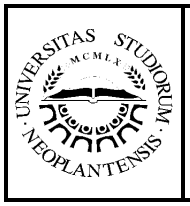


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

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

ДИПЛОМСКИ – МАСТЕР РАД

Кандидат: **Vladimir Jovanović**
Број индекса: **11807**

Тема рада: **Jedno rešenje programske podrške za automatsko generisanje koda za digitalne kamere na bazi Matlab/Simulink paketa**

Ментор рада: **prof.dr. Nikola Teslić**

Нови Сад, jul 2009

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**:		Владимир Јовановић
Ментор, **МН**:		проф. др. Никола Теслић
Наслов рада, **НР**:		Једно решење програмске подршке за аутоматско генерисање кода на бази Матлаб/Симулинк пакета
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2009
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/		
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кqучне речи, **ПО**:		
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		
Датум прихватања теме, **ДП**:		24.06.2009
Датум одбране, **ДО**:		17.07.2009
Чланови комисије, **КО**:	Председник:	dr. Zlokolica Vladimir
	Члан:	dr. Nebojša Pjevalica
	Члан, ментор:	prof.dr. Nikola Teslić

Потпис ментора

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

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Vladimir Jovanović	
Mentor, **MN**:		PhD Nikola Teslić	
Title, **TI**:		One Software solution for automatically genereted code for digital camera based on Matlab/Simulink package	
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**:		2009	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)			
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:			
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:			
Accepted by the Scientific Board on, **ASB**:		24.06.2009	
Defended on, **DE**:		17.07.2009	
Defended Board, **DB**:	President:	PhD. Zlokolica Vladimir	
	Member:	PhD. Nebojša Pjevalica	Menthor's sign
	Member, Mentor:	PhD. Nikola Teslić	

Sadržaj

Skraćenice

CCD - Charge-Coupled Device

CCS - Code Composer Studio

CMOS - Complementary Metal–Oxide Semiconductor

CSL - Chip Support Library

DDR - Double Data Rate

DAC - Digital to Analog Converter

DRAM - Dynamic RAM

DSP - Digital Signal Processor

EMIF - External Memory Interface

EPROM - Erasable Programmable ROM

EVM - Evaluation module

I^2C - Inter-Integrated Circuit

IDE - Integrated Development Enviroment

ISRAM - Internal SRAM, Interna statička RAM memorija (unutar DSP)

JTAG - Joint Test Action Group

LAN - Local Area Network

LED - Light Emitting Diode

NTSC - National Television System(s) Committee

OSD - On Screen Display

PAL - Phase Alternation Line

PC - Personal Computer

PCI - Periferal Component Interconect

PCM - Pulse Code Modulation

PWM - Pulse Width Modulation

RAM - Random Access Memory

RGB - Red-Green-Blue output

ROM - Read-Only Memory

SRAM - Static Random Access Memory

TI - Texas Instruments

1 Uvod

Sa porastom upotrebe računara raste potreba i korišćenje digitalne obrade signala. Digitalna obrada signala (DSP) je nauka koja se bavi signalima u digitalnoj reprezentaciji i metodima njihove obrade. Algoritmi koji se koriste kod digitalne obrade signala se često implementiraju na specijalizovanim mikroprocesorima koji se zovu "procesori za digitalnu obradu signala" (takođe DSP). Kod njih se obrada signala izvršava u realnom vremenu.

Programiranjem DSP procesora u asembleru je najpribližnije mašinskom jeziku, ali mane takvog programiranja su loša čitljivost, kompleksnost koda i nemogućnost konvertovanja na drugu procesorsku platformu. Zbog toga se koriste viši programski jezici kao što je C programski jezik. Pisanje u višem programskom jeziku za DSP procesore donekle je oslobođeno vođenje računa o svakom registru i toku podataka ali još uvek je potrebno poznavanje platforme zbog specifičnih načina rada i korišćenja resursa.

Za pisanje programske podrške za određenu platformu potrebno je poznavati podržani programski jezik i arhitekturu platforme. Automatskim generisanjem koda se skraćuje vreme izrade i uprošćava samo programiranje. Kod obrade slike, odnosno razvoja nekog algoritma na ovaj način, odmah može da se proveri rad algoritma na nekoj fizičkoj arhitekturi. U ovom radu je prikazano jedno rešenje automatskog generisanja koda.

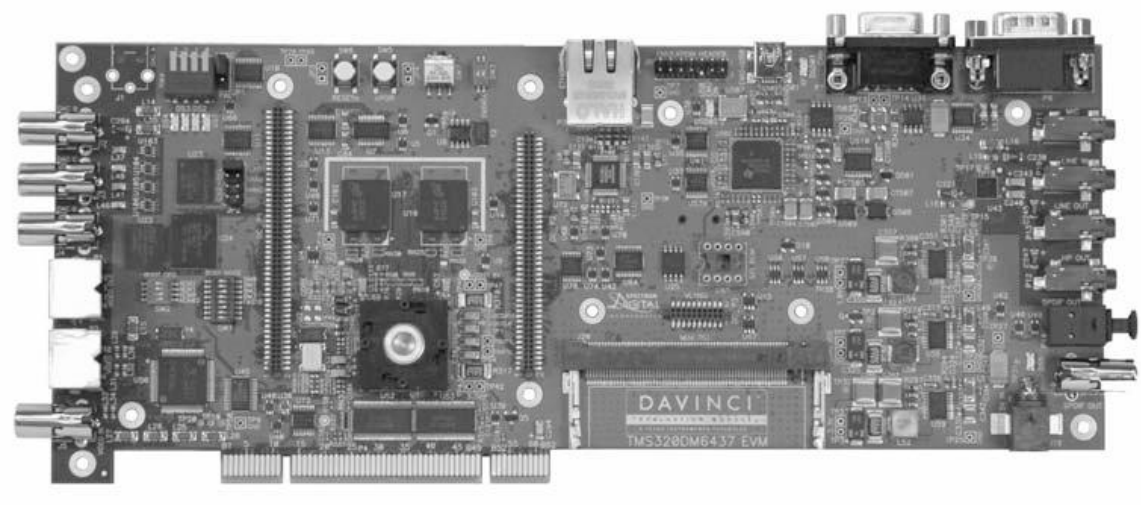
Zadatak ovog projekta je realizacija programske podrške za automatsko generisanje koda za digitalni CMOS senzor na bazi Matlab/Simulink paketa. Rešenje je realizovano na razvojnoj ploči TMS320DM6437 EVM (eng.Evaluation module), koja je već podržana u programskom paketu Matlab/Simulink i za koju je moguće generisati programski kod za predefinisane gradivne blokove Simulinka, vezanih za mogućnost razvojne ploče i opštih funkcija DSP-a. Realizovan je gradivni blok koji omogućuje upotrebu CMOS senzora na ciljnoj platformi u Simulink okruženju, kao i gradivni blokovi za generisanje odgovarajućeg programskog koda koji omogućuju upotrebu ugrađenih modula procesora koji podržavaju: skaliranje slike, određivanje histograma, i podrške za automatski fokus i automatsko osvetljenje i balans belog.

2 Opis platforme i razvojnog okruženja

2.1 Opis fizičke arhitekture

2.1.1 Razvojna ploča DM6437 EVM

Razvojna platforma TMS320DM6437 EVM je PCI bazirana ili samostalna razvojna platforma, koja omogućava korisnicima razvoj i ispitivanje aplikacija za TI (eng.Texas Instruments) *DaVinci* familiju procesora. EVM dolazi sa velikim rasponom komponenti za različite zahteve hardverskog okruženja što omogućava evaluaciju određenih karakteristika DM6437 da bi se odredilo da li procesor zadovoljava zahteve aplikacije a takođe služi i kao referenca za kreiranje ciljne platforme za DM6437 DSP.



Slika 1 – Izgled DM6437 EVM ploče

Glavni elementi koje ploča sadrži su:

• *Texas Instruments* DM6437 procesor koji radi na frekvenciji do 600 Mhz.

• TVP5146M2 video dekoder, podržava kompozitni i s-video ulaz

• 4 video DAC izlaza - komponentni, RGB, kompozitni

• 128 MB DDR2 DRAM memorije

• UART, CAN I/O sprega

• 16 MB NOR fleš memorije, 64 MB NAND fleš, 2 MB SRAM

• AIC33 stereo kodek

• I^2C sprega

• 10/100 MBS mrežna sprega

• Podesive opcije za podizanje (eng. Boot) sistema

• Ugrađena JTAG sprega

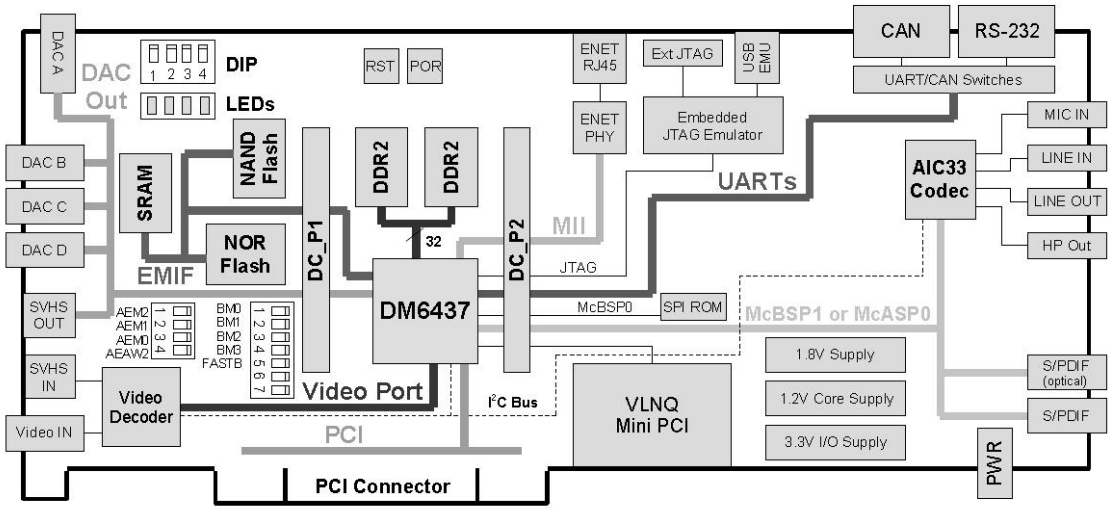
• 4 LED i 4 preklopnika za potrebe korisnika

• Napajanje (+5V)

• Priključci za dodatne kartice i proširivanje mogućnosti

• VLYNQ sprega

• S/PDIF sprega, analogna i optička



Slika 2 – Dijagram DM6437 EVM ploče

Procesor DM6437 kao glavni element na EVM ploči (Slika 2) je povezan sa periferijama preko ugrađenih sprega i osmobitne EMIF (eng. External Memory Interface) magistrale. DDR2 memorija je povezana na svoju dodeljenu 32-bitnu magistralu. EMIF magistrala može biti preko kratkospojnika povezana na NOR, SRAM, NAND memoriju ili karticu za proširenje.

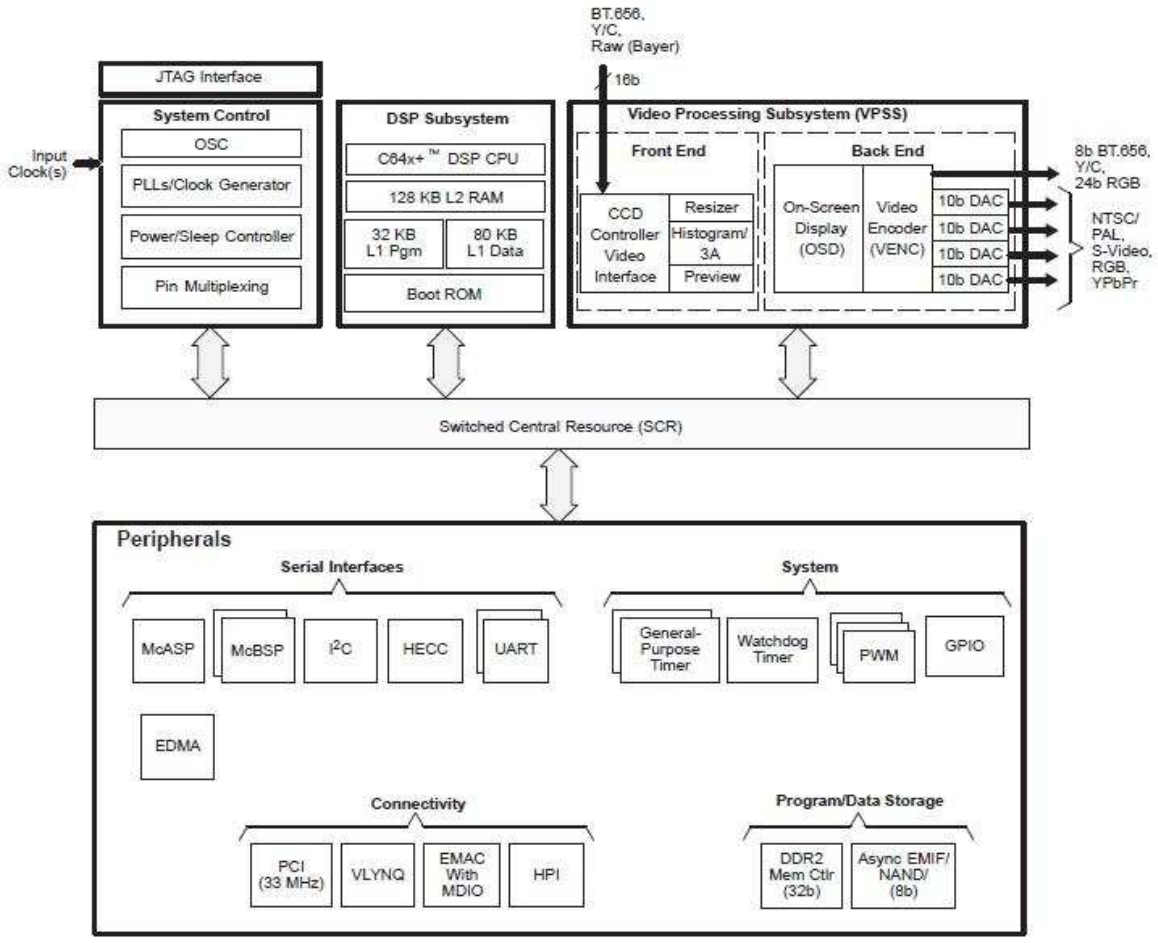
Video dekoder (TVP5146M2) predstavlja ulaznu video spregu za kodirane video signale a enkoderi (4 DAC kanala) u čipu predstavljaju izlaznu video spregu za procesor. AIC33 kodek omogučuje DSP-u da prima i šalje analogne audio signale. Ploča sadrži 4 LED i 4 DIP prekidača za slobodnu upotrebu od strane korisnika. Za standardnu komunikaciju sa

dodatnim uređajima postoji mogućnost za CAN i RS232 spregu. Za direktnu spregu sa procesorom preko postojećih magistrala tu su konektori DC_P1 i DC_P2.

EVM je dizajniran da radi sa TI *Code Composer Studio* paketom za razvoj koda. *Code Composer Studio* komunicira sa pločom preko ugrađenog emulatora ili eksternog JTAG emulatora.

2.1.2 Procesor DM6437

Sam procesor serije TMS320 DM6437 pripada TMS320C64x+ DSP generaciji najviših performansi sa izvršenjem matematičkih operacija u nepokretnom zarezu. DM6437 procesor je baziran na trećoj generaciji procesora visokog nivoa performansi sa naprednom VLIW (eng. Very Long Instruction Word) arhitekturom razvijenoj od strane TI, što čini ovaj procesor odličanim izborom za digitalnu audio-video obradu. C64+ podržava dodatne funkcije i proširene instrukcije iz predhodnih serija a izvršava do 5600 miliona instrukcija u sekundi (MIPS) na taktu od 700 MHz.

