



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: Наташа Перковић  
Број индекса: РА 137/2013

Тема рада: Развој ADAS апликација које се наизмјенично/конкуренто извршавају на TDA2x платформи у програмском окружењу *Vision SDK*

Ментор рада: доц. др Небојша Пјевалица

Нови Сад, јул, 2017.



## 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:	Наташа Перковић		
Ментор, МН:	доц. др Небојша Пјевалица		
Наслов рада, НР:	Развој ADAS апликација које се наизменично/конкуренто извршавају на TDA2x платформи у програмском окружењу Vision SDK		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2017		
Издавач, ИЗ:	Ауторски репрингт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страница/цитата/табела/слика/графика/прилога)	7/27/6/2/16/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника и рачунарске комуникације		
Предметна одредница/Кључне речи, ПО:	Системи за помоћ возачу, Камера, Алфа,		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У оквиру овог рада представљено је решење за наизменично извршавање апликација за помоћ возачу. За реализацију решења користили смо ALPHA AMV платформу и <i>Vision Software Development Kit</i> окружење. Концепт решења јесте да се више апликација извршавају на истом чипу и да се омогући брзо и једноставно покретање/заустављање тренутно активне апликације.		
Датум прихватања теме, ДП:	30.06.2017.		
Датум одбране, ДО:	20.07.2017.		
Чланови комисије, КО:	Председник:	Доц. др Бјелица Милан	
	Члан:	Доц. др Каштелан Иван	Потпис ментора
	Члан, ментор:	доц. др Пјевалица Небојша	



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO:</b>			
Identification number, <b>INO:</b>			
Document type, <b>DT:</b>	Monographic publication		
Type of record, <b>TR:</b>	Textual printed material		
Contents code, <b>CC:</b>	Bachelor Thesis		
Author, <b>AU:</b>	<b>Nataša Perković</b>		
Mentor, <b>MN:</b>	<b>Nebojša Pjevalica, PhD</b>		
Title, <b>TI:</b>	<b>Development of ADAS applications for alternative/concurrent execution on TDA2x platform in development environment Vision SDK</b>		
Language of text, <b>LT:</b>	Serbian		
Language of abstract, <b>LA:</b>	Serbian		
Country of publication, <b>CP:</b>	Republic of Serbia		
Locality of publication, <b>LP:</b>	Vojvodina		
Publication year, <b>PY:</b>	2017		
Publisher, <b>PB:</b>	Author's reprint		
Publication place, <b>PP:</b>	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, <b>PD:</b> <small>(chapters/pages/ref./tables/pictures/graphs/appendices)</small>	7/27/6/2/16/0/0		
Scientific field, <b>SF:</b>	Electrical Engineering		
Scientific discipline, <b>SD:</b>	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, <b>S/KW:</b>	Advanced Driver Assistance System, Camera, Alpha		
<b>UC</b>			
Holding data, <b>HD:</b>	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, <b>N:</b>			
Abstract, <b>AB:</b>	<p>A solution for real-time switching of driver assistance applications is presented in this paper. The solution is implemented on ALPHA AMV platform by using <b>Vision Software Development Kit</b>. Solution concept enables execution of several applications on the same SoC and provides possibility for quick and easy starting/stopping of currently active application.</p>		
Accepted by the Scientific Board on, <b>ASB:</b>			
Defended on, <b>DE:</b>			
Defended Board, <b>DB:</b>	President:	Bjelica Milan, PhD	
	Member:	Kaštelan Ivan, PhD	Mentor's sign
	Member, Mentor:	Pjevalica Nebojša, PhD	

## **Zahvalnost**

Zahvaljujem se dr Željku Lukaču i dr Mileni Milošević na stručnoj pomoći i savetima prilikom izrade ovog rada.

Posebno se zahvaljujem svojoj porodici na pruženoj podršci tokom cijelog školovanja.

Na kraju se zahvaljujem institutu RT-RK, kao i svima ostalima koji su omogućili i na bilo koji način doprinjeli realizaciji ovog rada.



# УНИВЕРЗИТЕТ У НОВОМ САДУ

## ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



### SADRŽAJ

1.	Uvod .....	1
2.	Teorijske osnove.....	3
2.1	Napredni sistemi za pomoć vozaču – ADAS.....	3
2.2	Alpha platforma .....	4
2.2.1	TDA2x SoC .....	6
2.3	Razvojno okruženje VSDK.....	7
2.4	Generisanje i učitavanje algoritma.....	8
3.	Koncept rješenja .....	9
3.1	Algoritmi koji su korišteni za realizaciju rada .....	9
3.1.1	Camera mirror system - CMS.....	9
3.1.2	Surround view – SV .....	12
3.2	Ideja za naizmjenično izvršavanje .....	14
4.	Programsko rješenje .....	15
4.1	Proces spajanja CMS i SV aplikacija.....	15
4.2	Ulazni podsistem.....	17
4.3	Izlazni podsistem .....	18
4.4	Gate blok .....	20
5.	Rezultati.....	23
6.	Zaključak .....	26
7.	Literatura .....	27

## SPISAK SLIKA

Slika 1 : Prikaz senzora za podatke o spoljašnjosti vozila.....	4
Slika 2: Alpha ADAS platforma .....	5
Slika 3: Šematski prikaz ALPHA ploče .....	6
Slika 4: Šematski prikaz Texas Instruments TDA2x SoC-a.....	7
Slika 5: TI Vision SDK slojevita arhitektura softvera .....	8
Slika 6: Dijagram toka CMS-a.....	10
Slika 7: Aplikacija CMS - za nadzor mrtvog ugla.....	11
Slika 8: Dijagram toka SV-a.....	12
Slika 9: Aplikacija SV - pregled okruženja vozila .....	13
Slika 10: Dijagram toka CMS i SV zajedno .....	16
Slika 11: Ulazni podsistem .....	17
Slika 12: Cjelokupan prikaz displej podsistema .....	19
Slika 13: Blok dijagram grafičkog podsistema .....	20
Slika 14: Prikaz Gate_CMS bloka kao prekidača.....	21
Slika 15: Prikaz Gate_SV bloka kao prekidača .....	22
Slika 16: Vrijeme prebacivanja na drugi algoritam .....	25

## **SPISAK TABELA**

Tabela 1: Zauzetost procesorskih jezgara Surround view .....	23
Tabela 2: Zauzetost procesorskih jezgara Camera mirror system .....	24

## Skraćenice

<b>ADAS</b>	- Advanced <b>D</b> river <b>A</b> sistance <b>S</b> ystems – Napredni sistemi za pomoć vozaču
<b>AMV</b>	– <b>A</b> utomotive <b>M</b> achine <b>V</b> ision - Sistemi za obradu video signala za potrebe auto industrije
<b>CMS</b>	- <b>C</b> amera <b>M</b> irror <b>S</b> ystem – Sistem za zamjenu ogledala
<b>DSP</b>	- <b>D</b> igital <b>S</b> ignal <b>P</b> rocessor – Digitalni signal processor
<b>EVE</b>	- <b>E</b> mbedded <b>V</b> ideo <b>E</b> ngine – Jedinica za obradu video sadržaja na niskom i srednjem nivou
<b>HDMI</b>	- <b>H</b> igh <b>D</b> efinition <b>M</b> ultimedia <b>I</b> nterface – Multimedijalna sprega visoke rezolucije
<b>IPC</b>	- <b>I</b> nter <b>P</b> rocess <b>C</b> ommunication – Komunikacija među procesorima
<b>SBL</b>	- <b>T</b> he <b>S</b> econdary <b>B</b> ootloader – Inicijalni punjač drugog stepena
<b>SDK</b>	- <b>S</b> oftware <b>D</b> evelopment <b>K</b> it – Okruženje za razvoj programske podrške
<b>SoC</b>	- <b>S</b> ystem <b>o</b> n <b>C</b> hip – Računarski sistem u okviru integrisanog kola
<b>SV</b>	- <b>S</b> urround <b>V</b> iew – Sistem za prikaz potpunog okruženja vozila
<b>VIN</b>	- <b>V</b> ideo <b>I</b> nput – Video ulaz
<b>VSDK</b>	- <b>V</b> ision <b>S</b> oftware <b>D</b> evelopment <b>K</b> it – Okruženje za razvoj programa
<b>ZORDER</b>	– <b>Z</b> -axis <b>O</b> rder – Redosled iscrtavanja slojeva na ekranu po Z-osi

## 1. Uvod

Poslednjih decenija možemo da vidimo ubrzan razvoj automobila. Od 1970. (140 modela), preko 1998. (260 modela), pa sve do 2012. (684 modela), broj modela automobila raste eksponencijalno. Samim povećanjem broja automobila, povećava se i svijest za sigurnom vožnjom. Sa svakom promjenom vozač je sve manje uključen u vožnju. Razvojem automobilske industrije, sa stanovišta tehnologije, moderna vozila na izvjestan način počinju sve više i vjerodostojnije spoznavati vlastito okruženje. Upravo ta sposobnost spoznaje sopstvenog okruženja, daje im određen nivo autonomije u odnosu na vozača kao i mogućnost samostalnog učestvovanja u saobraćaju. U cilju dostizanja potpune automotizacije automobila, nadajući se da će jednog dana i sam vozač da bude putnik, imamo 5 nivoa autonomije.

Nivo 1 - Automatizacija specifičnih, upravljačkih funkcija, kao sto su kontrola brzine, prepoznavanje saobraćajnih traka i automatizovano paralelno parkiranje. Vozači su u potpunosti angažovani i odgovorni za cijelokupnu kontrolu vozila (u svakom trenutku ruke na volanu i stopalo na pedali).

