузетост пријемне стране	Заузетост предајне стране
Слање само Y компоненте	YUV 420 SP	1280 x 720	53,3 %	49,8 %
Слање свих компоненти	YUV 420 SP	1280 x 720	80,4 %	51,6 %

Табела 3 : Заузетост процесора приликом балансирања оптерећења

Овакве бројке могу да заварају и да се помисли да је само балансирање комплексна операција али утврђено је да је као и за кашњење тако и за заузетост процесора одговорна веома скупа операција копирања меморије. Када се копирање премести са процесора и када га буде радила јединица директног приступа меморији очекује се да ће оптерећење бити значајно мање.

6. Закључак

У оквиру рада представљено је решење за превазилажење проблема мањка перформанси једног SoC-а пребацивањем дела пбраде на други SoC. Потреба за системима са више процесорске моћи је све већа, а ово је једно универзално решење које се може применити готово свуда. Балансирање оптерећења ће као концепт бити све присутнији међу уграђеним уређајима јер ни једна физичка архитектура не може да прати брзину развоја програмске подршке која на њој мора да се извршава.

Представљен је концепт и имплементација система за балансирање оптерећења између два SoC-а коришћењем PCIe сабирнице. Описан је начин комуникације, трансфера података као и редослед операција у оквиру комуникације.

Што се тиче програмског решења најкомплекснији део имплементације био је прилагођавање развојног окружења тако да се омогући коришћење PCIe сабирнице на највишем нивоу у окружењу, нивоу везе пошто га развојно окружење до сад није подржавало.

Када је реч о самом баланисирању, ово је тек почетак. У блиској будућности решење ће бити проширено и побољшано. Проблем кашњења приликом копирања ће се решити коришћењем директног приступа меморији (енг. DMA, **D**irect **M**emory **A**ccess) за копирање података уместо процесора који то ради у тренутној реализацији. Следеће што треба урадити јесте провлачење PCIe – а кроз програмски стек да би се алгоритам позивао само када стигну нови подаци чиме би се растеретило заузеће процесора.

7. Литература

[1] Advanced driver assistance systems, http://www.qnx.com/content/qnx/en/products/adas/index.html, учитано 01.07.2017

[2] TI Vision SDK, Optimized Vision Libraries for ADAS Systems, http://www.ti.com/lit/wp/spry260/spry260.pdf, учитано 01.07.2017

[3] PCIe, TDA2x S for Advanced Driver Assistance Systems (ADAS) silicon revision 2.0, 1.x (SPRUHK5AD), поглавље 24.8 PCIe Controller, страна 6712 - 6913

[4] Automotive machine vision alpha reference board on Texas Instrumensts SoCs http://www.rt-rk.com/news/195-rt-rk-presents-automotive-machine-vision-alphareference-board-on-texas-instruments-socs, учитано 01.07.2017

[5] TDA2x ADAS System-on-Chip, http://www.ti.com/lit/ml/sprt681/sprt681.pdf , учитано 01.07.2017

[6] Cortex-A15, https://developer.arm.com/products/processors/cortex-a/cortex-a15, учитано 01.07.2017

[7] ARM Processor Architecture, https://www.arm.com/products/processors/instruction-set-architectures/index.php, учитано 01.07.2017