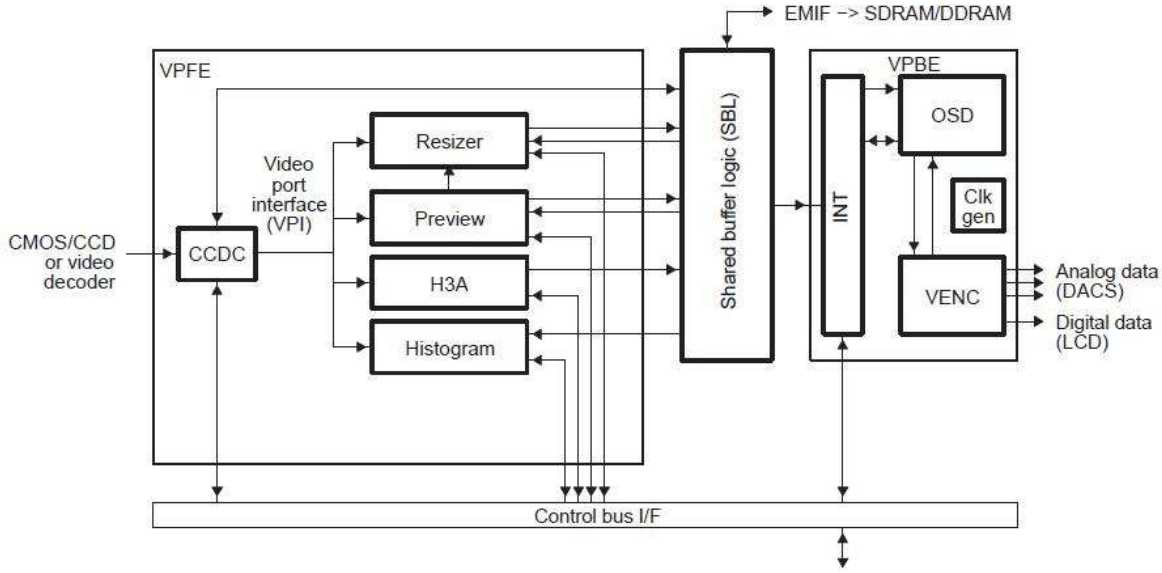


Slika 3 – Funkcionalni blok dijagram DM6437 procesora

DM6437 koristi skrivenu (eng.Cache) memoriju u dva nivoa. Nivo jedan programske keš memorije L1P sadrži 32KB, a L1D 80KB. Nivo dva L2 sadrži 128KB memorije koji je deljen između programskog prostora i prostora za podatake.

Moduli procesora uključuju: 2 video sprege, 10/100 Mb/s mrežnu spregu (EMAC) sa ulazno/izlaznim kontrolnim modulom (MDIO) , VLYNQ sprega, I^2C sprega, 2 UART modula, 3 PWM, CAN sprega, PCI EMIFA[33 MHz], DDR2 dodeljenu sinhronu magistralu, 111 pina za korisnički ulaz/izlaz (GPIO), dva 64-bitna tajmera...

DM6437 procesor sadrži video procesni podsistem VPSS(eng. Video Processing SubSystem) sa dva modula: prednja video obrada VPFE (eng. Video Processing Front End) za obradu ulaznog video signala i zadnja video obrada VPBE (eng. Video Processing Back End) za prosleđivanje i kodovanje izlaznog signala (Slika 4.).



Slika 4 – Funkcionalni dijagram video procesnog sistema VPSS

VPFE se sastoji od CCD kontrolera, preglednika slika (eng. Previewer), histogram modula, automatske ekspozicije/automatskog balansa belog/automatskog fokusa (H3A) i skalera slike.

CCD kontroler omogućava spregu sa video dekoderima, CMOS i CCD senzorima. Služi za komunikaciju i formatiranje podataka. Iz CCD kontrolera slika može da se prosledi u neki od narednih modula za obradu ili da bude zapisana u memoriju.

IPIPE (eng. The preview engine image pipe - Previewer) je modul za procesiranje slike u realnom vremenu, koji prima podatke slike u sirovom obliku iz CMOS/CCD senzora ili iz memorije i pretvara je iz bajer formata u YUV 4:2:2.

Histogram i H3A moduli omogućuju statističku informaciju o slici, koja kasnije može da se upotrebi za poboljšanu obradu slike. H3A može da se iskoristi za automatizaciju tri parametara slike: fokus, osvetljenje i balans belog

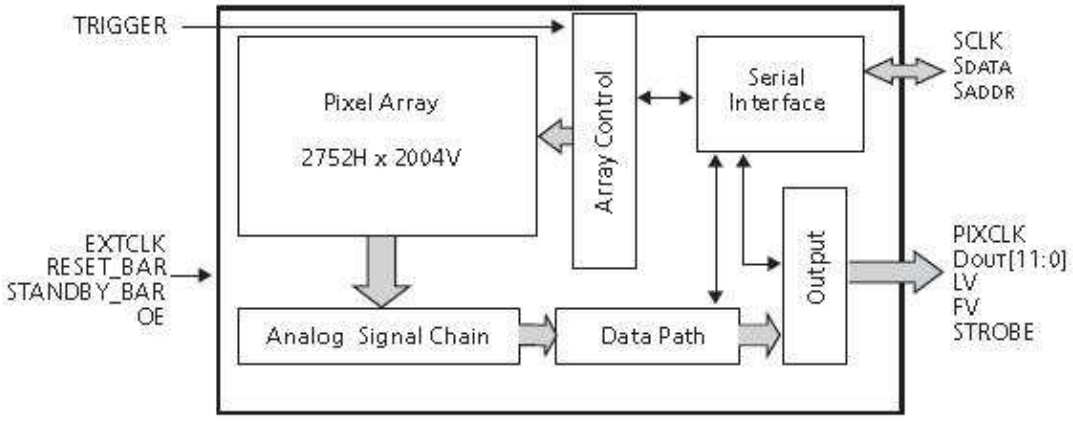
Skaler slike prihvata sliku koju skalira horizontalno i vertikalno od 1/4x do 4x, koracima od 256/N gde je N između 64 i 1024.

VPBE sadrži OSD (eng. On-Screen Display) modul i video encoder VENC (eng. Video Encoder). VPBE modul može da kombinuje prikaz dva video prozora i dva OSD prozora. VENC komponenta može uz pomoć četiri digitalno analogna DA (eng. Digital Analog) konvertora na 54MHz da prosledi sliku kao NTSC/PAL video, S-video, ili komponentni video izlaz.

2.1.3 CMOS senzor MT9P031

Digitalni senzor MT9P031 je 1/25 inčni (4:3) 5Mp CMOS senzor sa aktivnim poljem od 2592Hx1944V u koji su ugrađene neke funkcije kamere i koji je moguće programirati preko I^2C sprege.

MT0P031 senzor može da radi u podrazumevanom modu ili da po potrebi budu programirane podržane opcije kao što su veličina slike, ekspozicija, pojačanja signala i drugih parametara. Analogno digitalni konvertor na čipu ADC (eng. Analog Digital Converter) prosleđuje signal Dout[11:0] širine 12-bita, koji predstavlja vrednost boje određene tačke. Signali FRAME_VALID, LINE_VALID i STROBE kao i PIXEL CLOCK služe za sinhronizaciju validnih podataka (Slika 5).

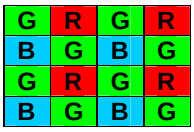


Slika 5 – Funkcionalni dijagram CMOS senzora

MT9P031 senzor kao izlaz daje progresivnu sliku konstatnom frekvencijom, koju generiše PLL (eng. Phase Locked Loop) elektronski sklop na osnovu ulaznog takta koji može biti od 6 do 27 MHz. Maksimalna brzina očitavanja podataka je 96 000 tačaka u sekundi (96 Mp/s), što se postiže na taktu 96 MHz. Interakcija korisnika sa senzorom se obavlja preko I^2C sprege, kojom se podešavaju registri senzora odnosno način rada i

parametri slike. Vrednost tačaka se očitava po kolonama u Bajer formatu (eng. Bayer pattern).

Bajer format je stadardni format koji se upotrebljava kod CMOS/CCD senzora. Ovaj format je sastavljen (Slika 6) od niza vrednosti za boje i to u jednom redu zelene i crvene (GRGRG...) u narednom zelene i plave (GBGBG...). Što znači da pola podataka predstavlja vrednosti zelene boje a po jedna četvrtina crvene i plave, što je urađeno zbog veće osetljivosti oka na zelenu boju. Pošto za svaku tačku postoji samo vrednost za jednu boju, za dobijanje podatka za celu sliku odnosno vrednosti ostalih boja, koriste se razni algoritmi za računanje što utiče i na krajnju sliku.



Slika 6 – Bajer format struktura

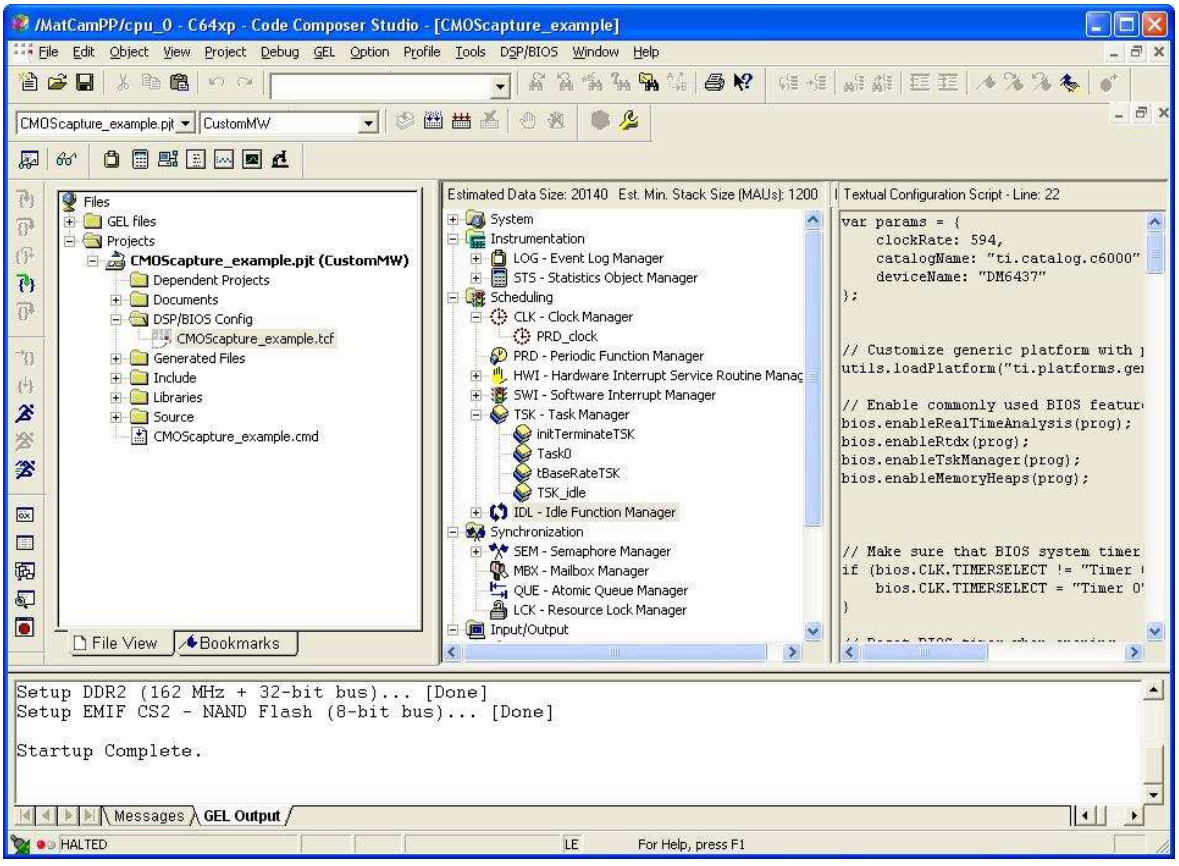
Izlazna slika je definisana sa četiri registra: *Column_Start* i *Row_Start* definišu X i Y koordinatu levog gornjeg ugla izlazne slike. Registar *Column_Size* definiše širinu, i *Row_Size* definiše visinu slike.*Column_Start* i *Row_Start* moraju biti parni brojevi. *Column_Size* and *Row_Size* polja moraju biti neparni brojevi što rezultuje da veličina slike bude paran broj. Rezolucija slike može biti smanjena sa dve metode: *skiping* (eng. Skipping) i *bining* (eng. Binning). *Skiping* metodom rezolucija se smanjuje preskakanjem celih redova ili kolona, što može posebno da se podesi. *Bining* metoda smanjuje rezoluciju kombinovanjem susednih tačaka iste boje da predstavi jednu izlaznu tačku. Na ovaj način sve tačke učestvuju u izlaznoj slici, što rezultuje boljom slikom sa manje artefakata. Time se takođe poboljšava slika sa manje osvetljenja.

2.2 Programski paketi

2.2.1 Programski paket *Code Composer Studio*

Code Composer Studio (Slika 7) je integrisano okruženje za razvoj IDE (eng. Integrated Development Enviroment) koje donosi alate za sve platforme i podršku za razvoj projekata na bazi TMS320 DSP procesora.

IDE uključuje podršku za DSP/BIOS, mogućnost analize u realnom vremenu, alate za kontrolisano izvršavanje programa i optimizaciju, C/C++ prevodilac, asemplerski prevodilac, vizualno upravljanje projektom i razne simulatore i emulacione rukovaoce uređaja.



Slika 7 – Programski paket *Code Composer Studio*

2.2.2 Operativni sistem DSP/BIOS

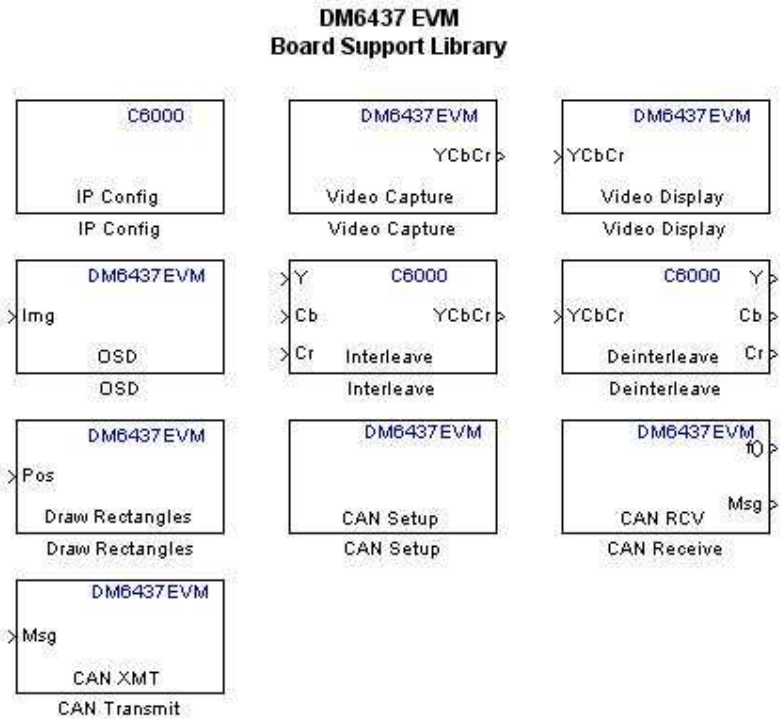
DSP/BIOS operativni sistem je sistem za rad realnom vremenu i podrškom za paraleno izvršavanje više niti, specijano projektovan za platforme na bazi procesora TMS320C6000™, TMS320C5000™, and TMS320C28x™. DSP/BIOS donosi standardizovane API (eng. Application Programming Interface) funkcije za C6000, C5000 i C28x DSP platforme i optimizovan je za korišćenje na DSP jezgrima u OMAP (eng. Open Multimedia Application Platform) uređajima. DSP/BIOS je moguće koristiti samostalano ili kao integralni deo *Code Composer Studio* IDE i uključuje grafičko podešavanje elemenata sistema. Namenjen je za rad u realnom vremenu, kreiranju, raspodeli, sa alatima za analizu u realnom vremenu fokusiran na otkrivanje grešaka i podešavanje sinhronizacije niti(eng. Thread).