Nivo 2 - Automatizacija višestrukih i integrisanih upravljačkih funkcija, kao što je kontrola brzine sa prepoznavanjem saobraćajnih traka. Vozači su odgovorni za nadgledanje saobraćaja i očekuje se da će biti dostupni za kontrolu u svakom trenutku (istovremeno „puštenih“ ruku i bez stopala na pedali)

Nivo 3 – Vozači mogu pod određenim uslovima odstupiti od svih sigurnosno značajnih funkcija i oslanjati se da vozilo nadgleda okolinu oko sebe (eng. *Limited Self-Driving Automation*)

Nivo 4 – Automobili mogu vršiti sve funkcije vozača pod određenim uslovima (eng. *Self-Driving Under Specified Conditions*)

Nivo 5 – Automobili mogu vršiti sve funkcije vozača u svim mogućim uslovima (eng. *Self-Driving Automation*)

Do postizanja ovakvog cilja, nailazimo na mnoge probleme. Na platformi, sa jedne strane, nema dovoljno resursa da se sve aplikacije izvršavaju u paraleli, a sa druge strane sve aplikacije nam nisu potrebne u svim situacijama. Tako dolazimo do razmatranja ideje da se više različitih aplikacija naizmjenično izvšavaju, odnosno da se aplikacije učitavaju na početku i pokreću se i zaustavljaju bez ponovnog učitavanja iz stalne memorije. U ovom radu će biti priče o ovom problemu i njegovoj realizaciji. Rješenje je realizovano u sklopu Texas Instruments-ovog razvojnog okruženja VisionSDK, (eng. VSDK, *Vision Software Development Kit*) [1], verzije 02.12.01.00, u programskom jeziku C i testirano na AMV Alpha [2] (eng. AMV, *Automotive Machine Vision*) platformi, koju je izradio RT-RK institut.

Rad je organizovan u šest poglavlja.

- U drugom poglavlju navedeni su osnove iz oblasti, poput teorijskih osnova i opisa ADAS aplikacija, Alpha platforma, kao i razni senzori.
- Treće poglavlje opisuje koncept rješenja, od problema do njegovog rješenja.
- U četvrtom poglavlju dat je opis programske realizacije.
- U petom poglavlju se nalaze rezultati ovog rada.
- Šesto poglavlje sadrži kratak opis šta je urađeno u ovom radu i koji su pravci daljeg razvoja.

## 2. Teorijske osnove

U ovom poglavlju su opisane teorijske osnove na kojima je rad zasnovan. Opisano je stanje u automobilskoj industriji i dat je prikaz novih tehnologija kao što su napredni sistemi za pomoć vozaču, prikaz Alpha platforme i njenih performansi.

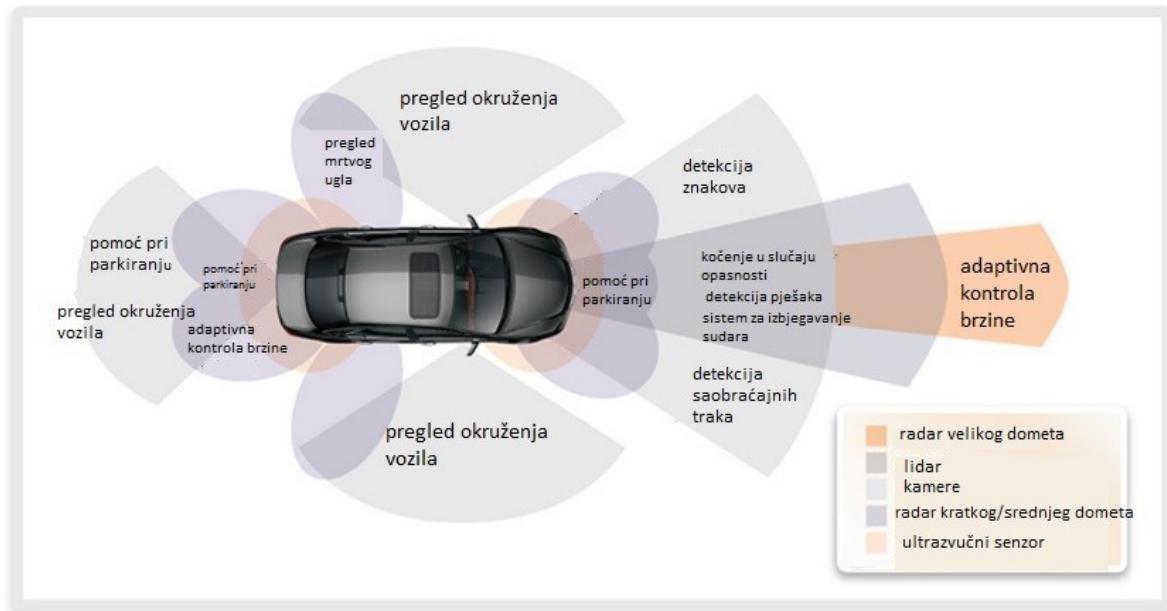
### 2.1 Napredni sistemi za pomoć vozaču – ADAS

Napredni sistemi za pomoć vozaču, ADAS [3], su sistemi razvijeni da poboljšaju sigurnost vozača, kao i sigurnost svih učesnika u saobraćaju. Sigurnosni dodaci su osmišljeni tako da izbjegnu saobraćajne nesreće, nudeći tehnologije koje upozoravaju vozača na potencijalne probleme ili preuzimaju kontrolu nad vozilom.

Primeri sistema za pomoć vozaču [4] :

- navigacioni sistem u automobilu za pružanje informacija o saobraćaju u realnom vremenu – TMC (eng. *Traffic Message Channel*);
- sistem za ublažavanje posledica sudara - PCS (eng. *Pre-Crash Safety*);
- sistem za detekciju pešaka (eng. *Pedestrian Safety Systems*);
- prilagodljivi tempomat – ACC (eng. *Adaptive Cruise Control*);
- sistem za upozorenje pri napuštanju trake (eng. *Lane Departure*);
- prepoznavanje objekata, npr. saobraćajnih znakova (eng. *Object detection*);
- detekcija pospanosti vozača (eng. *Drowsiness Detection System*);
- sistem za pomoć pri smanjenoj vidljivosti (npr. *Night Vision*);
- sistem za prikaz potpunog okruženja vozila (eng. *Surround View*);
- pomoć pri parkiranju (eng. *Park Assist*);
- pregled mrtvog ugla (eng. *Camera Mirror System*).

ADAS platforme su dostupne u više formi. Neka svojstva su ugrađena u automobile, a neka su dostupna kao dodatni paket. ADAS platforme se oslanjaju na više ulaza (uključujući ulaze sa kamera, LiDARe (laserski daljinomer), radare i sl.), obradu slike i na komunikaciju više sistema u vozilu.



Slika 1 : Prikaz senzora za podatke o spoljašnjosti vozila

## 2.2 Alpha platforma

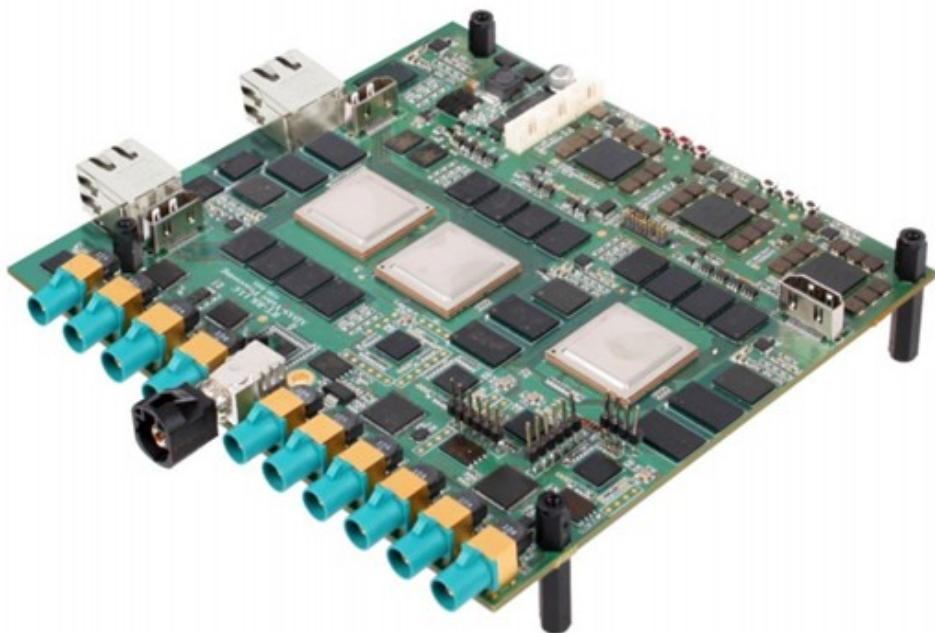
Danas postoji potreba za mnoštvom ADAS aplikacija kako bi učinili vožnju za vozača opušteniju i što je još bitnije sigurniju. Automobilska industrijia teži da postigne jedan cilj, a to je autopilot.

Da bi djelovali u svijetu kakav poznajemo, moramo biti u mogućnosti da sagledamo sve događaje i predmete koji se nalaze oko vozila. Kako bi uočili objekte u svom okruženju, automobili koriste razne senzore (ultrazvučne, radar, lidar, kamere, itd.). Nakon dobijanja informacija sa svih senzora, potrebno ih je obraditi i dalje proslijediti instrukcije vozilu, kako bi znalo kako u kom trenutku da se ponaša. Sve podatke treba obraditi u realnom vremenu. Iz tih razloga potrebna je moćna platforma sa više čipova (eng. SoC, *System on Chip*) koji će da obrađuju veliku količinu podataka.