2.2.3 Programski paket Matlab/Simulink

Matlab je visoko razvijeni programski jezik za tehničke proračune, koji u sebi obuhvata proračun, grafičko prikazivanje i programiranje u okruženju koje je veoma jednostavno za korišćenje i u kojem su problemi i rešenja prikazani u opšteprihvaćenoj matematičkoj notaciji. Matlab u svojoj osnovi predstavlja skript interpreter koji omogućava da se algoritmi implementiraju nezavisno od operativnog sistema. Takođe novije verzije Matlab-a sadrže alate uz pomoć kojih se vrlo jednostavno kod napisan u Matlab-u prevodi u C i C++, a biblioteke funkcija napisanih u Matlab-u lako transformišu u DLL-ove, za kasniju implementaciju u "*stand alone*" aplikacijama. Osnovna namena Matlab-a ostaje ipak razvojno okruženje koje omogućava kreiranje matematičkih aplikacija bez opterećivanja "programerskim" problemima. Matlab programski jezik ima veoma jednostavnu sintaksu, koja u osnovi podseća na C/C++ i Pascal, ali je znatno pojednostavljena i zahvaljujući interpreteru omogućava izuzetno brzo implementiranje algoritama za obavljanje raznih proračuna, koji ponekad uz primenu gotovih paketa alata (eng. toolboxes) mogu sadržati svega dva ili tri reda koda. Osim implementacije algoritama, Matlab omogućuje i akviziciju podataka, modelovanje, simuliranje, kao i razvoj aplikacija uključujući i izgradnju grafičkog korisničkog interfejsa.

Jedan od najznačajnijih paketa Matlab-a jeste Simulink, paket namenjen modelovanju, simuliranju i analizi dinamičkih sistema. Jednostavnim kreiranjem grafičkog modela može se kreirati, simulirati i analizirati veliki broj realnih dinamičkih sistema kao što su: sistemi automatskog upravljanja, električna kola, mehanički, termodinamički sistemi itd. Na ovaj paket se direktno nadovezuje RTW (eng. Real-Time Workshop), pomoću kojeg se izrađuju aplikacije bazirane na Simulink dijagramima, za ispitivanje i primenu sistema u realnom vremenu.

Simulink sadrži paket alata (eng. Target Support Package - TC6) koji predstavlja programsku podršku za DSP C6000 jezgra, u okviru kojeg je i paket alata za DM6437 EVM ploču, koji sadrži blokove za osnovne komponente na ploči (Slika 8).

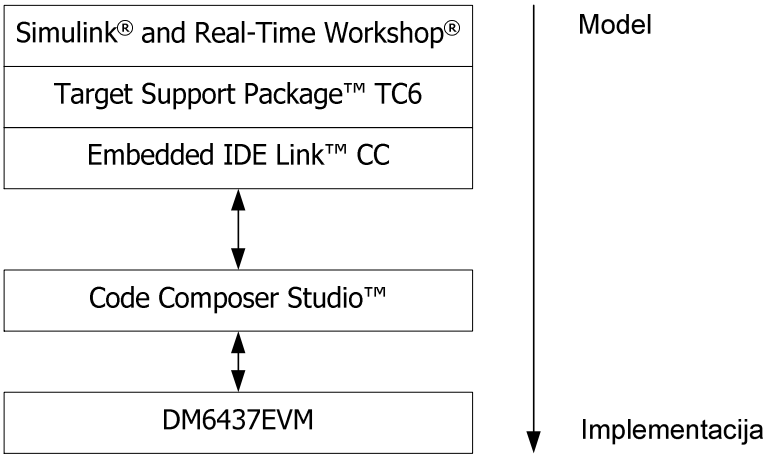


Slika 8 – Blokovi u biblioteci za programsku podršku DM6437 EVM ploči

3 Metodologija izrade

3.1 Opis načina realizacije

Putanja kojom se dolazi od modela do implementacije (Slika 9) je pravljenjem modela u Simulinku uz pomoć namenskih blokova za DM6437 EVM ploču iz TC6 paketa i generisanjem modela. Generisanjem modela Simulink se ugrađenom IDE vezom povezuje sa *Code Composer Studio* aplikacijom u kome stvara projekat, generiše potrebne kodove i pokreće prevođenje, povezivanje i spuštanje koda na DM6437 EVM ploču. Tako da je potrebno poznavati samo način rada u Simulink okruženju, u kome preko blokova definiše potrebnu obradu. Za koju se zatim generiše kod i učitava na platformu.



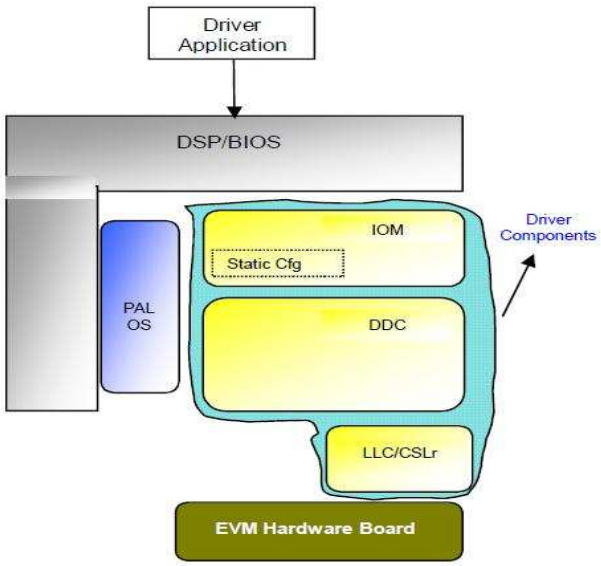
Slika 9 – Prikaz putanje od modela do realizacije

Znači za svaku obradu ili deo sistema treba da postoji blok u Simulink-u, sa svojim ulazima/izlazima, tako da može da se kombinuje sa ostalim blokovima, i potrebnih datoteka uz pomoć kojih se generiše sam kod.

Svaki blok se sastoji od:

- s-funkcije
- simulink grafičkog prikaza bloka sa maskom
- tlc skripte

Programska podrška odnosno kod (koji će biti generisan) je napisan u programskom jeziku C i realizovan je uz pomeć programske podrške koja je razvijena za DSP/BIOS operativni sistem od strane TI i koristi visok nivo apstrakcije (Slika 10).



Slika 10 –Funkcionalne komponente programske podrške

Komponente programske podrške su:

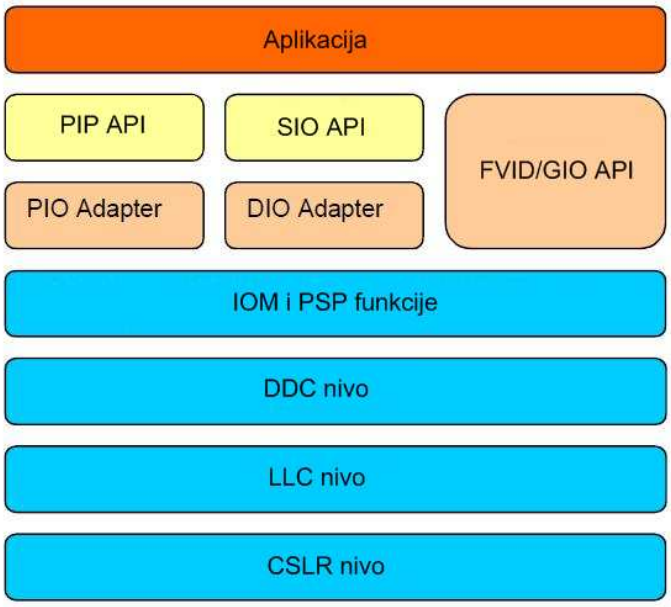
- IOM (eng. Input Output Menager)– Komponenta za pristup iz korisničke aplikacije, specifična adaptacija zavisno od operativnog sistema. IOM implementira aspekte, kao što su kreiranje i korišćenje niti, registracija prekida, komunikaciju sa definisanim funkcijama operativnog sistema.

- DDC (eng. Device Driver Core)– deo nezavistan od operativnog sistema, specifičan za platformu. Ne pristupa direktno fizičkom nivou, samo preko LLC komponente, kao što i funkcijama operativnog sistema pristupa preko IOM komponente. DDC pomaže postizanju uniformnosti API sintakse i semantike kroz sve podržane platforme i operativne sisteme.

- LLC (eng. Low Level Controller) – nivo je sastavljen od funkcija koje predstavljaju razne funkcije podržane od strane platforme. Parametre sistema podešava upisivanjem odgovarajućih vrednosti u registre. Ovaj nivo ne zavisi od operativnog sistema i ne koristi optimizaciju specifičnu datim prevodiocem.

- CSLR (eng. Chip Support Library Registers) – apstrakcija modula na nivou registara, sastavljen od simboličkih konstati koje predstavljaju registre i bit polja u samim registrima,kao i bit-maske.

- PALOS – sistemska komponenta, DSP/BIOS apstrakcija koja sadrži funkcije koje nisu mogle da se implementiraju u IOM nivo zbog performansi i prirode same funkcionalnosti.



Slika 11 – Arhitektura programske podrške

FVID/GIO API je najviši nivo apstrakcije oslonjen na IOM nivo, biblioteka funkcija preko kojih se operiše sa logičkim kanalima koji predstavljaju module u procesoru. Svaki modul je predstavljen strukturom preko koje se podešavaju parametri.

Funkcija za kreiranje logičkog kanala *FVID_create* popunjava statička podešavanja objekta odnosno parametre specificiranog modula, kreira i registruje pristupne tačke sa DPS/BIOS operativnim sistemom:

```
FVID_Handle FVID_create  ( String Name
                           Int mode
                           Int * status
                           Ptr chanParams
                           GIO_Attrs* attrs )
```

Brisanje logičkog kanala se obavlja funkcijom *FVID_delete* :

```
Int FVID_delete  ( FVID_Handle gioChan)
```

Funkcija za IO (eng.Input-Output) kontrolu logičkog kanala *FVID_control*:

```
Int  FVID_control  ( FVID_Handle gioChan
                     Int cmd,
                     Ptr cmdArg )
```

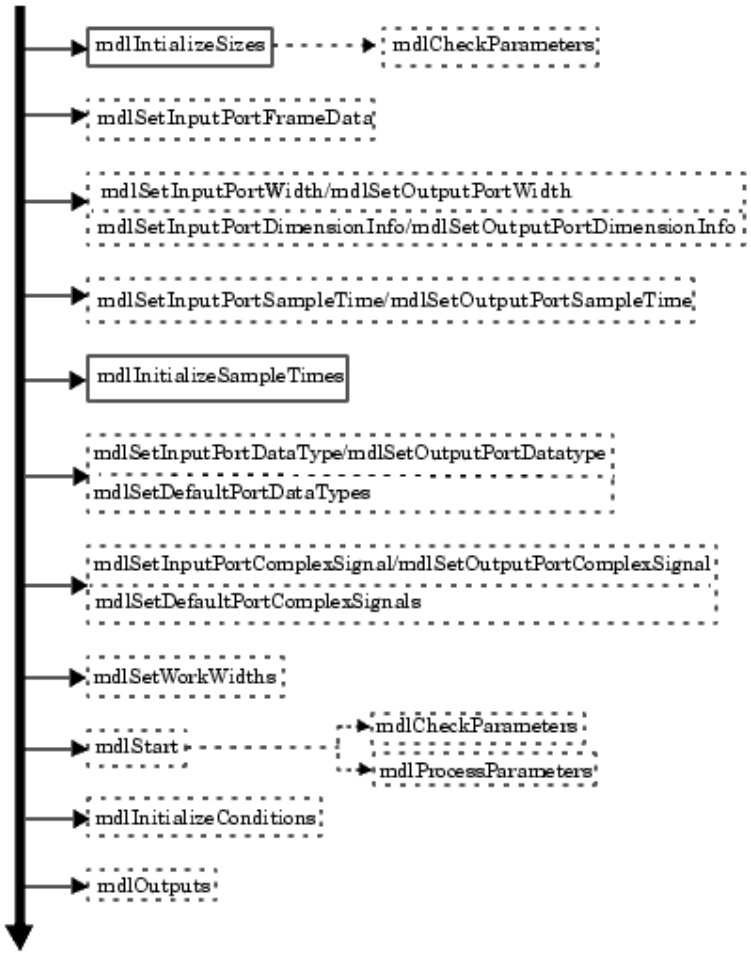
Funkcija kojom se iz aplikacije izvršava operacija upisa/čitanja *FVID_submit*:

```
Int FVID_submit ( GIO_Handle gioChan
                  Uns cmd
                  Ptr bufp
                  Uns* pSize
                  GIO_AppCallBack* appCallback )
```

3.2 S-funkcija

S-funkcija (eng. System Functions) predstavlja moćni mehanizam za proširivanje mogućnosti Simulink okruženja. S-funkcija je opis simulink bloka pisana u Matlab, C, C++, ada ili fortran programskom jeziku. C, C++, ada i fortran s-funkcije se prevode u MEX datoteke uz pomoć *mex* uslužnog programa. Kao i ostale MEX datoteke s-funkcije su dinamički povezane metode koje Matlab intepreter automatski učitava i izvršava. Prateći skup prostih pravila može se implementirati algoritam u s-funkciju i uz pomoć "S-Function" bloka postaviti u Simulink model. Posle pisanja s-funkcije i stavljanja njenog imena u "S-Function" blok može se postaviti i maska bloka, koja služi kao korisnička sprega za postavljanje parametara.

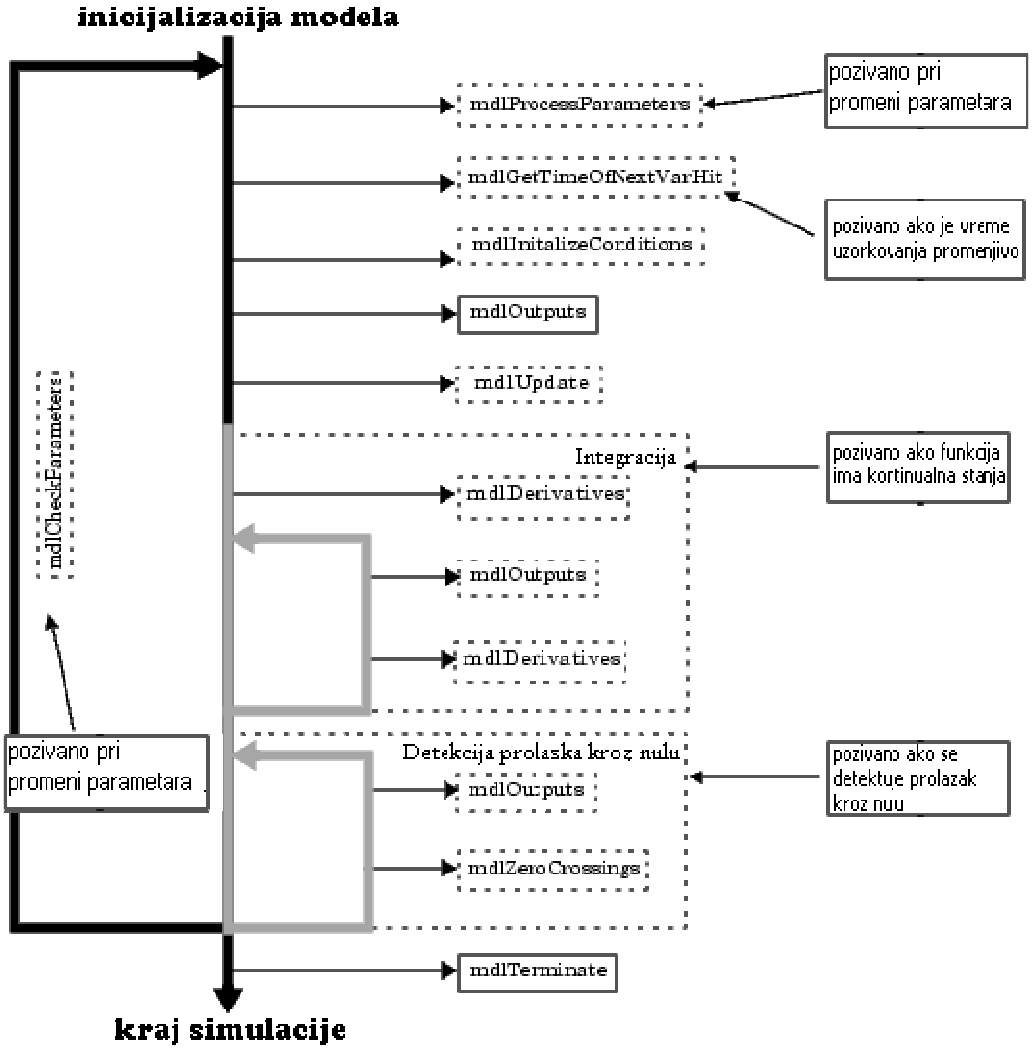
U inicijalizaciji modela (Slika 12), Simulink konfiguriše s-funkciju za predstojeću simulaciju. Simulink uvek poziva metode *mdlInitializeSizes* i *mdlInitializeSampleTime* kojima postavlja osnovne atribute S-funkcije kao što su ulazi i izlazi bloka, parametri itd. Ostale metode Simulink poziva po potrebi, zavisno od toga šta treba podesiti . Na primer ako su veličine ulaznih vektora definisani kao promenljivi u metodi *mdlInitializeSizes*, poziva se metoda *mdlSetInputPortWidth* u toku prenosa signala.



Slika 12 – Funkcije koje se pozivaju pri inicijalizaciji modela

Posle inicijalizacije Simulink startuje sledeću sumulacionu petlju (slika 13). Ako je simulacija prekinuta na bilo koji način (od strane korisnika ili usled greške), direktno se izvršava metoda *mdlTerminate*. Ako je simulacija prekunuta od strane korisnika prvo se kompletira trenutni korak.

Ako model sadrži više blokova s-funkcije, Simulink uključuje poziv svakoj metodi za svaki blok pre poziva naredne metode. Na primer prvo se pozivaju sve metode *mdlInitializeSizes* pre poziva metoda *mdlInitializeSampleTimes.* Simulink koristi sortirani red blokova za određivanje reda poziva s-funkcije. Ovaj red nije moguće podešavati ali je moguće dodeliti prioritet bloku, i tako označiti Simulinku relativni niz izvršavanja. Simulink tako uključuje podešavanja prioriteta pri pravljenju sortiranog reda blokova u skladu sa pravilima koje koristi za sortiranje.

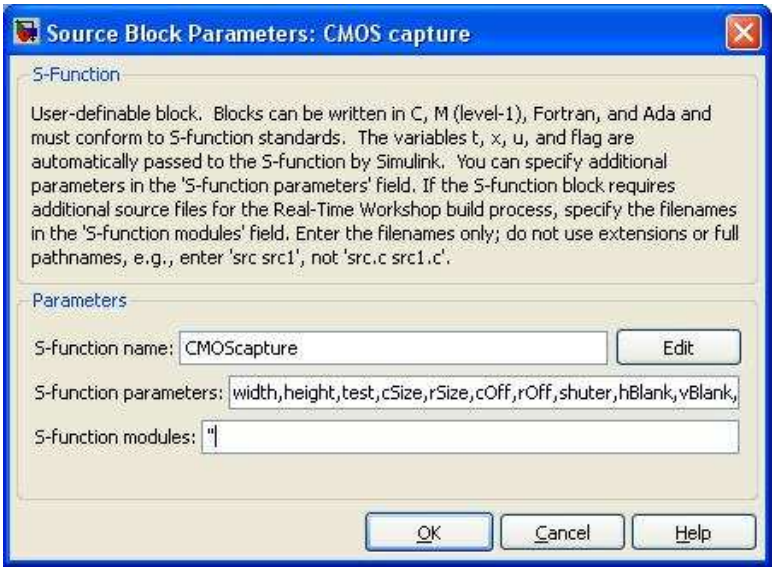


Slika 13 – Funkcije koje se pozivaju u simulacionoj petlji

Simulink obezbeđuje set funkcija za pristup poljima simulacione strukture s-funkcije (SimStruct). Metode s-funkcije koriste ove funkcije za smeštanje i preuzimanje informacija o s-funkciji.

3.3 Simulink model blok

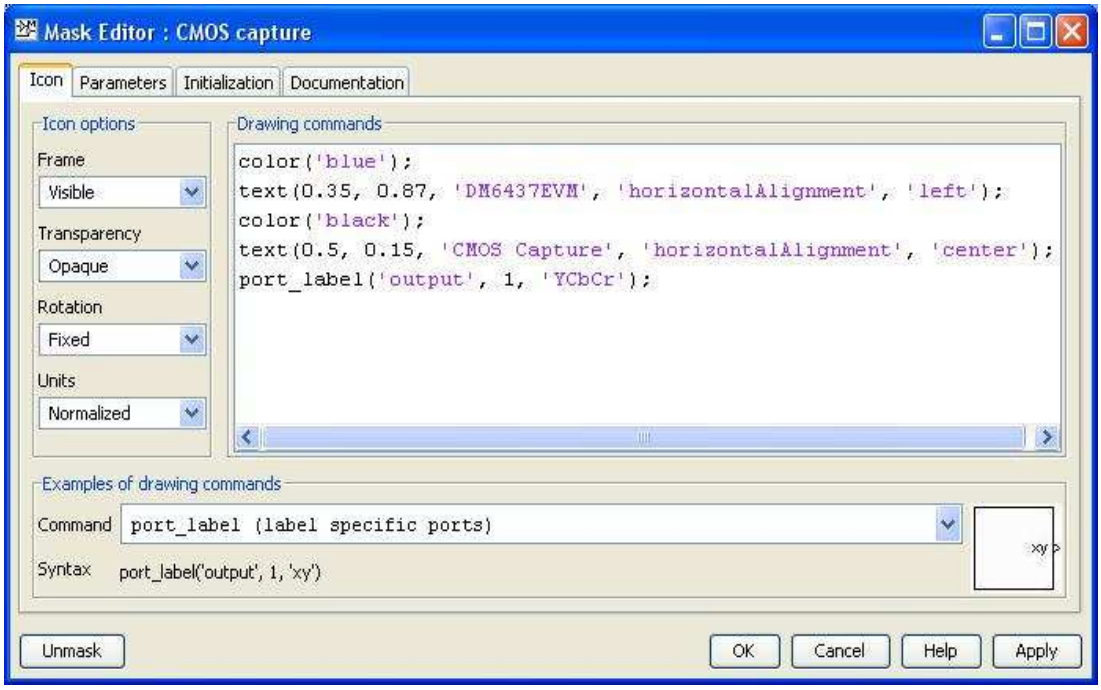
Korisnički Simulink grafički model blok se izrađuje uz pomoć generičkog bloka "S-Function" kome se prosleđuje naziv s-funkcije, naziv parametara i po potrebi dodatne datoteke (Slika 14) u kojima se nalazi kod potreban za prevođenje. "S-Function" blok iz funkcije očitava broj ulaza i izlaza i grafički ih prikazuje, a u slučaju greške obaveštava korisnika.



Slika 14 – Parametri generičkog bloka "S-Function"

Maska bloka se podešava preko korisničkog grafičkog okruženja (Slika 15). Postoje četiri grupe podešavanja:

- U prvoj grupi podešava izgled ikone kao što su boja, nazivi ulaza i izlaza...
- U drugoj grupi se unose nazivi parametara sa načinom unosa i tipom vrednosti, kao i opisom parametra. Po potrebi može da se definiše i funkcija koja će biti pozvana pri svakom korišćenju parametra, koja može da proveri sam parametar ili postavi neki drugi parametar modela.
- Treća grupa služi za inicijalizaciju parametara, koja se izvršava pri startovanju simulacije modela.
- Četvrta za naslov maske, opis bloka i uputstva za korišćenje.



Slika 15 – Grafičko okruženje za podešavanje maske bloka

3.4 TLC skripta

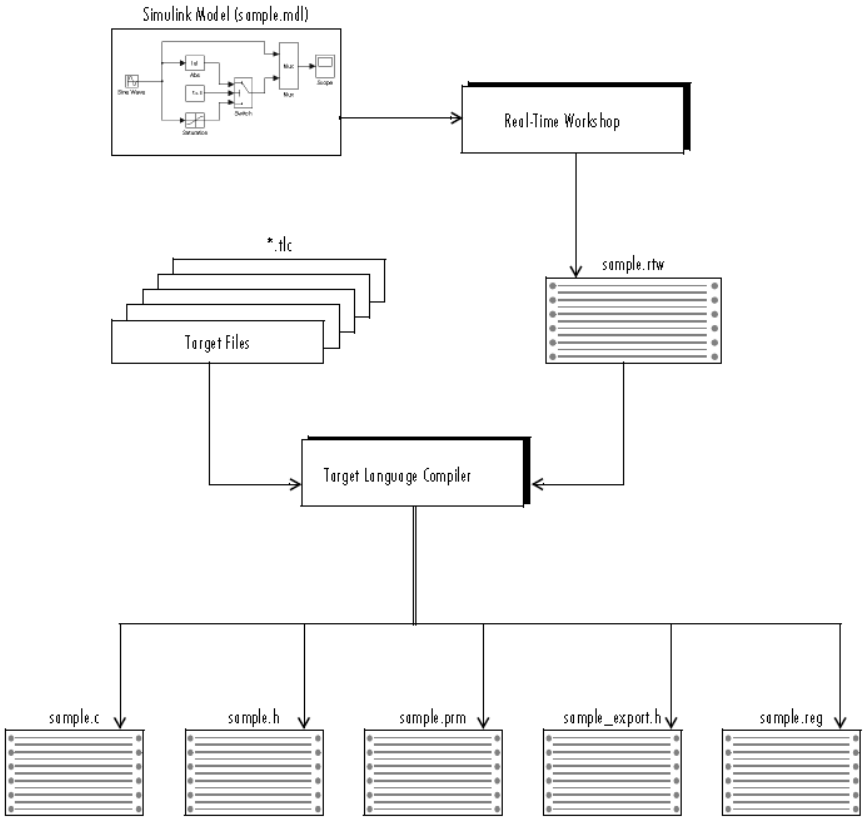
TLC (eng.Target Language Compiler) je alat uključen u RTW koji omogućava primenu i obradu C ili Ada koda generisanog iz bilo kog Simulink modela. Upotrebom TLC-a kod se može obraditi za specifičnu platformu, zbog performansi, veličine koda ili postojećih metoda potrebnih za upotrebu. RTW prevodi model u rtw datoteku, koja predstavlja tekstualni opis modela (Slika 12). TLC prevodilac generiše kod baziran na TLC skriptama, koje postoje za svaki blok.

TLC skripte su tekst datoteke koje sadrže određene komande i delove koda. Postoje dve vrste tipova TLC datoteka:

- model TLC
- blok TLC

Blok TLC služi za generisanje koda za specifičan blok. Za sve blokove koje sadrži Simulink, Matlab dostavlja i odgovarajući TLC kod. Za blokove koje dodaje korisnik mora se napisati dodatna blok TLC skripta.

Model TLC definiše generalnu strukturu koda za dati model. Pri generisanju poziva se za svaki blok odgovarajući TLC. I za svaku platformu postoji model TLC koji kontroliše generisanje kodova.

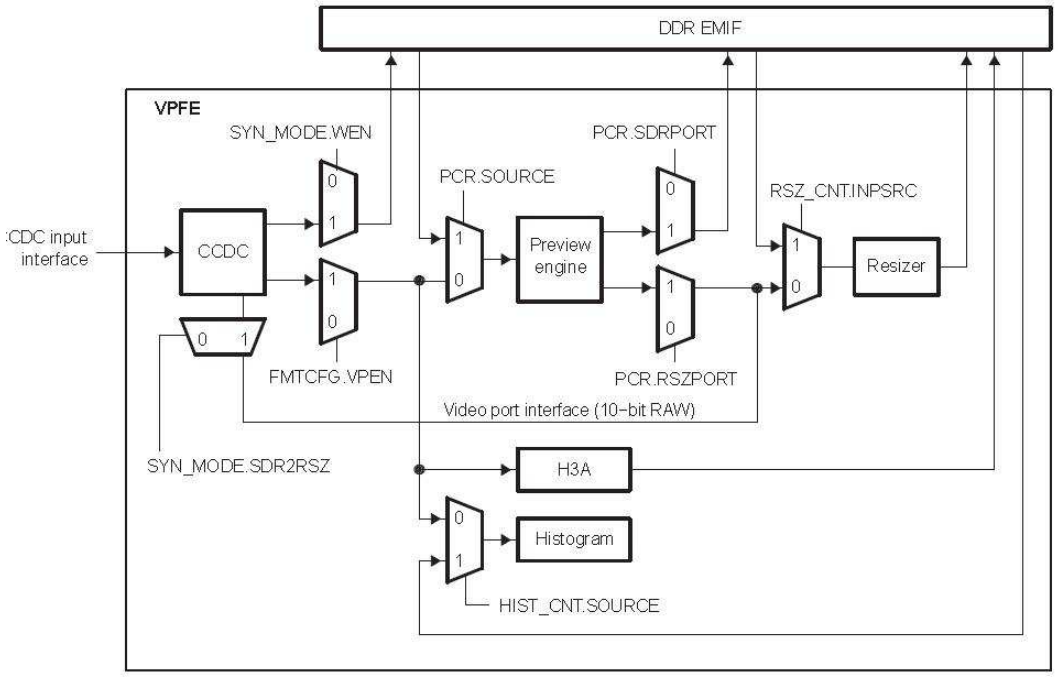


Slika 16 – Dijagram generisanja koda iz modela

Za dodavanje korisničkog bloka potrebno je generisati blok koji sadrži potrebne osobine i napisati blok TLC skriptu. Sam blok se generiše preko s-funkcije.

4 Opis rešenja

4.1 CMOS senzor blok



Slika 17 – Funkcionalni dijagram modula ulazne video obrade VPFE

CCD kontroler je odgovoran za prihvatanje neobrađene binarne slike/videa sa senzora (CMOS ili CCD) i prihvatanje YUV video podataka u raznim formatima koji obično dolaze sa video dekoderskih uređaja. U slučaju binarne slike potrebno je sliku obraditi i transformisati u potreban oblik. Ova obrada se može izvršiti u modulu za obradu ili programski. U paraleli sa obradom CCD kontroler može da izračunava statistiku slike (H3A i histogram) za eventualnu kontrolu podešavanja parametara slike.