Algoritmi kao što su pregled okruženja vozila (eng. *Surround view*), detekcija saobraćajnih traka (eng. *Line detection*), nadzor vozača (eng. *Driver monitoring system*), pregled mrtvog ugla (eng. *Camera Mirror System*), detekcija objekata (eng. *Object detection*), sistem za pomoć pri smanjenoj vidljivosti (eng. *Night vision*), i prepoznavanje pješaka (eng. *Pedestrian*

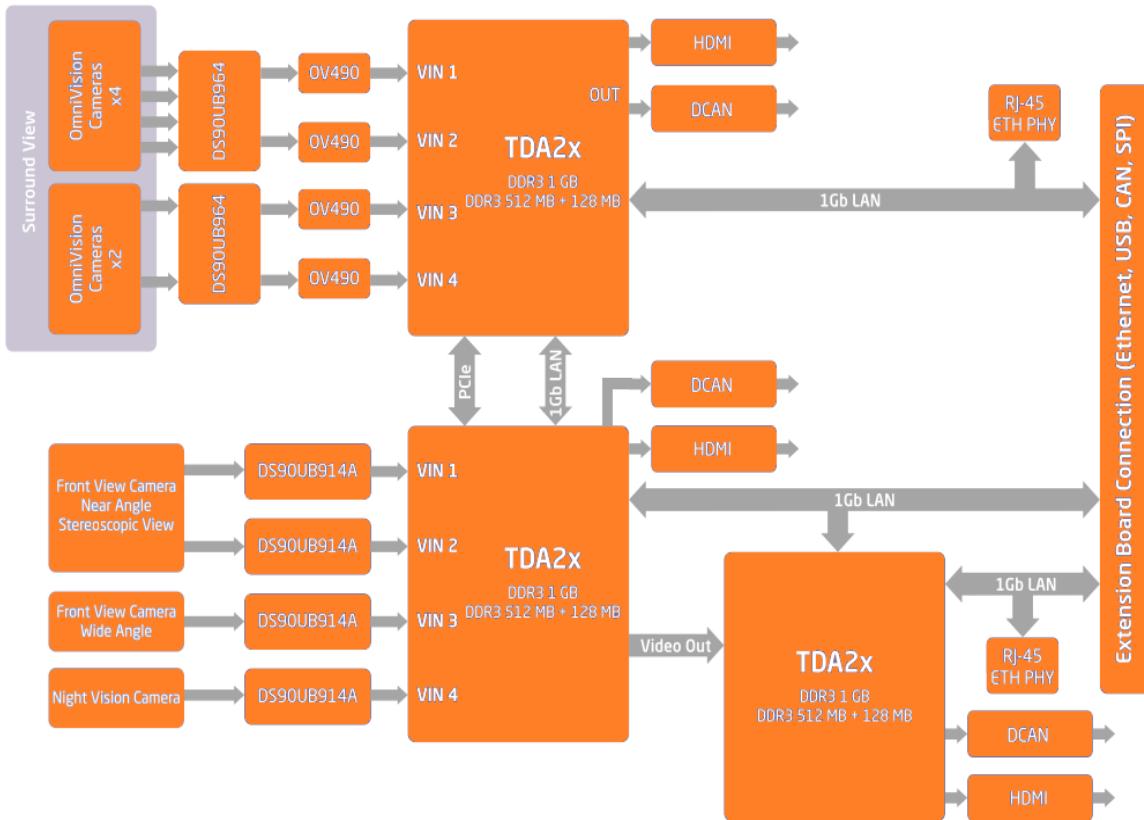
*Detection*) su veoma kompleksni i svakom od njih su potrebne odgovarajuće hardverske komponente, na kojima će se izvršavati.

Za realizaciju ovog rada, kao što je već rečeno, korišćena je ALPHA razvojna platforma. Alpha platforma je napravljena sa svim neohodnim hardverkim komponentama za potrebe ADAS aplikacija da bi se omogućio njihov efikasan rad. Na Slici 3. je prikazana Alpha ADAS platforma.



Slika 2: Alpha ADAS platforma

Platforma se sastoji iz tri TDA2xx [5] SoC-a proizvođača Texas Instruments-a. Prvi SoC se naziva SCV (eng. *Surround Camera View*). Povezan je na šest kanala video ulaza (eng. VIN, *Video INput*) i kao što mu i samo ime kaže koristi se za prikaz okoline vozila. Drugi SoC se naziva FFN (eng. *Front View Camera Near Angle Stereoscopic View*, *Front View Camera Wide Angle*, *Night Vision Camera*). On podržava četiri video ulazna kanala i namjenjen je za snimanje puta ispred vozila. Treći SoC se naziva FUS (eng. *FUSion*). On se koristi za sintezu informacija dobijenih sa senzora priključenih na prva dva SoC-a. U slučaju kada su podaci dobijeni obradom na prva dva SoC-a u protivrečnosti, algoritmi koji se izvršavaju na trećem SoC-u zaduženi su da odluče koje podatke će smatrati tačnim. Ova tri SoC-a su međusobno povezani raznim magistralama. Na Slici 4. može se vidjeti šematski prikaz platforme.

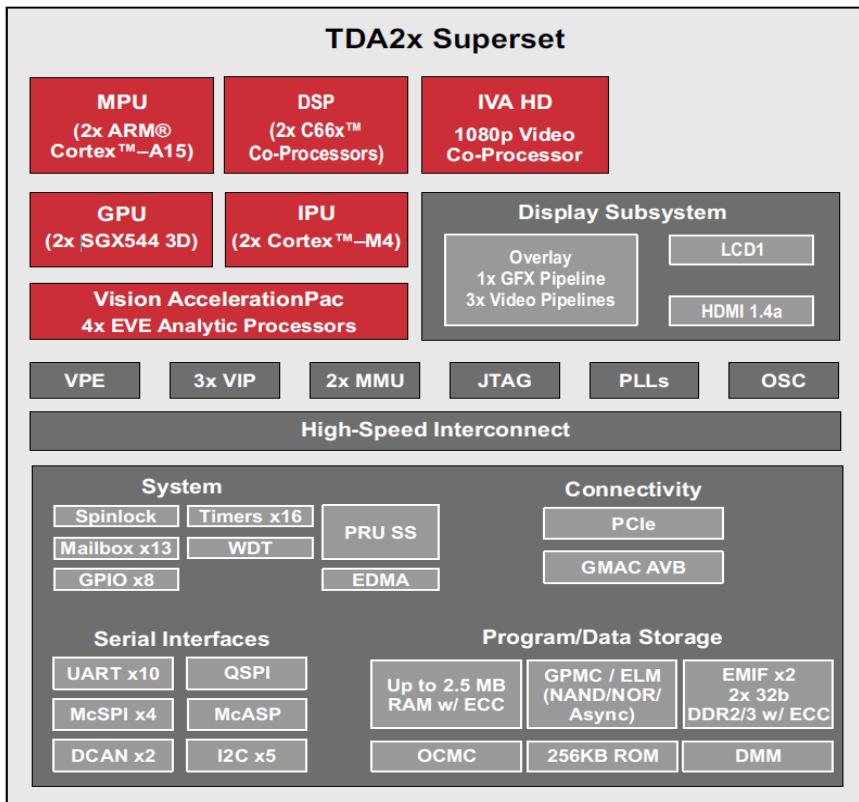


Slika 3: Šematski prikaz ALPHA ploče

### 2.2.1 TDA2x SoC

Sa šematskog prikaza sa Slike 4. možemo da zaključimo da je riječ o multiprocesorskom sistemu. Prisutna su četiri dvojezgarna procesora opšte namjene, od kojih su dva arhitekture ARM Cortex M4, sa oznakama IPU 1 i IPU 2 i dva arhitektura ARM Cortex A15, sa oznakom MPU. Prisutna su i dva procesora namjenjena digitalnoj obradi signala, arhitekture Texas Instruments C66x, sa oznakom DSP. Na SoC-u postoje i četiri jedinice za obradu video sadržaja niskog i srednjeg nivoa (*Embedded Vision Engine* – EVE). Pristutna je i VPE (*Video Processing Engine*) komponenta, čija je funkcija obrada video sadržaja.

Za komunikaciju sa računarcem opšte namjene koristi se UART (*Universal Asynchronous Receiver/Transmitter*) sprega, kao i Ethernet sprega. I<sup>2</sup>C (*Inter-Integrated Circuit*) sprega se koristi za komunikaciju sa jedinicama za deserijalizaciju podataka dopremljenih sa kamera. Aplikacija i početni punjač se učitavaju sa Micro SD kartice. Preko HDMI (*High Definition Media Interface*) sprega, dobija se izlazna slika na monitoru. Radi formiranja izlazne slike koristi se displej podsistem (eng. *Display Subsystem*).