Bitne stvari koje CCD kontroler podržava su: ulazni bajer format, generisanje HD/VD signala za sinhronizaciju, podržavanje senzora sa izvorom progresiv slika ili polu-slika, senzore sa takatom do 90MHz, ulazni YCbCr 4:2:2 format bilo 8 ili 10 bita dužine.

IPIPE modul je odgovoran za transformaciju binarnih podataka sa senzora u format YCbCr 4:2:2 koji je pogodan za kompresiju ili prikazivanje na ekran. Programiranje se vrši preko kontrolnih i konfiguracionih registara. Podržane opcije su: primanje ulaznog signala u standardnom bajer formatu, prihvatanje slike bilo sa CCD/CMOS senzora ili iz SDRAM/DDRAM memorije, izlaz slike do 1280 tačaka širine, primene horizontalnog

median filtera, programibilnog filtera šuma, digitalno pojačanje boja i balans belog, gama korekciju, programibilnu konverziju boja (RGB u YUV)…

CMOS senzor blok se sastoji od s-funkcije (u datoteci *CMOScapture.c*), pripadajuće tlc skripte (*CMOScapture.tlc*) i modela sa maskom parametre(*CMOScaptureBlock.mdl*).

4.1.1 S-funkcija bloka

U datoteci *CMOScapture.c* se nalazi s-funkcija CMOScapture. Ona se sastoji od sledećih metoda:

- *mdlCheckParameters(SimStruct *S)* metoda za proveru parametara
- *mdlInitializeSizes(SimStruct *S)* metoda sadrži definiciju za broj ulaza i izlaza.

```
if (!ssSetNumInputPorts(S, 0)) return;
if (!ssSetNumOutputPorts(S, 1)) return;
```

I definiciju parametara ulaza i izlaza. U ovom slučaju su veličine dinamičke jer veličina slike zavisi od predhodnog bloka.

```
ssSetOutputPortDataType(S, 0, SS_UINT8);
ssSetOutputPortMatrixDimensions(S,0,DYNAMICALLY_SIZED,
DYNAMICALLY_SIZED)
ssSetOutputPortComplexSignal(S, 0, COMPLEX_NO);
ssSetOutputPortOptimOpts(S, 0, SS_REUSABLE_AND_LOCAL);
ssSetOutputPortOutputExprInRTW(S, 0, 0);
```

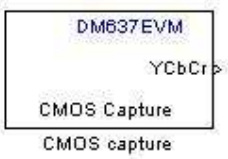
- *mdlInitializeSampleTimes(SimStruct *S)* metoda definiše vreme koraka simulacije što u ovom slučaju nije od značaja.
- *mdlSetWorkWidths(SimStruct *S)* metoda sadrži registraciju parametara.

Naredne tri metode služe za postavljanje dimenzije i tipa izlaznog signala:

- *mdlSetOutputPortDimensionInfo(SimStruct *S, int_T portIndex, const DimsInfo_T *dimsInfo)*
- *mdlSetDefaultPortDimensionInfo(SimStruct *S)*
- *mdlSetDefaultPortDataTypes(SimStruct *S)*
- *mdlOutputs(SimStruct *S, int_T tid)* metoda mora da postoji a u ovom slučaju nije od značaja.

4.1.2 Simulink blok

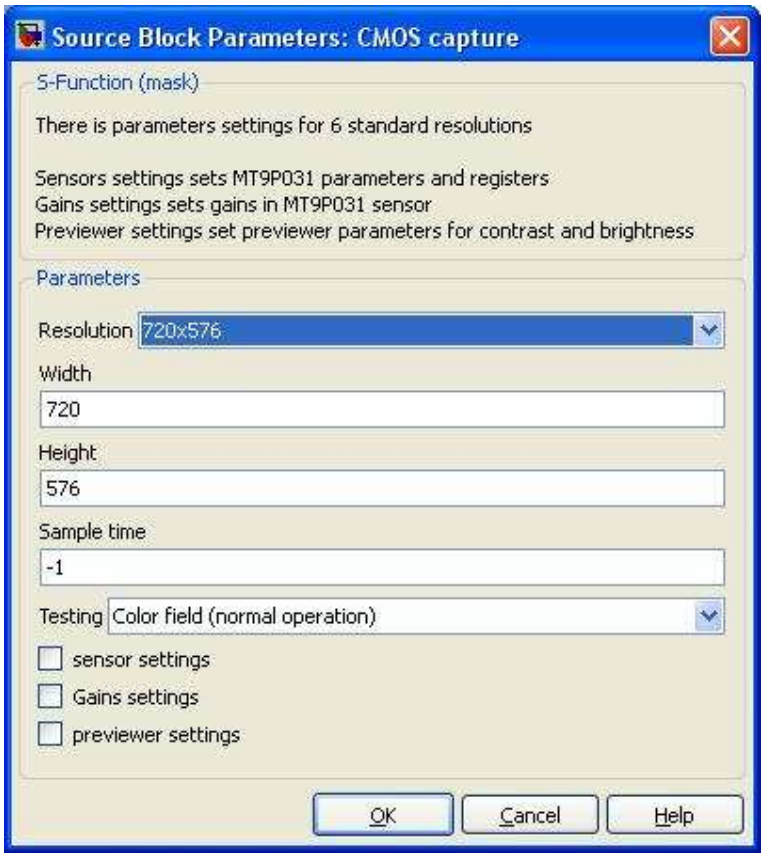
Simulink blok (Slika 18) ima jedan izlaz čije su dimenzije 2 puta rezolucija zadate slike i koji predstavlja izlaz iz IPIPE modula procesora.



Slika 18 – Izgled CMOS senzor bloka

Maska bloka predstavlja grafičku spregu za unos parametara(Slika 19). U ovom slučaju ima 32 parametara koja se mogu podešavati. Dva glavna parametra su visina i širina, odnosno rezolucija slike za koju postoje i više predefinisanih vrednosti. Ostali parametri su grupisanu u tri grupe: podešavanje senzora (sensor settings), podešavanje pojačanja (gain settings) i podešavanje pregleda (previewer settings) .

- Podešavanje senzora se obavlja podešavanjem sledećih parametara: veličine polja, pomeraja početka po horizontali i vertikali, osvetljenje, i četiri registara za modove očitavanja (column address, row address, read mode 1, read mode 2).
- Podešavanje pojačanja može da bude analogno i digitalno i obavlja se za svaku boju u bajerovom formatu posebno.
- Podešavanje pregleda uključuje podešavanje kontrasta i osvetljenja.



Slika 19 – Parametri CMOS senzor bloka

4.1.3 TLC skripta

Datoteka *CMOScapture.tlc* predstavlja TLC skriptu koja uz pomoć funkcija koje su definisane u TLC skripti modela, generiše CCS projekat i odgovarajuće datoteke.

Kodovanje se obavlja uz pomoć date programske podrške odnosno strukture za svaki modul koji se koristi. Za CMOS senzor blok je korišćen CCD kontroler i IPIPE modul.

Data je struktura kojom se definišu parametri za CCD kontroler. Komande TLC skripte počinju znakom %. Za preuzimanje parametara i drugih parametara samog bloka koriste se bibliotečke funkcije kao na primer **%<LibBlockParameterValue(height, 0)>** kojom se na dato mesto postavlja vrednost visine.

```
      %assign HEIGHT = LibBlockParameterValue(height, 0)
      %assign WIDTH = LibBlockParameterValue(width, 0)

// CCDC configuration parameters
PSP_VPFECcdcConfigParams ccdcParams =
{
    FVID_CCDC_RAW_FORMAT,          /* dataFlow     */
    FVID_FIELD_MODE,               /* ffMode       */
    %<HEIGHT>     ,                /* height       */
    %<WIDTH>,                      /* width        */
    (%<WIDTH> *2),                 /* pitch        */
    0,                             /* horzStartPix */
    0,                             /* vertStartPix */
    NULL,                               /* appCallback  */
    {
        MT9P031_Open,                   /* extVD Fxn    */
        MT9P031_Close,
        MT9P031_Control,
    },
    0,                                  /*segId        */
    {
        PSP_VPFE_BITS12,                /* dataSize     */
        PSP_VPFE_PACK8_16BITS_PIXEL,    /* pack8        */
        PSP_VPFE_DataPol_Normal,        /* dataPol      */
        PSP_VPFE_SyncPol_Positive,      /* VDSyncPol    */
        PSP_VPFE_SyncPol_Positive,      /* HDSyncPol    */
        PSP_VPFE_SyncDir_Input,         /* HDVDMaster   */
        50,                             /* HDSyncWidth  */
        4,                              /* VDSyncWidth  */
        800,                            /* numPxlPerLine*/
        1000,                           /* numLinPerFld */
        PSP_VPFE_ALaw_Disable,          /* ALawEnable   */
        PSP_VPFE_ALaw_bits15_6,         /* ALaw_Width   */
    }

};
```

U parametrima za CCD kontroler nalaze se i funkcije *MT0P9031_Open*, *MT0P9031_Close*, *MT0P9031_Control* koje služe za inicijalizaciju, deinicijalizaciju i kontrolu senzora. Ove funkcije su sadržane u datotekama *MT9P031_extImageSensor.c* i *MT9P031_extImageSensor.h* koje su uključene u projekat.

U narednom kodu se nalazi struktura za podešavanje parametara senzora. Svi parametri se preuzimaju iz bloka odgovarajućim funkcijama

```
// MT0P031 configuration parameters
MT9P031_FormatParams MT9P031Params={
      0x8000, //pixel clock control
    48,// M
    5,//  N
    4,// P1
      %<LibBlockParameterValue(rOff, 0)> ,//row start
      %<LibBlockParameterValue(rSize, 0)>,//row size
      %<LibBlockParameterValue(rOff, 0)> ,//col start
      %<LibBlockParameterValue(cSize, 0)>,//col size
      %assign mode = LibBlockParameterValue(test, 0)
      %switch (mode)
        %case 1
                  0,
            %break
        %default
                  %<mode> * 8 +1,
            %break
    %endswitch
      %<LibBlockParameterValue(rm1, 0)>,//read mode 1
      %<LibBlockParameterValue(rm2, 0)>, //read mode 2
      %<LibBlockParameterValue(hBlank, 0)>,//horizontal blank
      %<LibBlockParameterValue(vBlank, 0)>,//vertikal blank
      %<LibBlockParameterValue(shuter, 0)>,//shutter width
      %<LibBlockParameterValue(radd, 0)>,//row adrress mode
      %<LibBlockParameterValue(radd, 0)>,//col address mode
      0,//test green pattern
      0,//test red pattern
      0//test blue pattern
};
```

Parametri za podešavanje IPIPE modula se nalaze u datoteci *previewer.c*. Tu se nalaze parametri slike kao i grupe koeficijenata za balans belog, RGB u RGB (eng. RGB blending), CFA (eng. Color Filtar Array) i RGB u YUV.

Same komande za kreiranje logičkog kanal i podešavanje slike se prosleđuju preko FVID funkcija na sledeći način.

Prvo kreiranje IPIPE logičkog kanala i prosleđivanje parametara:

```
PrevHandle = FVID_create("/previewer", IOM_INOUT, NULL, &prevChannel,NULL);
FVID_control(PrevHandle,PSP_PREVIEWER_IOCTL_SET_PARAMS, &PrevConfigParams)
```

Kreiranje logičkog kanala CCD kontrolera i prosleđivanje parametara:

```
ccdcHandle = FVID_create("/VPFE0",IOM_INOUT, NULL, &vpfeParams,&gioAttrs);
FVID_control(ccdcHandle,VPFE_ExtVD_BASE+PSP_VPSS_EXT_VIDEO_DECODER_CONFIG,&
MT9P031Params);
```

Organizacija prijema slike je takva da postoji ulazni kružni bafer sa minimalno 3 polja za sliku. CCD kontroler smešta dolazne slike u bafer brzinom kojom pristižu dok se pri obradi uzima najstarija slika iz bafera čime su obrada i prijem slike vremenski nezavisni. Dodavanje alocirane memoriju u red bafera se vrši komandom

```
FVID_queue(ccdcHandle, ccdcFrm[i]);
```

Preuzimanje bafera sa slikom se vrši sa

```
FVID_dequeue(ccdcHandle, &capFrm);
```

Dok se automatski dodavanje praznog i preuzimanje punog bafer obavlja sa

```
FVID_exchange(ccdcHandle, &capFrm);
```

U generisanom kodu u procesnoj petlji je smešten naredni deo koda koji samo prvi put preuzme pun bafer a svaki sledeće zameni prazan sa punim. Koji zatim dodeljuje pokazivač na podatke slike promenljivoj sa nazivom izlaznog signala tako da bi sledeći blok u nizu mogao da preuzme taj podatak.

```
       if (firstFrame) {
        /* Transfer ownership of one frame from driver to application */
           FVID_dequeue(ccdcHandle, &capFrm);
           firstFrame = FALSE;
       }
       else {
           FVID_exchange(ccdcHandle, &capFrm);
       }
     %assign outSignalName = FEVAL("getSignalName",
LibBlockOutputSignal(0, "", "", 0))
     %<outSignalName> = (uint8_T *) capFrm->frame.frameBufferPtr;
```

4.2 Skaler slike (Resizer)

Skaler slike je deo VPFE modula koji je fizička implementacija polifaznog filtera za operaciju skaliranja. Skaler slike može da operiše na YCbCr 4:2:2 formatu slike ili formatu sa odvojenim poljem boja na primer RGB 8:8:8. Skaler koristi sledeće algoritme za skaliranje:

- Bi-linearnu interpolaciju
- Bi-kubik interpolaciju
- Nikvistov niskopojasni filtar

Za povećanje slike može da se koristi bilo koji od ova tri metoda, a za smanjenje se prvo koristi niskopojasni filtar da bi se izbegao *aliasing* efekat. Od 1/4X do X, skaler koristi četvorofazni polifazni filtar, a za X do 4X skaliranje koristi osmofazni polifazni filtar. Izbor filtra se vrši automatski na bazi faktora skaliranja Skaler radi u 2 etape, horizontalnoj i vertikalnoj. Ulaz može da bude sa CCD kontrolera ili iz memorije.