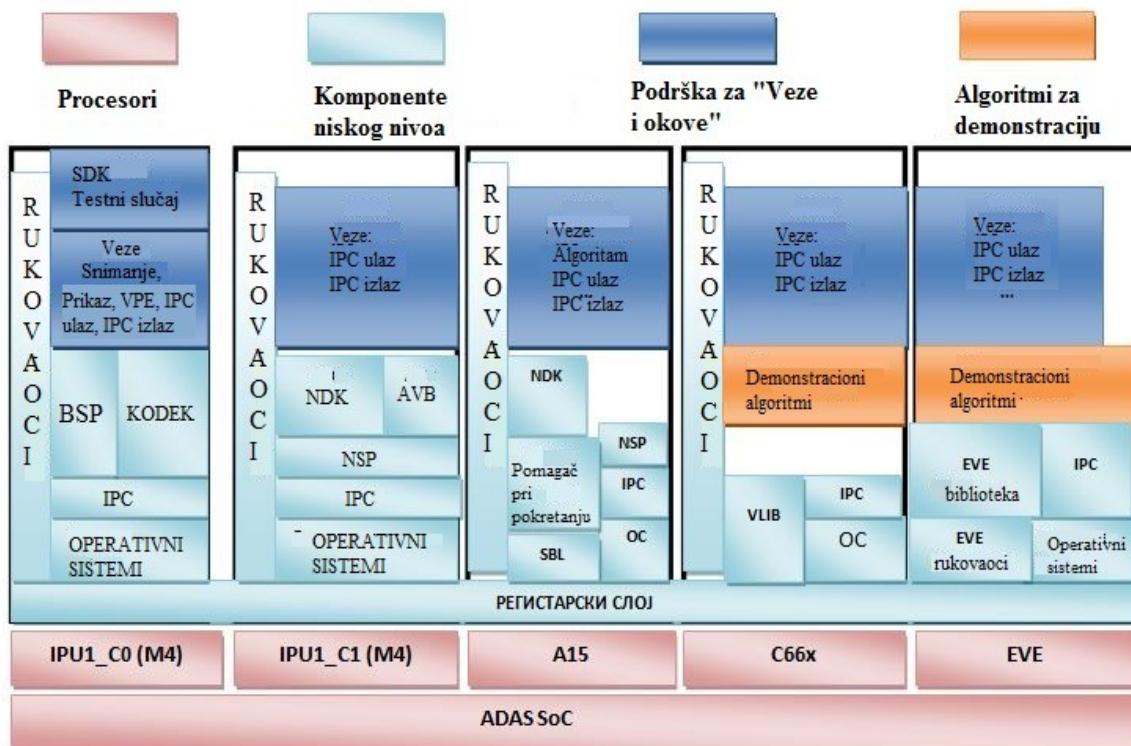


Slika 4: Šematski prikaz Texas Instruments TDA2x SoC-a

## 2.3 Razvojno okruženje VSDK

Razvojno okruženje koje je korišteno za razvoj aplikacija jeste okruženje za razvoj programske podrške za TDA2xx SoC (eng. VSDK, *Vision Software Development Kit*). VSDK je multiprocesorska platforma za razvoj softvera za ADAS. VSDK sadrži DSP I EVE biblioteke sa osnovnim softverskim drajverima i algoritmima za analizu i obradu signala, kao i pojedine primjere aplikacija. Kao takav predstavlja pogodno razvojno okruženje. EVE i DSP biblioteke nude optimizovane funkcionalnosti na niskom i srednjem nivou obrade slike u razvijanju i implementaciji algoritama na aplikativnom sloju. Softverski radni okvir dozvoljava korisniku da razvija kreiranje različitih tokova podataka za ADAS aplikacije što uključuje prihvatanje video signala, preprocesiranje video signala, algoritam za analizu i video prikaz.

VSDK je baziran na radnom okviru “Veze i lanci” (eng. *Links and Chains*). VSDK instalacioni paket uključuje sve alate i komponente potrebne za pokretanje aplikacija, a to podrazumijeva alate za generisanje koda, BIOS (eng. *Binary Input Output System*), IPC (engl. *Inter Processor Communication*), rukovoce (eng. *starterware*), BSP drajvere (eng. *Board Support Package*), mrežne stekove, kodeke kao i kernele algoritma. Na Slici 5. je dat detaljan opis VSDK slojeva.



Slika 5: TI Vision SDK slojevita arhitektura softvera

## 2.4 Generisanje i učitavanje algoritma

Da bi se algoritam pustio u rad, potrebno je odraditi par koraka. Prvi od njih jeste generisanje početnog punjača (eng. *bootloader*). Bootloader je računarski program koji učitava operativni sistem ili neki drugi softver za računar. U procesu pokretanja učitava se i pokreće softver. Bootloader se učitava u glavnu memoriju iz interne memorije. Zatim, početni bootloader učitava i izvršava procese koji dovršavaju pokretanje. Kod pokretanja dolazi do predefinisanosti lokacije. Ako je ta lokacija previše ograničena, iz nekog razloga, taj primarni bootloader poziva inicijalni bootloader drugog stepena (eng. *Secondary Bootloader*). U našem slučaju bootloader drugog stepena se naziva MLO.

Sledeći korak jeste generisanje AppImage-a. AppImage jeste binarni fajl u kome se nalaze izgenerisani svi algoritmi, spremni za rad.

Ova dva fajla zajedno idu na SD karticu i na taj način se pokreću na Alpha ploči.

## 3. Koncept rješenja

U ovom radu je predstavljeno naizmjenično izvršavanje aplikacija. U toku vožnje javlja se potreba za naizmjeničnim izvršavanjem aplikacija, što zbog nedostatka resursa da se sve odjednom izvršava, što zbog toga što nije neophodno da sve aplikacije budu istovremeno uključene. Za realizaciju ovog rada koristili smo algoritam koji zamjenjuje sistem retrovizora na vozilu (eng. CMS, *Camera Mirror System*) i algoritam koji daje sliku okruženja oko vozila (eng. SV, *Surround View*), kao primjere algoritama koji imaju potrebu da se naizmjenično izvršavaju u realnom okruženju.

Vozač u toku vožnje najviše koristi CMS aplikaciju, međutim, ako u nekom trenutku poželi iz nekog razloga da se parkira, treba da bude u mogućnosti da se aplikacija, SV, koja će mu pomoći pri parkiranju, pokrene za najkraće moguće vrijeme.

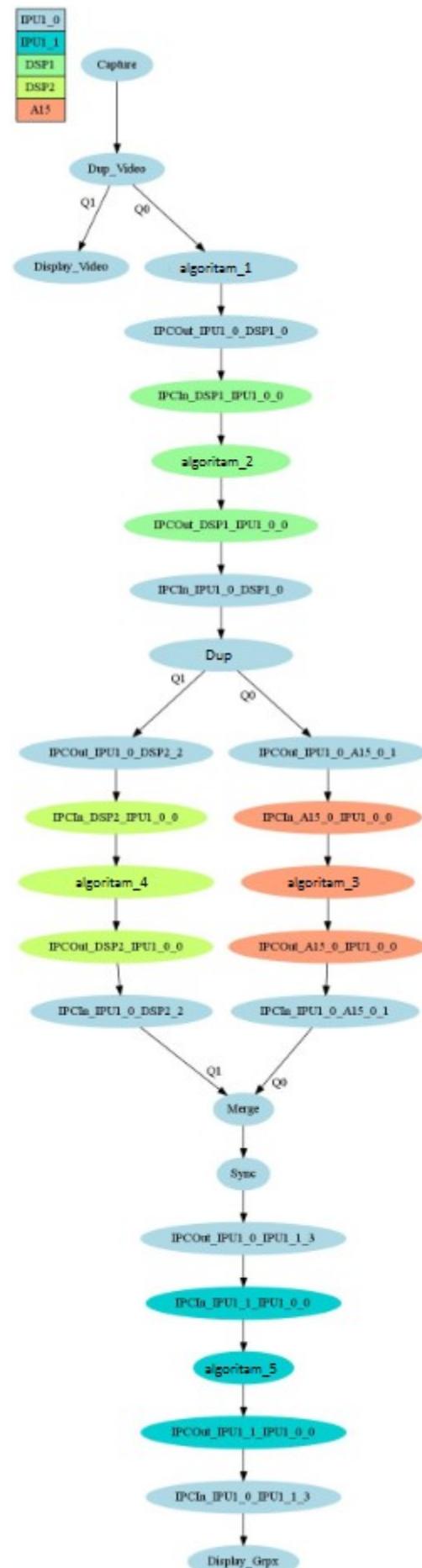
U ovom poglavlju biće opisana te dvije aplikacije, kao i ideja za njihovo naizmjenično izvršavanje.

### 3.1 Algoritmi koji su korišteni za realizaciju rada

#### 3.1.1 Camera mirror system - CMS

Prvi korišten algoritam je *Camera mirror system*. Kao ulaz koriste se dvije kamere, koje se nalaze na mjestu retrovizora. Podaci dobijeni sa kamera obrađuju se dalje kroz algoritam, na DSP, A15, IPU procesorima. Takvi obrađeni podaci se šalju na ekran. Sa ovom aplikacijom imamo bolji pregled automobila koji prilaze sa strane i detekciju istih.

Na Slici 7. prikazan je dijagram toka podataka sistema za CMS aplikaciju, od preuzimanja podataka sa kamera do formiranja izlazne slike.



Slika 6: Dijagram toka CMS-a

Svaki blok u dijagramu toka podataka predstavlja nit (eng. *thread*), koja se izvršava na nekom od raspoloživih procesorskih jezgara unutar SoC-a. U gornjem lijevom ugлу predstavljeni su procesori koji se koriste u tom algoritmu. Strelice predstavljaju proslijđivanje podataka između niti, koje se vrši pomoću blokova podataka, bafera.

*Capture* blok je zadužen za prikupljanje sadržaja sa kamere. Zatim se podaci šalju do *Dup* bloka. On duplira bafera i svaka grana je zadužena za posebnu obradu istog sadržaja. Sa lijeve strane se nalazi *Display\_Video* blok. Ova nit prosleđuje bafera prema displej podsistemu. Sa desne strane se nalazi algoritam koji prati kretanje automobila koji dolaze sa strane, iza vozila, detektuje ga i na ekranu, kao rezultat, iscrtava pravougaonik oko njega. Takođe, sa desne strane na ekranu se ispisuje rastojanje do automobila koje nam prilazi, brzina njegovog kretanja, kao i vrijeme do potencijalnog sudara. Kao što možemo da vidimo sa slike, svi blokovi se ne izvršavaju na jednom procesoru. Međuprocesorska komunikacija (eng. *IPC – Inter Processor Communication*) se koristi za saradnju različitih zadataka koji se pokreću na različitim procesorima na SoC-u. *IPCIn* je blok namijenjen dostavljanju bafera podataka sa različitog procesorskog jezgra, a *IPCOut* je blok namijenjen prosleđivanju bafera drugom procesorskom jezgru. Na kraju dijagrama se nalazi *Display\_GrpX* blok, koji sav ovaj sadržaj iscrtava i prikazuje na ekranu.