4.2.1 S-funkcija bloka

U datoteci *resizer.c* je definisana s-funkcija sa imenom „resizer“. Ona se sastoji od sledećih metoda:

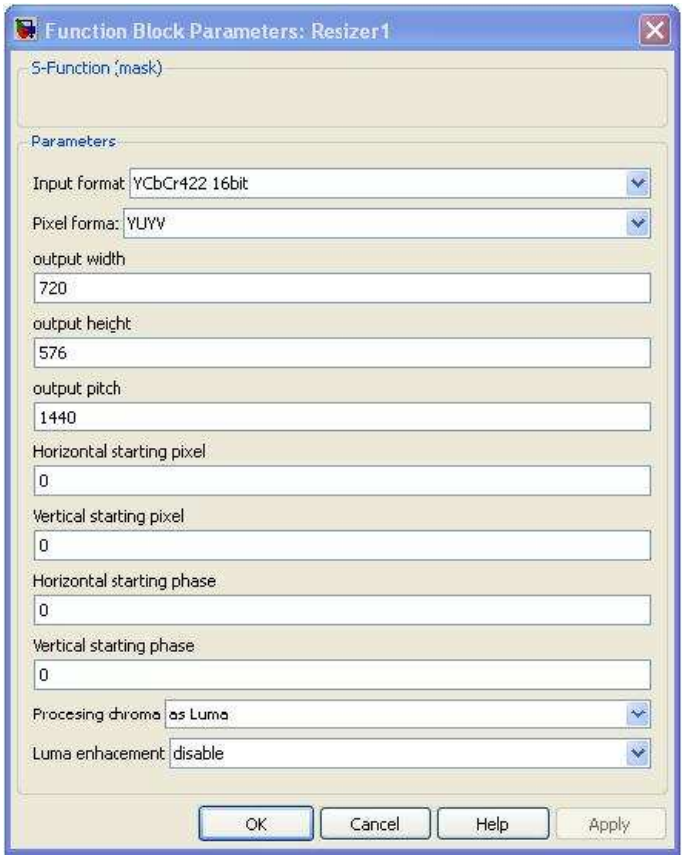
- *mdlCheckParameters(SimStruct *S)*
- *mdlInitializeSizes(SimStruct *S)*
- *mdlInitializeSampleTimes(SimStruct *S)*
- *mdlSetWorkWidths(SimStruct *S)*
- *mdlSetInputPortDimensionInfo(SimStruct *S, int_T portIndex, const DimsInfo_T *dimsInfo)*
- *mdlSetOutputPortDimensionInfo(SimStruct *S, int_T portIndex, const DimsInfo_T *dimsInfo)*
- *mdlSetDefaultPortDimensionInfo(SimStruct *S)*
- *mdlSetInputPortDataType(SimStruct *S, int port, DTypeId dataType)*
- *mdlSetOutputPortDataType(SimStruct *S, int port, DTypeId dataType)*
- *mdlSetDefaultPortDataTypes(SimStruct *S)*
- *mdlOutputs(SimStruct *S, int_T tid)*

4.2.2 Simulink model

Model skalera slike (Slika 20) može da se koristi u sprezi sa bilo kojim izvorom slike u odgovarajućem formatu. Preko ulazne sprege preuzima rezoluciju ulazne slike a u parametrima se postavlja rezolucija izlazne slike.

Slika 20 – Izgled bloka za skaliranje slike

Pored rezolucije u parametrima skalera slike (Slika 21) može da se postavi i deo slike koji će biti skaliran, kao i načini skaliranja.



Slika 21 – Parametri bloka za skaliranje slike

4.2.3 TLC skripta

Struktura za podešavanje parametara skalera slike sve vrednosti preuzima iz odgovarajućeg bloka:

```
      /* Params structure for configuration of parameters for channel 1 */
PSP_RSZparams params =
{
    %<LibBlockInputSignalDimensions(0)[0]> /2, /* Input image width */
    %<LibBlockInputSignalDimensions(0)[1]>,    /* Input image height */
    %<LibBlockInputSignalDimensions(0)[0]>,    /* Input image pitch */

      %switch(LibBlockParameterValue(imgType, 0))
      %case 1
```

```
            PSP_RSZ_INTYPE_YCBCR422_16BIT,  /* Input image type */
      %break
      %case 2
            PSP_RSZ_INTYPE_PLANAR_8BIT,     /* Input image type */
      %break
      %endswitch


    %<LibBlockParameterValue(vPix, 0)>,    /* Vertical starting pixel */
    %<LibBlockParameterValue(hPix, 0)>,    /* Horizonatl starting pixel */

      %if (LibBlockParameterValue(LumaProc, 0) == 1)
      PSP_RSZ_CBILIN_DISABLE,             /* Chroma position */
      %else
    PSP_RSZ_CBILIN_ENABLE,                /* Chroma position */
      %endif

      %switch(LibBlockParameterValue(pixFormat, 0))
      %case 1
            PSP_RSZ_PIX_FMT_UYVY,        /* Image format */
      %break
      %case 2
            PSP_RSZ_PIX_FMT_YUYV,        /* Image format */
      %break
      %case 3
            PSP_RSZ_PIX_FMT_PLANAR,      /* Image format */
      %break
      %endswitch

    %<LibBlockParameterValue(ow, 0)>,   /* Intermediate image width */
    %<LibBlockParameterValue(oh, 0)>,   /* Intermediate image height */
    %<LibBlockParameterValue(oPitch, 0)>,/* Intermediate image pitch */
    %<LibBlockParameterValue(hStart, 0)>, /* Horizontal starting phase */
    %<LibBlockParameterValue(vStart, 0)>, /* Vertical starting phase */
    gRDRV_reszFilter4TapHighQuality, /* Horizontal filter coefficients */
    gRDRV_reszFilter4TapHighQuality, /* Vertical filter coefficients */

      %switch(LibBlockParameterValue(lumEnchance, 0))
      %case 1
            PSP_RSZ_YENH_DISABLE          /* Luma enhancement options */
      %break
      %case 2
            PSP_RSZ_YENH_3TAP_HPF         /* Luma enhancement options */
      %break
      %case 3
            PSP_RSZ_YENH_5TAP_HPF         /* Luma enhancement options */
      %break
      %endswitch
};
```

U inicijalizaciji se kreira logički kanal i prosleđuju se parametri:

```
rszfd =FVID_create("/resizer", IOM_INOUT, NULL,(void *)&memSegId
,&gioAttrs);
FVID_control(rszfd, PSP_RSZ_IOCTL_SET_PARAMS, &params);
```

Za skaliranje je još potrebno ukazati na ulazni i izlazni bafer i veličinu bafera a zatim izvršiti samo skaliranje. I naravno proslediti pokazivač na sliku narednom bloku:

```
resize.inBufSize=%<LibBlockInputSignalDimensions(0)[0]> *
<LibBlockInputSignalDimensions(0)[1]>*2;
resize.outBufSize =%<LibBlockParameterValue(oPitch, 0)> *
%<LibBlockParameterValue(oh, 0)>*2;
resize.inBuf = %<inSignalName>;
resize.outBuf = prevBuff->frame.frameBufferPtr;

FVID_control(rszfd, PSP_RSZ_IOCTL_RESIZE, &resize) )

%assign outSignalName=
FEVAL("getSignalName",LibBlockOutputSignal(0,"","",0))

%<outSignalName> =  prevBuff->frame.frameBufferPtr;
```

4.3 H3A blok

H3A modul je dizajniran za podršku kontrolne petlje za automatski fokus AF (eng Auto focus), automatski balans belog AWB (eng Automatic White Balance) i automatskog osvetljenja AE (eng. Auto Exposure) računanjem statistike slike ili videa. Statistika slike se koristi za podešavanje raznih parametara u procesu obrade slike.

Postoje dva glavna bloka u H3A modulu:

- Auto fokus (AF)
- Automatsko osvetljenje (AE) i automatski balans belog (AWB)

AF izvodi i filteriše podatke za crvenu zelenu i plavu boju i dobavlja bilo nagomilavanja ili maksimume vrednosti u specificiranom regionu. Specificirani region je dvodimenzionalni blok podataka koji predstavlja prozor nad kojim se vrši obrada. AF podržava veći broj prozora, čija je veličina ista i može da se podesi.

AE/AWB akumulira i proverava zasićenje vrednosti u određenom regionu slike. Broj prozora/regiona je takođe podesiv kao i visina i širina.

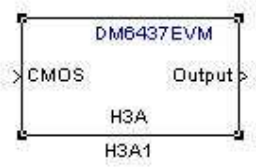
4.3.1 S-funkcija bloka

U datoteci *h3a.c* je definisana s-funkcija h3a. Ona se sastoji od sledećih metoda:

- *mdlCheckParameters(SimStruct *S)*
- *mdlInitializeSizes(SimStruct *S)*
- *mdlInitializeSampleTimes(SimStruct *S)*
- *mdlSetWorkWidths(SimStruct *S)*
- *mdlSetInputPortDimensionInfo(SimStruct *S, int_T portIndex, const DimsInfo_T *dimsInfo)*
- *mdlSetOutputPortDimensionInfo(SimStruct *S, int_T portIndex, const DimsInfo_T *dimsInfo)*
- *mdlSetDefaultPortDimensionInfo(SimStruct *S)*
- *mdlSetInputPortDataType(SimStruct *S, int port, DTypeId dataType)*
- *mdlSetDefaultPortDataTypes(SimStruct *S)*
- *mdlOutputs(SimStruct *S, int_T tid)*

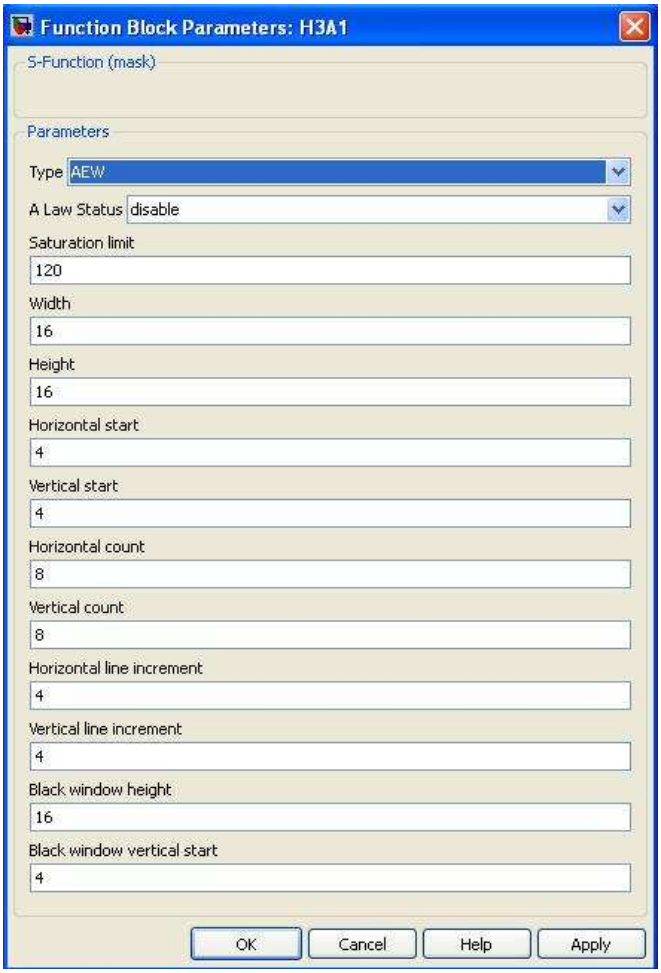
4.3.2 Model bloka

H3A blok (Slika 22) se povezuje na izlaz CMOS senzor bloka. Izlaz H3A bloka je vektor podataka čija veličina zavisi od broja prozora.



Slika 22 – Izgled H3A bloka

U parametrima bloka se bira osnovna funkcija: automatsko osvetljenje/balans belog AEW ili automatski fokus AF.



Slika 23 – Parametri H3A bloka za automatsko osvetljenje i balans belog

U parametrima za automatsko osvetljenje i balans belog (Slika 23) se definiše veličina i pozicija prozora na slici nad kojim se izračunava statistika, kao i broj prozora i pomeraj po vertikali i horizontali.

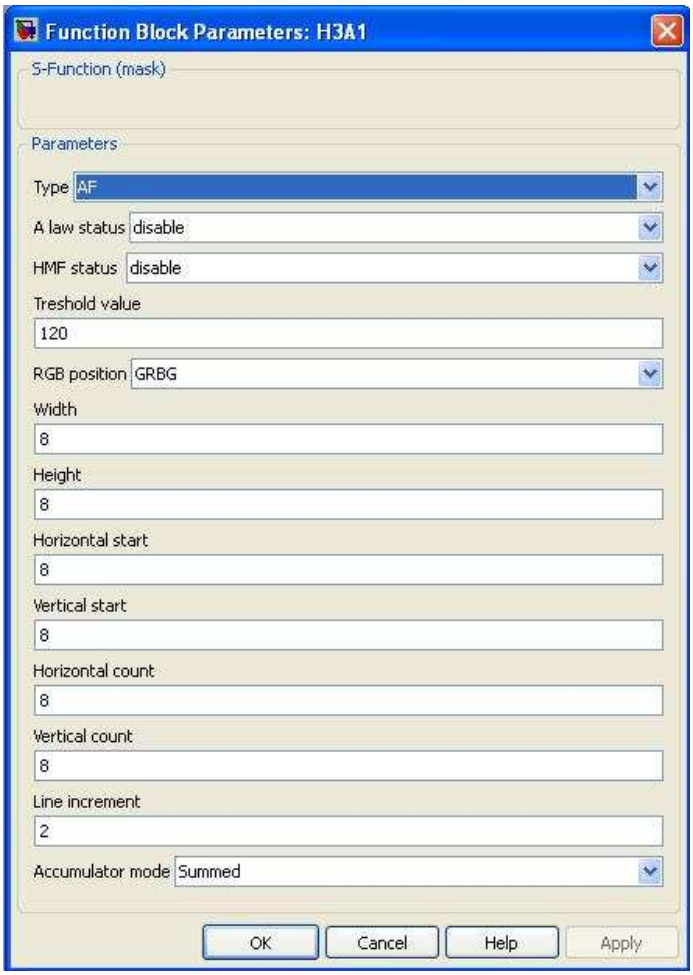
Podaci koji se dobijaju za jedan prozor su:

```
Green sub sample accumulator
Red sub sample accumulator
Blue sub sample accumulator
Green sub sample accumulator
Green saturator accumulator
Red saturator accumulator
Blue saturator accumulator
Green saturator accumulator
```

U parametrima za automatski fokus AF (Slika 24) se takođe definišu veličina i pozicija prozora na slici nad kojim se izračunava statistika, kao i broj prozora i pomeraj.

Vrednosti koje se dobijaju za jedan region su:

```
Green Sum
Peak sum from IIR filter 0 (green)
Peak sum from IIR filter 1 (green)
0
Red Sum
Peak sum from IIR filter 0 (red)
Peak sum from IIR filter 1 (red)
0
Blue Sum
Peak sum from IIR filter 0 (blue)
Peak sum from IIR filter 1 (blue)
```



Slika 24 – Parametri H3A bloka za automatski fokus

4.3.3 TLC skripta

U TLC skripti se nalaze struktura za postavljanje AF parametara:

```
PSP_AFParams H3Aparams ={
                  %if (LibBlockParameterValue(af_Alaw, 0) == 1)
                  PSP_AF_ALAW_ENABLE,                /* A Law Status */
                  %else
                  PSP_AF_ALAW_DISABLE,                /* A Law Status */
                  %endif
                  {
                        %if (LibBlockParameterValue(mEnable, 0) == 1)
                          PSP_AF_HMF_ENABLE,             /* HMF Status  */
                        %else
                          PSP_AF_HMF_DISABLE,             /* HMF Status  */
                        %endif
                        /* Threshold value */
                        %<LibBlockParameterValue(mTreshold, 0)>,
                  },
                  %switch( LibBlockParameterValue(rgbPos, 0) )
                  %case 1
                    PSP_AF_GR_GB_BAYER,
                  %break
                  %case 2
                    PSP_AF_RG_GB_BAYER,
                  %break
                  %case 3
                    PSP_AF_GR_BG_BAYER,
                  %break
                  %case 4
                    PSP_AF_RG_BG_BAYER,
                  %break
                  %case 5
                    PSP_AF_GG_RB_CUSTOM,
                  %break
                  %case 6
                    PSP_AF_RB_GG_CUSTOM,
                  %break
                  %default
                    PSP_AF_GR_BG_BAYER,                 /* RGB position */
                  %break
                  %endswitch

                  /* IIR Filter Configuration */
                  {
                  2u,                             /* IIR Horizontal start
*/
      /*IIR Filter coefficients for set 0 */
        {
            64, 0, 0, 21, 22, 21, 0, 0, -16, 32, -16,
        },
```

```
       /*IIR Filter coefficients for set 1 */
        {
            64, 0, 0, 21, 22, 21, 0, -29, -16, 32, -16,
        },
    },
    /* Paxel Configuration  */
    {
        %<LibBlockParameterValue(pWidth, 0)>,          /* width */
        %<LibBlockParameterValue(pHeight, 0)>,         /* height */
        %<LibBlockParameterValue(pHstart, 0)>,         /* hzStart */
        %<LibBlockParameterValue(pVstart, 0)>,         /* vtStart */
        %<LibBlockParameterValue(pHcount, 0)> ,        /* hzCnt */
        %<LibBlockParameterValue(pVcount, 0)> ,        /* vtCnt */
        %<LibBlockParameterValue(pLine, 0)>,           /* lineIncr */
    },
      %if (LibBlockParameterValue(mode, 0) == 1)
        PSP_AF_ACCUMULATOR_SUMMED,  /* Accumulator Mode */
      %else
        PSP_AF_ACCUMULATOR_PEAK,
      %endif

};
```