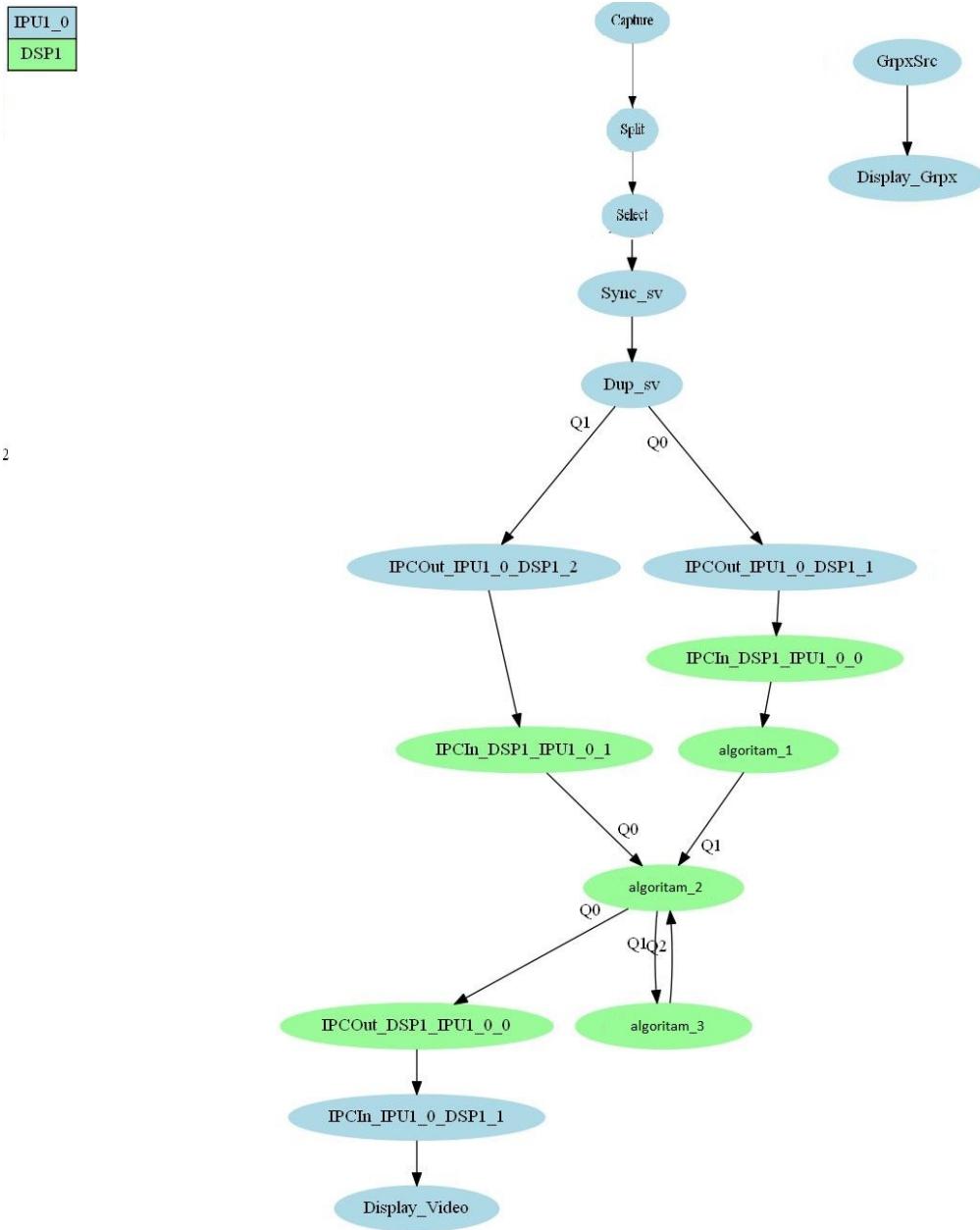


Slika 7: Aplikacija CMS - za nadzor mrvog ugla

### 3.1.2 Surround view – SV

Drugi algoritam koji je korišten jeste Surround view. Ovaj algoritam, takođe, kao ulaz koristi kamere. Za ovaj algoritam potrebne su četiri kamere. Kamere su raspoređene na četiri strane vozila (jedna naprijed, jedna nazad i po jedna na svakoj strani vozila). Slike dobijene sa kamera se obrađuju, takođe na DSP i IPU procesorima, i na ekranu se dobije prikaz od 360 stepeni oko automobila. Ova aplikacija pomaže sagledavanje objekata koji se nalaze oko automobila. Takođe, služi kao pomoć pri parkiranju.

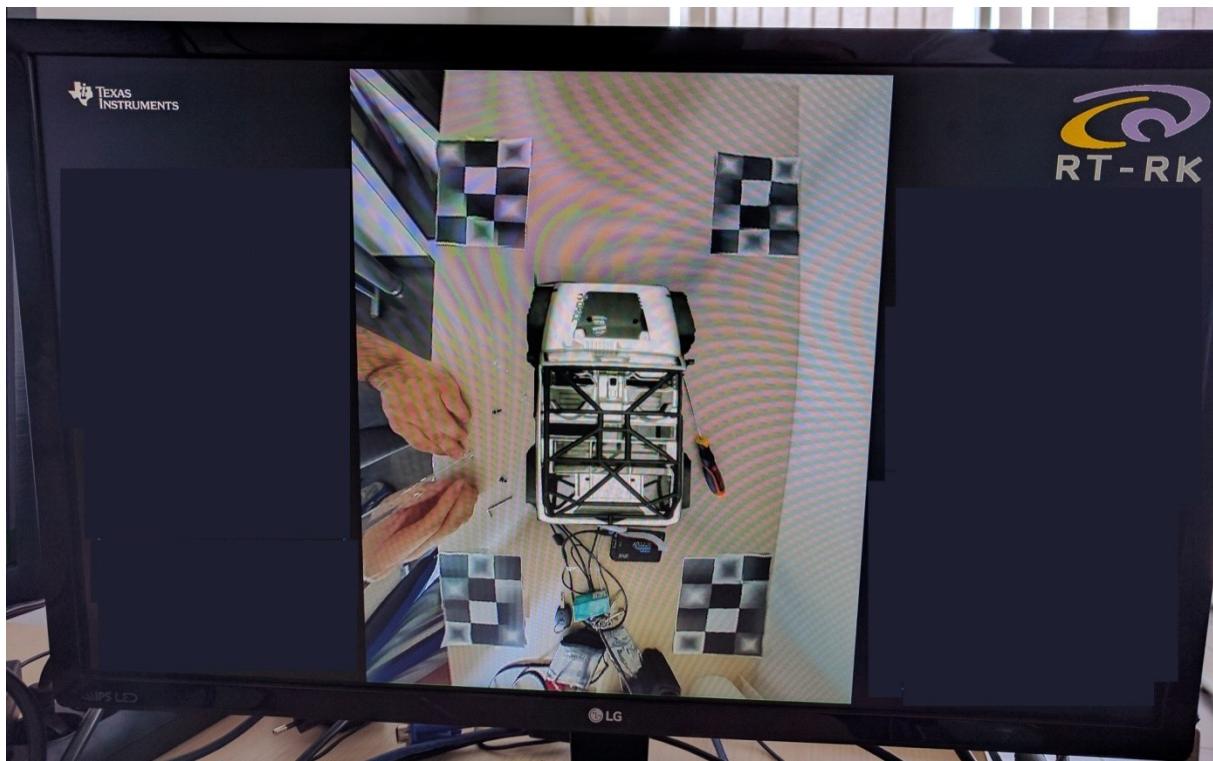
Kao i za prethodni algoritam, na Slici 8. je prikazan dijagram toka podataka.



Slika 8: Dijagram toka SV-a

*Capture* blok prikuplja podatke sa kamera. *Select* blok pristigle podatke proslijeđuje na više izlaza u unaprijed definisanom rasporedu. Sledeći u nizu je *Sync* blok. Njegova svrha

jest da grupiše slike u jedan bafer na osnovu vremenskih oznaka za hvatanje. Ovi podaci, idu dalje i obrađuju se kroz zadati algoritam i prosleđuju prema monitoru. Na kraju algoritma, koji se završava sa *Display\_sv* blokom, prosleđuje se sadržaj na video kanal, koji se dalje prosleđuje na izlaz. U desnom dijelu na slici, nalazi se blok *GrpxSrc*, (eng. *Graphics Source*), izvor grafičkog sadržaja. Nakon formiranja, baferi sa ovim sadržajem prosleđuju se bloku *Display\_GrpX*, koji ih dalje prosleđuje displej podsistemu, tačnije grafičkom kanalu. Na ekranu, dobijamo sliku na kojoj se u samom centru nalazi vozilo, a oko njega podaci prikupljeni sa kamera. Odnosno, sve ono sto okružuje automobil.



Slika 9: Aplikacija SV - pregled okruženja vozila

## 3.2 Ideja za naizmjenično izvršavanje

Prvi pokušaji realizacije naizmjeničnog izvršavanja aplikacija nisu bili dovoljno dobri. Ideja je bila da se prebacivanje sa jednog na drugi algoritam izvršava pomoću restart tastera na ploči. Pritisak na ovaj taster dolazi do potpunog gašenje i ponovnog paljenja SoC-a, a na ekranu smo dobijali drugu aplikaciju. Ovo i jeste bio cilj, ali njegova mana je što je trajalo predugo ponovno gašenje SoC-a. Iz tih razloga se tražilo novo rješenje koje će da bude mnogo brže.

Rezultat koji smo htjeli da postignemo sa novim rješenjem jeste da pritisak na taster omogućava brzo prebacivanje sa rada jednog algoritma, npr CMS-a, na rad drugog, SV, i obrnuto. Čitav ovaj proces treba da radi u realnom vremenu, bez kašnjenja. Osnovna ideja je bila da napravimo funkciju sa kojom ćemo moći da prebacujemo sa jednog na drugi algoritam, bez ponovnog učitavanja u memoriju.

Realizaciju ovog problema smo podijelili na više ciljeva. Prvi cilj je da svaki use-case posebno, nakon pokretanja, možemo u bilo kom trenutku da zaustavimo i ponovo pokrenemo u realnom vremenu. Odnosno, da stopiramo slanje podataka sa kamere, uvezvi u obzir da blokovi neće ništa raditi ako im se podaci ne proslede. Paljenje i gašenje samih kamera nismo ni uzimali u razmatranje, zato što oduzima mnogo vremena. Ideja je da se napravi blok koji će da kontroliše dotok slika sa kamera. Nakon uspješne realizacije ovog cilja, sledeće je bilo da se napravi novi use-case koji sadrži oba algoritma, CMS i SV. Obzirom da ova dva algoritma koriste iste resurse, npr kamera, ekran, procesorska jezgra, sledeći cilj je omogućavanje naizmjeničnog korišćenja tih resursa.