Struktura za postavljanje AEW parametara:

```
PSP_AEWParams AewParams =
      {
      %if (LibBlockParameterValue(aew_Alaw, 0) == 1)
        PSP_AEW_ALAW_ENABLE,                /* A Law Status */
      %else
        PSP_AEW_ALAW_DISABLE,               /* A Law Status */
      %endif
    %<LibBlockParameterValue(satLimit, 0)>,         /* Saturation limit */
    {
        %<LibBlockParameterValue(wWidth, 0)>,       /* width    */
        %<LibBlockParameterValue(wHeight, 0)>,      /* height   */
        %<LibBlockParameterValue(wHstart, 0)>,      /* horizontal start */
        %<LibBlockParameterValue(wVstart, 0)>,      /* vertical start   */
        %<LibBlockParameterValue(wHcount, 0)> ,     /* horizontal count */
        %<LibBlockParameterValue(wVcount, 0)> ,     /* vertical count   */
        %<LibBlockParameterValue(wHline, 0)>,/*horizontal line increament*/
        %<LibBlockParameterValue(wVline, 0)>,/*vertical line increament  */
    },
    {
      %<LibBlockParameterValue(bwHeight, 0)>,/*black window height     */
      %<LibBlockParameterValue(bwStart, 0)>,/*black window vertical start*/
    }
};
```

Kreiranje logičkog kanala i prosleđivanje parametara se smešta u funkciju za inicijalizaciju:

```
afHandle = FVID_create("/H3A0",IOM_INOUT,&gioStatus,&typeAF,&gioAttrs);
FVID_control(afHandle, PSP_H3A_IOCTL_SET_PARAM, &H3Aparams))
```

Kod H3A bloka se prosleđivanje praznog i preuzimanje punog bafera izvršava na drugačiji način.Prvo se podesi odgovarajuću bafer :

```
      /* Enqueue the buffer, This will enable AEW engine */
      enbufferAF.size = H3A_AF_BUFFER_SIZE;
      enbufferAF.ramIpAddr = (Ptr)(arrayh3AAF);
      bufferSize = sizeof(PSP_H3ABuffer);

FVID_submit(afHandle, PSP_VPSS_QUEUE,&enbufferAF,&bufferSize, NULL);
```

Kreiranje i podešavanje parametara je standardno kao za ostale module sistema:

```
aewHandle = FVID_create("/H3A0",IOM_INOUT,&gioStatus,&typeAew,&gioAttrs);
FVID_control(aewHandle, PSP_H3A_IOCTL_SET_PARAM, &AewParams);
```

U delu za obradu slike se postavljaju naredbe za dodeljivanje praznog i preuzimanje punog bafera, kao i prosleđivanje pokazivača na bufer:

```
FVID_submit(afHandle, PSP_VPSS_DEQUEUE,&debufferAF,&bufferSize, NULL);
FVID_submit(afHandle, PSP_VPSS_QUEUE,&enbufferAF,&bufferSize, NULL);

%assign outSignalName=
FEVAL("getSignalName",LibBlockOutputSignal(0,"","",0))
%<outSignalName> =
(%<LibBlockOutputSignalDataTypeName(0,"")>*)debufferAF.ramIpAddr;
```

4.4 Histogram blok

Histogram modul prihvata sirovu sliku ili video (3 ili 4 boje) i slaže grupe od tačaka na bazi vrednosti (za svaku boju posebno). Svaka grupa sadrži broj tačaka čija vrednost upada u odgovarajući opseg.

Histogram preuzima sliku sa CCD/CMOS senzora. Podržava do četiri regiona i svaki region ima svoju horizontalnu/vertikalnu početnu i krajnju poziciju. Ako se regioni preklapaju tačke se akumuliraju u region najvišeg prioriteta (najveći prioritet region0>region1>region2>region3). Broj grupa može biti 32, 64, 128 ili 256 za svaku boju po regionu.

Za broj regiona 1, dopušteno je 32, 64, 128 ili 256 grupa po boji

Za broj regiona 2, dopušteno je 32,64 ili 256 grupa po boji.

Za broj regiona 3 i 4, dopušteno je 32 ili 64 grupa po boji.

4.4.1 S-funkcija bloka

U datoteci *histogram.c* je definisana s-funkcija histogram. Ona se sastoji od sledećih metoda:

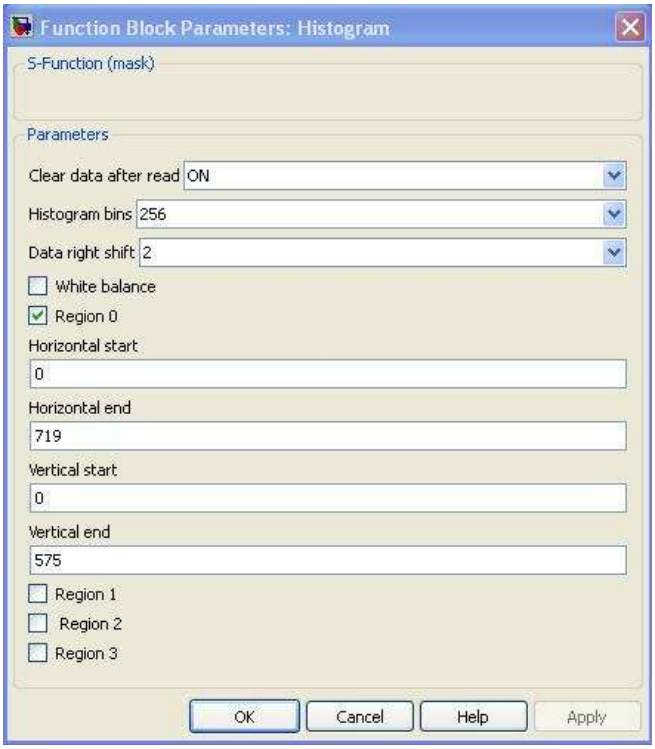
- *mdlCheckParameters(SimStruct *S)*
- *mdlInitializeSizes(SimStruct *S)*
- *mdlInitializeSampleTimes(SimStruct *S)*
- *mdlSetWorkWidths(SimStruct *S)*
- *mdlSetInputPortDimensionInfo(SimStruct *S, int_T portIndex, const DimsInfo_T *dimsInfo)*
- *mdlSetOutputPortDimensionInfo(SimStruct *S, int_T portIndex, const DimsInfo_T *dimsInfo)*
- *mdlSetDefaultPortDimensionInfo(SimStruct *S)*
- *mdlSetInputPortDataType(SimStruct *S, int port, DTypeId dataType)*
- *mdlSetOutputPortDataType(SimStruct *S, int port, DTypeId dataType)*
- *mdlSetDefaultPortDataTypes(SimStruct *S)*
- *mdlOutputs(SimStruct *S, int_T tid)*

4.4.2 Model bloka

Histogram blok (Slika 25) može da se poveže samo na CMOS senzor blok sa koga preuzima sliku za izračunavanje histograma. Izlaz je vektor podataka veličine 512 ako je broj grupa 32 inače 1024.

Slika 25 – Izgled histogram bloka

U parametrima (Slika 26) se podešava pozicija i veličina regiona i može ih biti maksimalno četiri. Pored toga može da se bira broj grupa iz padajućeg menija i skaliranje rezultata pomeranjem u desno.



Slika 26 – Parametri histogram bloka

4.4.3 TLC skripta

TLC skripta koja odgovara modulu histogram (*histogram.tlc*) sadrži strukturu kojom se sam modul podešava:

```
/* Histogram configuration parameters */
PSP_HISTparam histConfigParams =
{
      %if (LibBlockParameterValue(oClear, 0) == 1)
        PSP_HIST_CLEAR_AFTER_READ,      /* clearOpDataAfterRead */
      %else
        PSP_HIST_NO_CLEAR_AFTER_READ,      /* clearOpDataAfterRead */
      %endif

      %switch( LibBlockParameterValue(bin, 0) )
            %case 1
                  PSP_HIST_32BINS,              /* bins                 */
            %break
            %case 2
                  PSP_HIST_64BINS,              /* bins                 */
```

```
        %break
        %case 3
               PSP_HIST_128BINS,               /* bins                */
        %break
        %case 4
               PSP_HIST_256BINS,               /* bins                */
        %break
        %endswitch

        %switch( LibBlockParameterValue(hShift, 0) )
        %case 1
               PSP_HIST_DATA_RSHIFT_ZERO,      /* shift               */
        %break
        %case 2
               PSP_HIST_DATA_RSHIFT_ONE,       /* shift               */
        %break
        %case 3
               PSP_HIST_DATA_RSHIFT_TWO,       /* shift               */
        %break
        %case 4
               PSP_HIST_DATA_RSHIFT_THREE,     /* shift               */
        %break
        %case 5
               PSP_HIST_DATA_RSHIFT_FOUR,      /* shift               */
        %break
        %case 6
               PSP_HIST_DATA_RSHIFT_FIVE,      /* shift               */
        %break
        %case 7
               PSP_HIST_DATA_RSHIFT_SIX,       /* shift               */
        %break
        %case 8
               PSP_HIST_DATA_RSHIFT_SEVEN,     /* shift               */
        %break
        %default
               PSP_HIST_DATA_RSHIFT_TWO,       /* shift               */
        %break
        %endswitch

{
    %<LibBlockParameterValue(wbGain1, 0)>,  /* wbGain[0]            */
    %<LibBlockParameterValue(wbGain2, 0)>,  /* wbGain[1]            */
    %<LibBlockParameterValue(wbGain3, 0)>,  /* wbGain[2]            */
    %<LibBlockParameterValue(wbGain4, 0)>   /* wbGain[3]            */
},
{
    {
        %<LibBlockParameterValue(reg0_hs, 0)>,/* region[0].hStart     */
        %<LibBlockParameterValue(reg0_he, 0)>,/* region[0].hEnd       */
        %<LibBlockParameterValue(reg0_vs, 0)>,/* region[0].vStart     */
        %<LibBlockParameterValue(reg0_ve, 0)> /* region[0].vEnd       */
    },
    {
        %<LibBlockParameterValue(reg1_hs, 0)>,/* region[1].hStart     */
        %<LibBlockParameterValue(reg1_he, 0)>,/* region[1].hEnd       */
        %<LibBlockParameterValue(reg1_vs, 0)>,/* region[1].vStart     */
        %<LibBlockParameterValue(reg1_ve, 0)> /* region[1].vEnd       */
    },
```

```
        {
            %<LibBlockParameterValue(reg2_hs, 0)>,/* region[2].hStart     */
            %<LibBlockParameterValue(reg2_he, 0)>,/* region[2].hEnd       */
            %<LibBlockParameterValue(reg2_vs, 0)>,/* region[2].vStart     */
            %<LibBlockParameterValue(reg2_ve, 0)> /* region[2].vEnd       */
        },
        {
            %<LibBlockParameterValue(reg3_hs, 0)>,/* region[3].hStart     */
            %<LibBlockParameterValue(reg3_he, 0)>,/* region[3].hEnd       */
            %<LibBlockParameterValue(reg3_vs, 0)>,/* region[3].vStart     */
            %<LibBlockParameterValue(reg3_ve, 0)> /* region[3].vEnd       */
        }
    }
};
```

Kreiranje logičkog kanala i prosleđivanje parametara je standardno:

```
histHandle= FVID_create("/histogram",IOM_INPUT, &gioStatus,
hEdma,&gioAttrs);
FVID_control(histHandle, PSP_HIST_IOCTL_SET_PARAMS,&histConfigParams)
```

Povezivanje praznog bafera u red se vrši kao i za H3A blok:

```
FVID_submit(histHandle, PSP_VPSS_QUEUE,(Ptr)(&inbuff[par]),
(Ptr)(&sizeOfbuff), NULL)
```

Pri obradi video slike razmena praznog i punog bafera i prosleđivanje pokazivača sledećem bloku je data u narednom delu koda:

```
FVID_submit(histHandle, PSP_VPSS_DEQUEUE,(Ptr)(&outbuff),
(Ptr)(&sizeOfbuff), NULL) )

%assign outSignalName = FEVAL("getSignalName", LibBlockOutputSignal(0, "",
"", 0))
%<outSignalName> = (%<LibBlockOutputSignalDataTypeName(0,"")> *)
   outbuff->ramIpAddr;
```

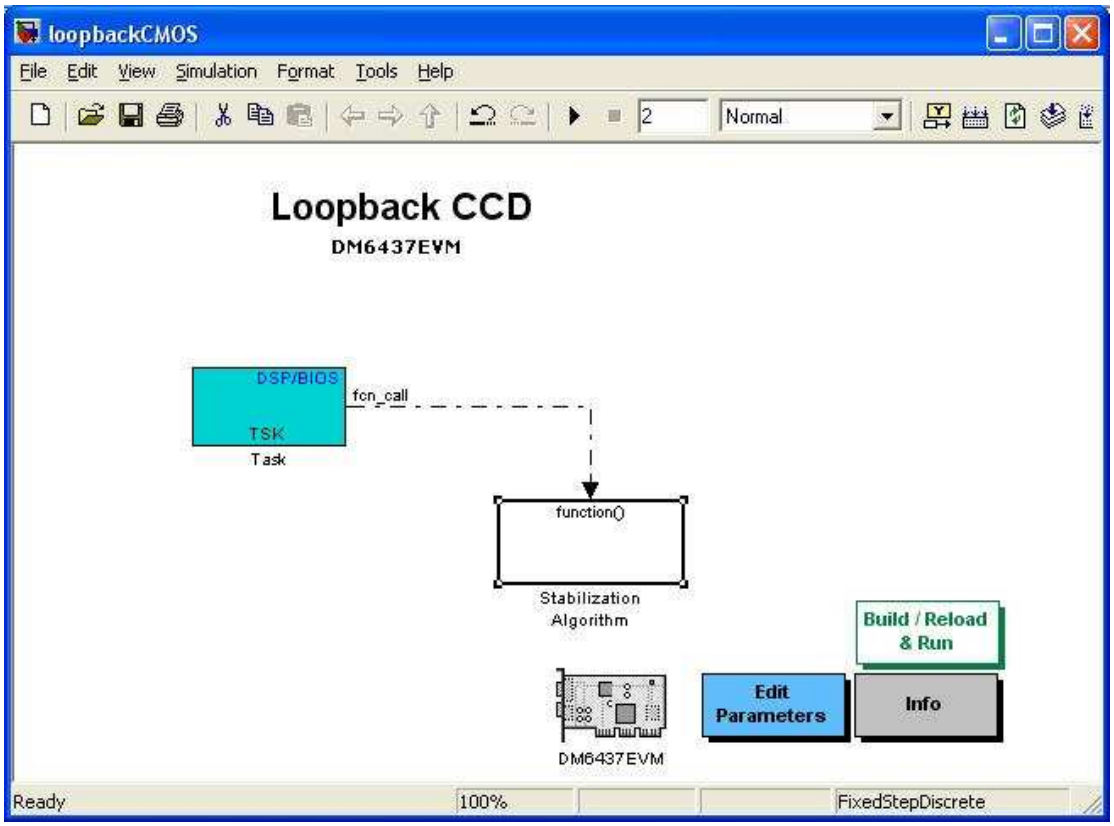
5 Ispitivanje

Ispitivanje je obavljeno startovanjem simulink modela koji sadrže date blokove i proveravanjem ispravnosti rada generisanog koda.