Sledeći cilj je da se nađe način za kontrolisanje rada aplikacija, da li želimo da koristimo CMS aplikaciju ili SV. Odnosno, šta želimo da nam bude na izlazu, ekranu, da li da se nastavi obrada podataka kroz CMS algoritam ili SV, tj. kontrola cijelog use-case-a.

## 4. Programsko rješenje

U ovom poglavlju biće priče o procesu pisanja novog use-case-a i switch funkcije. Novi use-case objedinjava dvije prethodno pomenute aplikacije SV I CMS, dok switch funkcija omogućavanja njihovo naizmjenično izvršavanje.

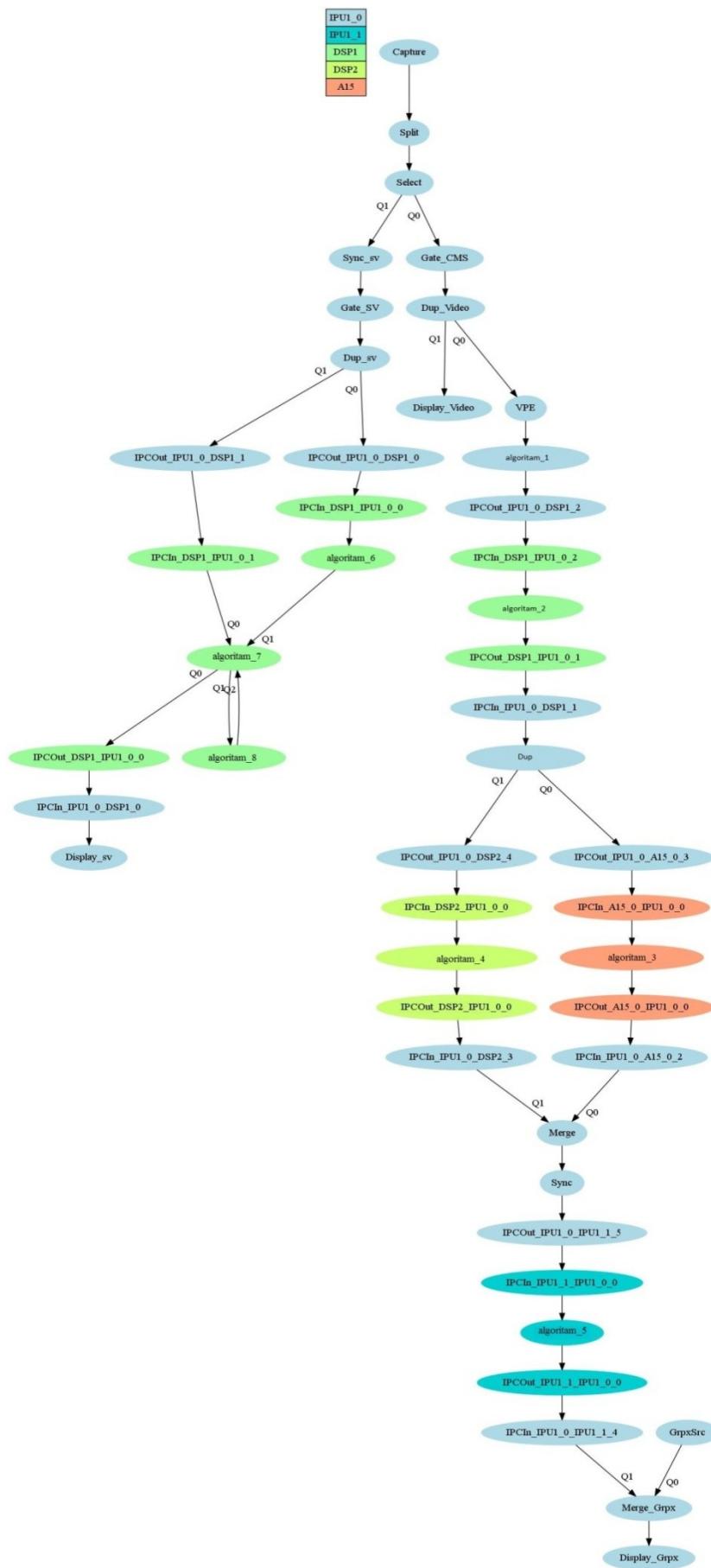
### 4.1 Proces spajanja CMS i SV aplikacija

Proučavanjem svih blokova sa kojima raspolaze VSDK okruženje, dolazimo do zaključka da bi trebalo da je najbolje rješenje da od dva zasebna algoritma, CMS i SV, napravimo jedan. Za ulazne blokove koji će da kontrolisu podatke dobijene sa kamera smo uzeli *Capture*, *Split* i *Select*, o kojima će biti nešto više riječi u daljem tekstu. Sledeći blok koji nam je bio potreban jeste *Gate* blok, a zatim i sami algoritmi. Čitav ovaj dijagram toka treba da se završi sa *Display* blokovima.

U okruženju, kakav je VSDK, za generisanje novog use case-a, potrebno je napisati .txt fajl. Taj fajl sadrži napisane blokove, onim redom kako želimo da se naš algoritam izvršava, npr:

```
Capture -> Split -> Select -> Gate_CMS -> Dup_Video -> Display_Video
```

Nakon pisanja ovog fajla, okruženje generiše ostale potrebne fajlove uz pomoć *use\_case\_gen* alata. Ovaj alat nam pomaže da brzo i lako povežemo blokove u funkcionalnu cjelinu. Preostaje da napišemo sam algoritam, kao i podešavanja generisanih blokova. Na Slici 10. je prikazan dijagram toka podataka nakon generisanja željenog use-case-a sa obje aplikacije.



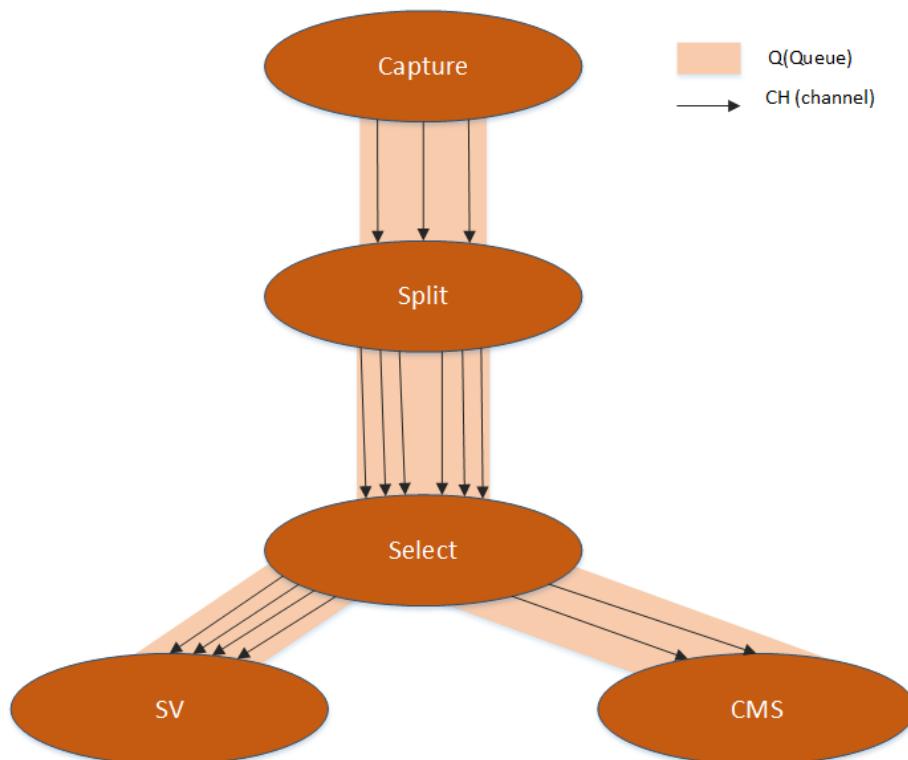
Slika 10: Dijagram toka CMS i SV zajedno

## 4.2 Ulazni podsistem

Kao što se može vidjeti, oba algoritma kreću od *Capture* bloka, koji, kako smo već naveli, prikuplja podatke sa kamera i prosleđuje dalje na obradu. U toku spajanja CMS i SV algoritma, dolazi se do ograničenja platforme da jedan use-case ne može imati više od jednog *Capture* bloka. Samim tim potrebno je dodatno izmijeniti ulazne blokove u ove algoritme. *Capture* blok je prvi blok koji je potrebno drugačije konfigurisati. On pored podešavanja mnoštva parametara, uključuje i izbor kamera putem funkcije:

```
Void ChainsCommon_Amv_SingleCam_SetSCCapturePrms(
    CaptureLink_CreateParams *pPrm,
    UInt32 captureInWidth,
    UInt32 captureInHeight,
    UInt32 captureOutWidth,
    UInt32 captureOutHeight,
    Chains_CaptureSrc captureSrc,
    SC_VideoSensor_Id sensorId),
```

Na ovaj način se prosleđuje identifikacioni broj kamera koje želimo da uključimo (u ovom slučaju, to je svih šest (*SC\_VIDEO\_SENSOR\_1\_2*; *SC\_VIDEO\_SENSOR\_3\_4*; *SC\_VIDEO\_SENSOR\_5\_6*)).



Slika 11: Ulazni podsistem

Na Slici 11. Su predstavljeni ulazni blokovi u zajedničkom use-case-u. Između blokova se nalaze Q-ovi. Svaki od ovih Q-ova sadrži određen broj kanala. Kanali nisu ništa drugo nego podaci dobijeni sa kamera.

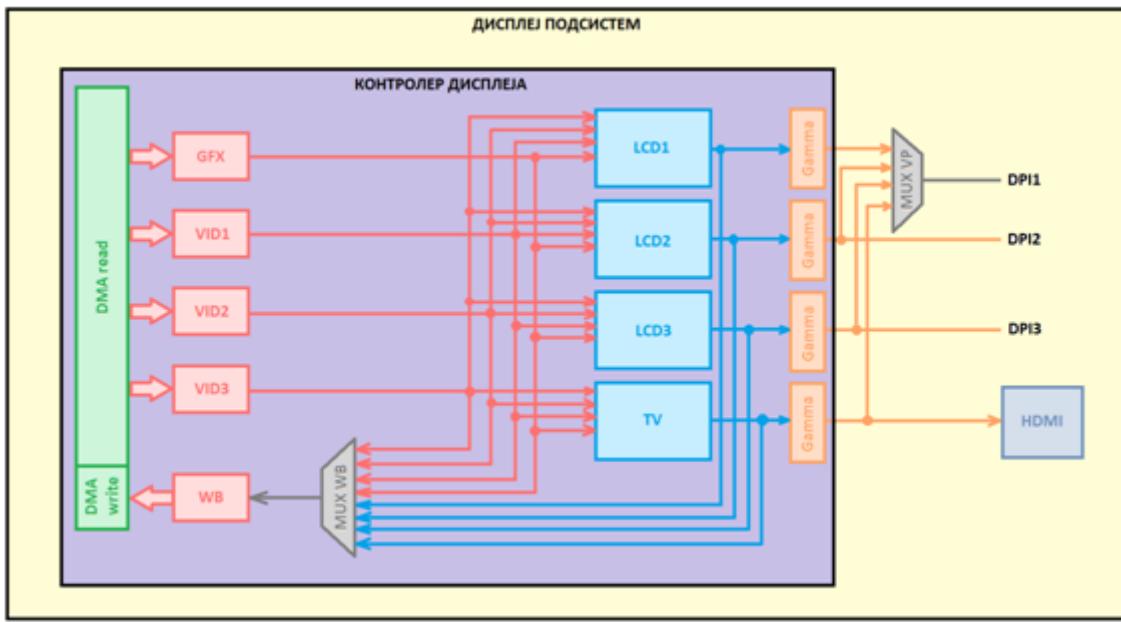
Prikupljenje podatke sa kamera šaljemo dalje na *Split* blok. *Split* dobija tri kanala, pokupljenih sa kamera, u jednom Q-u. U svakom od kanala se nalaze slike sa po dvije kamere. Zatim ih on „splituje“, tj. raščlanjuje na zasebne kanale. U našem slučaju imamo šest izlaznih kanala (CH1-CH6). Takvih šest kanala od *Split* bloka idu do *Select* bloka. *Select* je zadužen da primljene kanale prosledi na odgovarajući izlaz (Q0 ili Q1), u zavisnosti od toga kako je konfigurisan. Obzirom da je za rad SV aplikacije potrebna po jedna kamera sa sve 4 strane vozila, četiri kamere se biraju da idu na jedan izlazni Q, a 2 na drugi (za CMS).

Sa desne strane (Q0) se nalazi algoritam koji obrađuje podatke za CMS aplikaciju, dok sa lijeve strane (Q1) je algoritam za SV aplikaciju. Na Q0 prosleđujemo dva kanala (CH5 i CH6), a na Q1 četiri (CH1-CH4).

### 4.3 Izlazni podsistem

Sa Slike 10. može se vidjeti da se algoritmi završavaju sa *Display* blokovima (*Display\_Video* blok, *Display\_sv* blok i *Display\_Grp*). Ovi blokovi, *Display\_Video* i *Display\_sv*, proslijeđuju sadržaj na video kanal, koji se dalje prosleđuje monitoru, dok *Display\_Grp* iscrtava grafički sadržaj.

Kako bi obrađeni video sadržaj bio vidljiv korisniku, proslijeđuje se displej podsistemu na konačnu obradu i dalje na *HDMI* spregu, na kojoj je priključen monitor koji korisnik posmatra. Radi formiranja izlazne slike koristi se displej podistem (eng. *Display Subsystem*). U ovom podsistemu moguće je imati maksimalno tri izlazna videa i jedan grafički video. Kanali koji su predviđeni za obradu video sadržaja označeni su na slici 12. kao VID1, VID2, VID3, a GFX je kanal namjenjen grafičkoj obradi elemenata. Konačni algoritam smo morali da prilagodimo ovim uslovima.



Slika 12: Cjelokupan prikaz displej podsistema

Ciljni use-case se sastoji iz dva izlazna displeja, jedan za CMS, a drugi za SV, i jednog grafičkog displeja, u koji je spojena, preko Merge bloka, grafičko iscrtavanje od SV-a i CMS-a na jedan izlaz (Slika 13.). Time smo zadovoljili uslove koje postavlja displej podsistem.

SV aplikaciji smo dodijelili izlaz na DISPLAY\_LINK\_INST\_DSS\_VID1, a CMS aplikaciji DISPLAY\_LINK\_INST\_DSS\_VID2. Da ne bi bilo miješanja signala između ova dva algoritma prilikom poziva funkcije switch, neophodno je da se, sve za šta je vezana jedna aplikacija, prekine i da se pokrene sve ono što pokreće drugu aplikaciju i obrnuto. To znači da treba i kanale za obradu video sadržaja, VID1 i VID2, „switch-ovati“ (prebaciti sa jednog na drugi, tj. određeni kanal postaviti kao gornji i samim tim vidljiv).

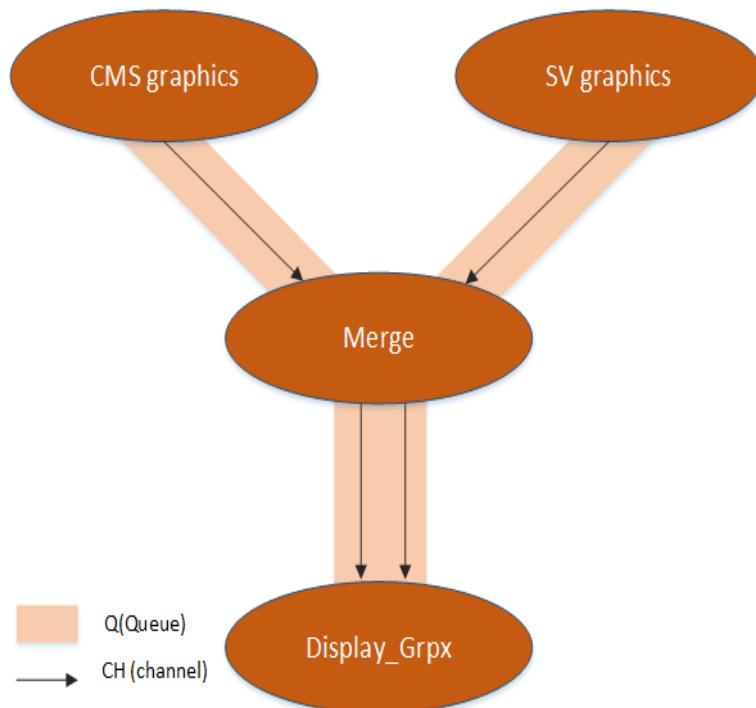
U tu svrhu realizovana je funkcija:

```
Int32 ChainsCommon_ChangeZOrder(int gornjiKanal);
```

preko koje se uspješno prebacuje sa jednog kanala na drugi, gdje `gornjiKanal = 0` jeste VID1, odnosno aplikacija SV, a `gornjiKanal = 1` je VID2, tj CMS aplikacija.

Takođe, pored prebacivanja samih video sadržaja, njih mora da prati i određeno grafičko iscrtavanje kako bi izlaz na ekran bio razumljiv svim vozačima. Isto kao što smo prebacivali same video sadržaje, tako ćemo i grafički podsistem. Ovo se radi biranjem ulaznog kanala `Display_Gpx` bloka za prikaz.

```
System_linkControl(pObj->ucObj.Display_GpxLinkID,
                    DISPLAY_LINK_CMD_SWITCH_CH,
                    &displayPrm,
                    sizeof(displayPrm),
                    TRUE);
```



Slika 13: Blok dijagram grafičkog podsistema

## 4.4 Gate blok

U nastavku možemo vidjeti podešavanje parametara za Gate blok, koji služi za kontrolisanje koja će aplikacija da se izvršava. Gate blok, kao što mu i sam naziv kaže, ponaša se kao kapija. Kada „se otvore vrata“, podaci pristigli sa kamera, se dalje prosleđuju kroz Gate blok. Gate ono što dobije na ulazu, proslijedi dalje na izlaz bez ikakve obrade pristiglih podataka.

```
Void chains_parallel_SetGatePrms(GateLink_CreateParams *pPrm_GateSV,
                                    GateLink_CreateParams *pPrm_GateCMS)

{
    pPrm_GateSV->prevLinkIsCreated = TRUE;
    pPrm_GateCMS->prevLinkIsCreated = TRUE;
}
```

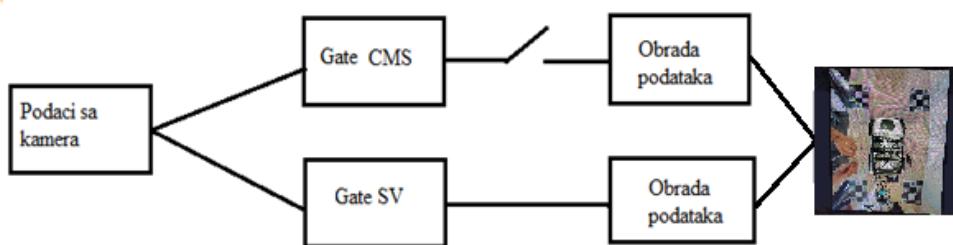
Pri smjenjivanju algoritama otvaramo i zatvaramo „kapiju“ za podatke, za određeni algoritam

```
System_linkControl(chainsObj.ucObj.Gate_SVLinkID,
    GATE_LINK_CMD_SET_OPERATION_MODE_ON,
    NULL,
    0,
    TRUE);
```

Na ovaj način smo otvorili kapiju SV algoritma i na displeju možemo da vidimo da se zaista podaci obrađuju za ovu aplikaciju.

```
System_linkControl(chainsObj.ucObj.Gate_CMSLinkID,
    GATE_LINK_CMD_SET_OPERATION_MODE_OFF,
    NULL,
    0,
    TRUE);
```

Na ovaj način zatvaramo prolaz za podatke da se obrađuju za CMS algoritam. Takođe, sve ovo važi i u obrnutom slučaju, da se podaci prosleđuju kroz Gate\_CMS, a ne kroz GATE\_SV. U tom slučaju dobijamo na izlazu Surround View aplikaciju, što može da se vidi na Slici 14. Zbog nedostatka resursa, potrebno je da jedna kapija bude otvorena, a druga zatvorena. U slučaju da se obje otvore, algoritmi neće više raditi u realnom vremenu.