Svaki model mora da sadrži:

- DM6437EVM blok u kome su podešavanja za konfiguraciju ploče.
- DSP/BIOS blok koji definiše naziv niti, prioritet, memoriju koju će koristiti i veličinu magacinske memorije.
- Blok funkcije, koji sadrži korisnički model koji će biti izvršavan u niti definisanoj u DSP/BIOS bloku.

Kod ispitivanja H3A i histogram bloka korišćen je pomoćni blok „To File“ koji ulazni vektor upisuje u zadatu datoteku.

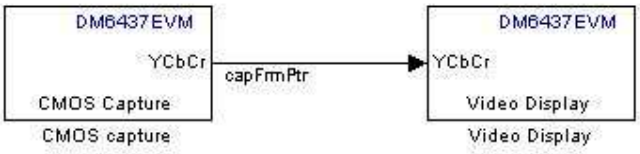


Slika 27 – Izgled osnovnog modela

5.1 Ispitivanje CMOS senzor bloka

Ispitivanje ovog bloka je je prvo urađeno modelom “s kraja na kraj” (eng.loopback), gde se slika sa CMOS senzora šalje direktno na izlaz. Izlaz je u kompozit PAL formatu i povezuje se na bilo koji uređaj odnosno ekran koji ima kompozitni ulaz. U ovom slučaju DM6437 EVM ploča radi kao obična kamera i prikazuje sliku koju prima senzor, na ekran.

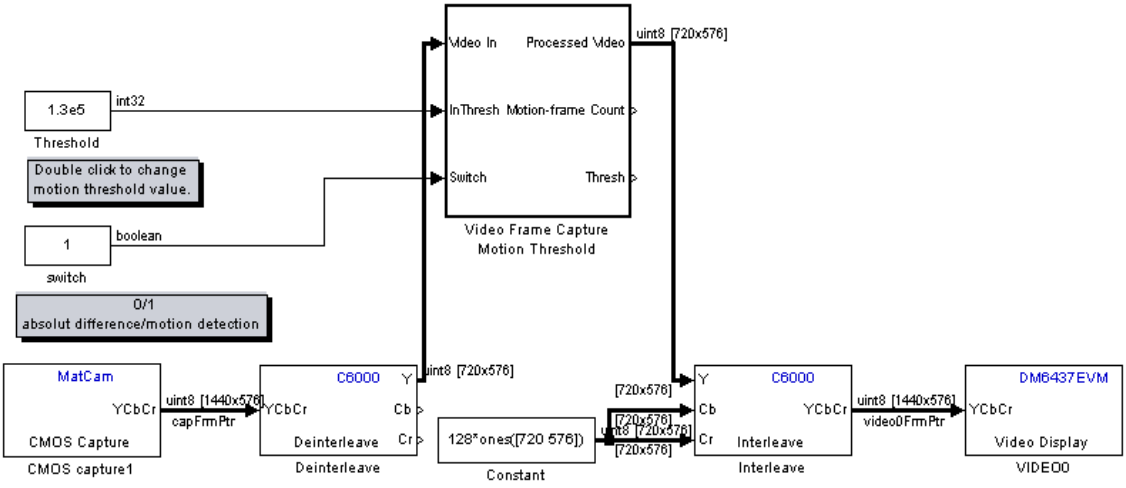
Generisanje koda je ispitivano sa raznim podešavanjem parametara CMOS senzor bloka i slika je bila promenjena u skladu sa promenjenim parametrima.



Slika 28 – Osnovni primer upotrebe CMOS senzor bloka

Slika 29 – Prikaz slike sa senzora na ekran

Model koji vrši obradu sa slikom kojim je blok testiran je prikazan na slici 30. Ovde se slika sa CMOS senzor bloka prvo odgovarajućim blokom razdvaja u komponente Y Cb Cr. Nad Y komponentom se vrši obrada, odnosno detektuje se pomeraj na slici. Slika sa obrađenom detekcijom pokreta se ponovo prepliće sa komponentama boje i šalje na izlazni blok.



Slika 30 –Primer modela za detekciju pokreta sa CMOS senzor blokom

Slika 31 –Prikaz slike sa detekcijom pokreta

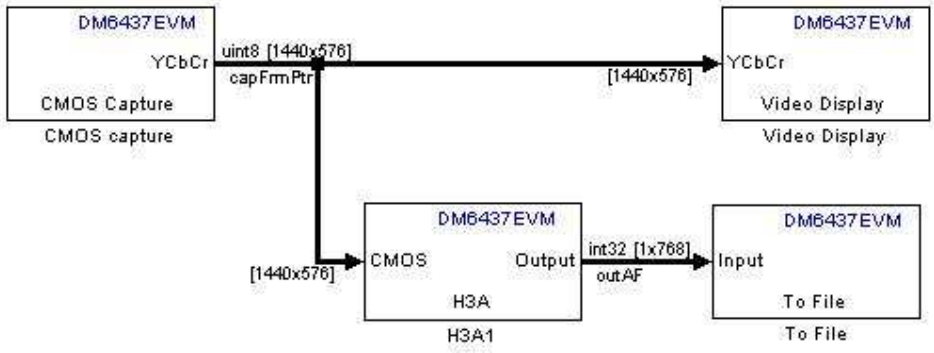
5.2 Ispitivanje skalera slike

Ispitivanje je odrađeno u modelu u kome blok prima sliku sa CMOS senzor bloka, skalira sliku i šalje na izlaz. Skaliranje je testirano povećavanjem i smanjivanjem slike.

Slika 32 –Primer modela sa upotrebom bloka za skaliranje slike

5.3 Ispitivanje H3A bloka

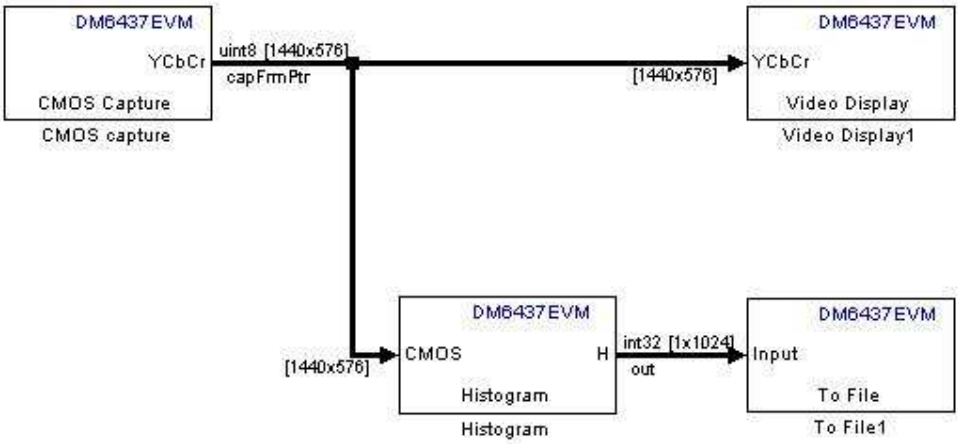
Modul H3A odrađuje statistiku slike koja može da se koristi za regulaciju osvetljenja i balansa belog ili automatski fokus. U ovom slučaju blok može da se poveže samo na CMOS senzor blok pošto se statistika izračunava samo na slici u bajer formatu. Izlazne vrednosti su zarad provere povezane na "To File" pomoćni blok koji generiše kod kojim se podaci upisuju u zadatu datoteku.



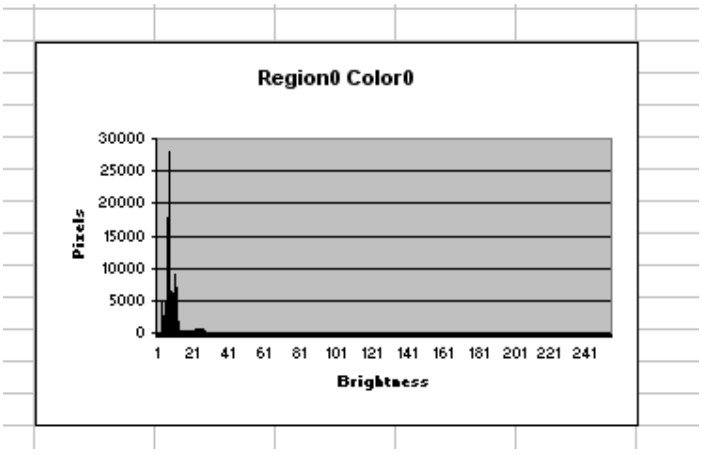
Slika 33 –Primer modela sa upotrebom H3A bloka

5.4 Ispitivanje histogram bloka

Histogram modul takođe izračunava tražene podatke sa slike bajer formata tako da je i u testnom modelu povezan na CMOS senzor blok. Izlaz se upisuje u zadatu datoteku pomoću "To File" bloka. Izlazni podaci se mogu kopirati u dati šablon za program "excel" i videti histogram dijagram.



Slika 34 – Primer modela sa histogram blokom



Slika 35 – Histogram jednog regiona

6 Zaključak

U ovom radu je opisano jedno rešenje automatskog generisanja koda na osnovu Simulink modela na ploči DM6437EVM sa povezanim CMOS senzorom MT9P031. Ovakvim generisanjem koda korisniku je prepušteno da se posveti samoj obradi ne obraćajući pažnju na nivo ispod, odnosno samu implementaciju. Dovoljno je da poznaje radu u Simulink okruženju.

Prednosti ovakvog „programiranja“ su:

- Doslednost i ekvivalentnost sepecificiranog modela i implementacije.
- Visok i konzistentan kvalitet generisanog koda uzevši u obzir greške i čitljivost.
- Efikasnija faza implementacije
- Brža izrada prototipova projekta

Uz moćnu platformu kakva je DM6437EVM čak i mnoge zahtevne obrade će raditi u realnom vremenu što povećava upotrebljivost ovog rešenja za ispitivanje i razvoj programske podrške za obradu slike.

Dalji razvoj je moguć dodavanjem dodatnih blokova koje ploča podržava kao što je modul za audio obradu ili serijske sprege čime bi se pokrila kompletna audio-video obrada i kreiranjem biblioteke. A izradom namenskih blokova za specifičnu obradu i mogućnost dodavanja korisničkih blokova na osnovu gotovog koda bi se proširila mogućnost upotrebe.

7 Literatura

[1] TMS320DM6437 EVM Technical Reference, March 2007

[2] TMS320DM6437 Digital Media Processor (Rev. D)

[2] TMS320DM643x DMP Video Processing Front End (VPFE) User's Guide (Rev. A)

[3] Understanding the Davinci Preview Engine (Rev. A)

[3] www.ti.com

[4] http://focus.ti.com/docs/prod/folders/print/tms320dm6437.html

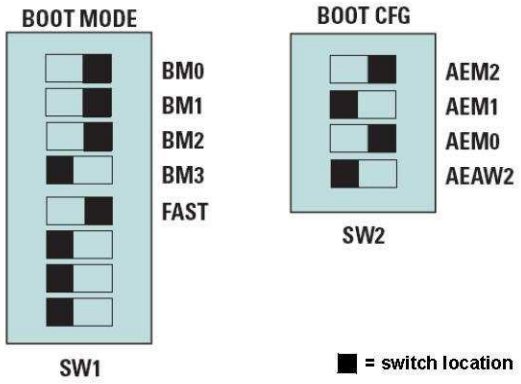
[5] http://www.mathworks.com/

[6] http://www.mathworks.com/access/helpdesk/help/helpdesk.html

8 Dodatak

8.1 Podizanje sistema iz NAND memorije

Da bi se ugrađeni program za podizanje sistema-*boot loader* podesio za podizanje sistema iz NAND memorije potrebno je *bootmode* (BM[3:0]) podešavanja u BOOTCFG registru podesiti na 0111. Što se na DM6437EVM razvojnoj ploči podešava uz pomoć prekidača SW1 kao na slici 36.



Slika 36 – Raspored prekidača na ploči za podizanje sistema iz NAND memorije

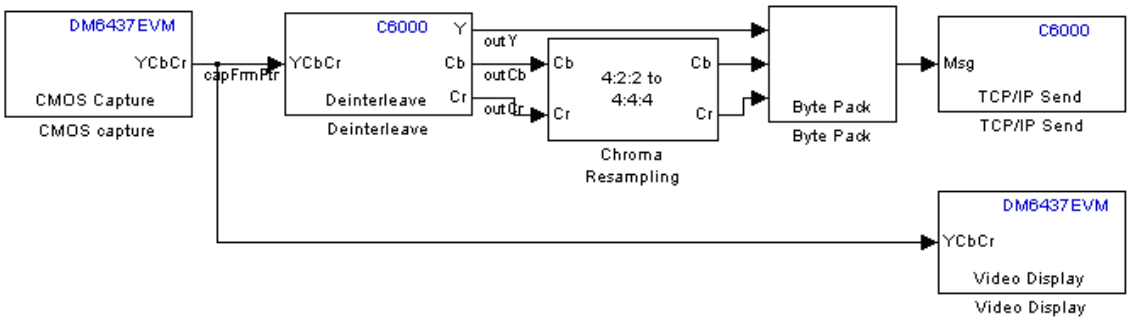
Podešavanje izlaznih pinova EMIF-e se podešava pomoću registra PINMUX0 i polja AEM [2:0]. AEM polje može biti podešeno na dva načina, a uz pomoć SW2 grupe prekidača:

- 100 - 8-bit EMIFA (NAND) + 8-bit CCDC (VPFE) + 16-bit VENC (VPBE)
- 101 - 8-bit EMIFA (NAND) + 16-bit CCDC (VPFE) + 8-bit VENC (VPBE)

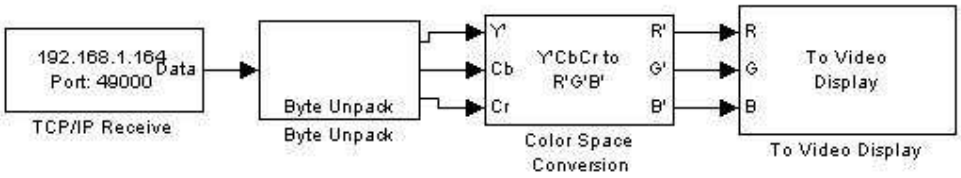
Bootloader prihvata informacije u formi skripte, nazvane AIS (eng. Application Image Script). Ova skripta je binarna datoteka koja sadrži razne komande koje interpretira i izvršava *booloader*. Prevedeni kod se uz pomoć perl skripte *genAIS.pl* prevodi u AIS format a upisivanje u NAND memoriju se vrši pomoću projekta NANDwriter. Čime je ploča spremna za podizanje program sistema iz NAND memorije.

8.2 Primer modela za prenosa slike preko LAN-a

Pošto ploča sadrži EMAC kontroler, omogućen je prenos podataka preko LAN (eng.Local Area Network) mreže. U datom primeru (Slika 37) slika se preuzima sa CMOS senzora i preko TCP/IP konekcije šalje prijemnoj strani. Prijemna strana (Slika 38) je model koji se izvršava na PC računaru, i prikazuje primljenu sliku u video prozoru (Slika 39).



Slika 37 – Primer modela za slanje slike preko TCP/IP protokola



Slika 38 – Primer modela za prijem slike na PC računaru

Slika 39 – Video prozor za prikaz slike