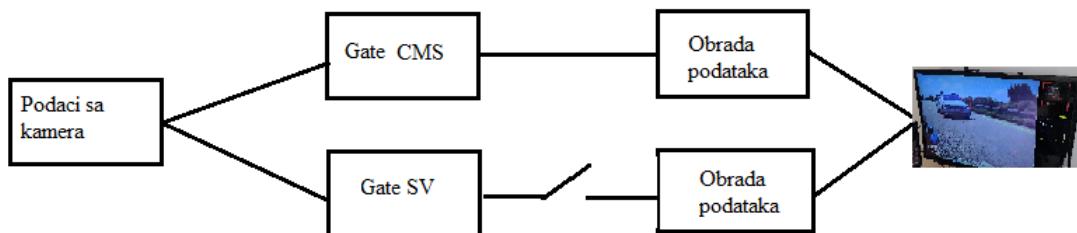
Slika 14: Prikaz Gate\_CMS bloka kao prekidača

Zatvaranje kapije za dalji protok podataka za SV aplikaciju, zatim otvaranje kapije za protok podataka za CMS aplikaciju, implementira naizmjenično izvršavanje

```
System_linkControl(  
    chainsObj.ucObj.Gate_SVLinkID,  
    GATE_LINK_CMD_SET_OPERATION_MODE_OFF,  
    NULL,  
    0,  
    TRUE);
```

```
System_linkControl(  
    chainsObj.ucObj.Gate_CMSLinkID,  
    GATE_LINK_CMD_SET_OPERATION_MODE_ON,  
    NULL,  
    0,  
    TRUE);
```

Na ovaj način radi naša funkcija, zatvori kapiju za dalji protok podataka (u ovom primjeru je to za SV aplikaciju), a otvoriti kapiju za protok podataka za CMS aplikaciju.



Slika 15: Prikaz Gate\_SV bloka kao prekidača

Na Slici 15. prikazan je obrnut slučaj. Prekidač je otvoren kod Gate SV bloka, što znači da je dalji protok podataka zaustavljen i na monitoru se dobija *Camera mirror system* aplikacija.

## 5. Rezultati

Testiranje i realizacija obavljeni su na ALPHA ploči, na prvom SoC-u. Cilj da se aplikacije učitavaju na početku i pokreću se i zaustavljaju bez ponovnog učitavanja iz stalne memorije, kako bi se dobilo prebacivanje sa jedne na drugu aplikaciju, u realnom vremenu.

U toku testiranja možemo da zaključimo da zauzetost procesorskih jezgara nije znatno promijenjeno sa realizacijom switch funkcije, u odnosu na zauzetost jezgara bez ove realizacije, što možemo vidjeti u Tabeli 1. i Tabeli 2. Povećanje zauzeća procesorskih jezgara za par procenata na jezgrima A15 i IPU je očekivano, obzirom da su novi blokovi koji su dodati u use-case stavljeni da se izvršavaju na tim jezgrima.

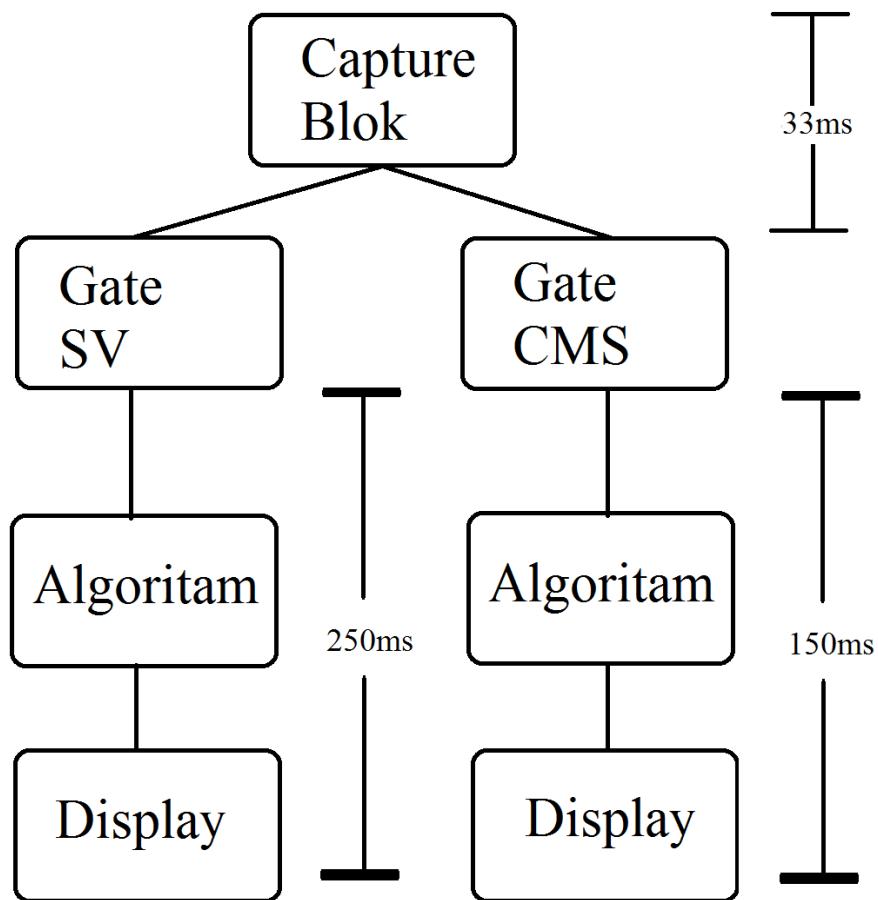
Procesorska jezgra	Bez realizacije switch funkcije	Sa realizacijom switch funkcije
<b>A15</b>	19%	22%
<b>DSP1</b>	75%	85%
<b>DSP2</b>	25%	30%
<b>IPU 1-0</b>	12%	17%
<b>IPU 1-1</b>	3%	6%

Tabela 1: Zauzetost procesorskih jezgara Surround view

Procesorska jezgra	Bez realizacije switch funkcije	Sa realizacijom switch funkcije
<b>A15</b>	64%	66%
<b>DSP1</b>	80%	98%
<b>DSP2</b>	76%	85%
<b>IPU 1-0</b>	17%	22%
<b>IPU 1-1</b>	10%	13%

Tabela 2: Zauzetost procesorskih jezgara Camera mirror system

Na slici 16. je prikazano vrijeme potrebno da se prebaci na drugi algoritam. Kao što je na slici prikazano, na svakih 33ms sa kamera pristiže jedna slika. Ako se slika pusti kroz Surround view protočnu strukturu, potrebno vrijeme da pređe od *Gate* do *Display* bloka je 250ms, dok isto za Camera mirror system algoritam traje 150ms. To znači da je toliko vremena potrebno da se pojavi slika na monitoru, nakon prebacivanja na određeni algoritam. Maksimalno vrijeme potrebno da se prebaci na CMS u ovom rješenju je približno 283ms, a na SV 183ms. U slučaju nekog drugog algoritma to vrijeme će zavisiti od toga koliko je tom algoritmu potrebno da propusti prvu sliku kroz protočnu strukturu.



Slika 16: Vrijeme prebacivanja na drugi algoritam

## 6. Zaključak

U ovom radu je prikazan način na koji se aplikacije u VSDK mogu naizmjenično izvršavati. Cilj formiranja ovakve aplikacije jeste nedostatak resursa da se sve aplikacije istovremeno pokreću na jednoj ploči, kao i što nema potrebe da sve aplikacije rade sve vrijeme.

Prva realizacija, da se na reset taster mijenjaju aplikacije nije ispunjavala sve potrebne uslove. Pritiskom na taster se isključuje i uključuje SoC, sa drugom aplikacijom na ekranu. Ponovno učitavanje u memoriju traje isuviše dugo. Problemi koje je imala prva realizacija su riješeni u realizaciji koja je opisana u ovom radu.

Upotrebom *Gate* blokova i modifikacijom ulaznih i izlaznih podsistema uspješno je omogućeno da se aplikacije naizmjenično izvršavaju, odnosno da se aplikacije učitavaju na početku i pokreću se i zaustavljaju bez ponovnog učitavanja iz memorije. Novi use-case smo prilagodili ograničenjima platforme. Jedno od tih ograničenja je konačan broj video i grafičkih display-a.

Testiranjem je utvrđeno da vrijeme prebacivanja na neki algoritam zavisi od samog algoritma, tj. od vremena prolaska slike kroz protočnu strukturu algoritma.

Iz ovog rada možemo zaključiti da je moguće na ovaj način učitati veći broj algoritama u memoriju, i kontrolisati njihovo izvršavanje, kao i brzo mijenjati aktivni algoritam.

## 7. Literatura

- [1] TI Vision SDK, Optimized Vision Libraries for ADAS Systems,  
<http://www.ti.com/lit/wp/spry260/spry260.pdf>, učitano 13.07.2017.
- [2] Automotive machine vision alpha reference board on Texas Instruments SoCs <http://www.rt-rk.com/news/195-rt-rk-presents-automotive-machine-vision-alpha-reference-board-on-texas-instruments-socs>, učitano 13.07.2017.
- [3] Advanced Driver Assistance System (ADAS) <http://www.ti.com/lsds/ti/applications/automotive/adas/overview.page>, učitano 13.07.2017.
- [4] A. Hakanović, "Sistemi za pomoć vozaču", Sarajevo, 2011
- [5] TDA2x ADAS System-on-Chip, <http://www.ti.com/lit/ml/sprt681/sprt681.pdf>, učitano 13.07.2017.