SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1695

Sustav za automobilsku kameru visokog dinamičkog opsega temeljen na FPGA sklopu

Nikola Šale

Zagreb, lipanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 9. ožujka 2018.

DIPLOMSKI ZADATAK br. 1695

Pristupnik: **Nikola Šale (0023102269)**
Studij: Elektrotehnika i informacijska tehnologija
Profil: Elektroničko i računalno inženjerstvo

Zadatak: **Sustav za automobilsku kameru visokog dinamičkog opsega temeljen na FPGA sklopu**

Opis zadatka:

U sklopu diplomskog rada potrebno je osmisliti, projektirati i realizirati ugradbeni računalni sustav za prihvat i obradu digitalne slike s dvodimenzionalnog senzora Sony IMX390 Automotive HDR sensor koji podržava oslikavanje visokog dinamičkog opsega svjetline u kadru fotografije. Potrebno je implementirati cjelokupni podatkovni cjevovod za prihvat i obradu slike sa senzora koristeći raspoložive Xylon ISP, HDR i AWB IP jezgre i softverske biblioteke, te razviti nove odnosno modificirati dijelove postojećeg dizajna za IMX290 senzor koji su nužni radi prilagodbe sustava na novi tip senzora. ISP podatkovni cjevovod treba biti konfiguriran i po potrebi modificiran za optimalnu kvalitetu slike. Sustav treba biti implementiran korištenjem porodice 7 Series Xilinx SoC All programabilnih sklopova. U svrhu implementacije prototipa sustava, koristit će se Sony IMX390 modul, Xylon FMC proto board i ZC706 razvojni sustav. Realizirani sustav kamere potrebno je testirati u propisanim uvjetima za optimalnu kvalitetu slike.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 29. lipnja 2018.

Mentor:
Prof. dr. sc. Davor Petrinović

Djelovođa:
Prof. dr. sc. Dražen Jurišić

Predsjednik odbora za diplomski rad profila:
Prof. dr. sc. Mladen Vučić

SADRŽAJ

1. Uvod

Suvremeni automobili izrađuju se sa sve većim brojem različitih ugrađenih sustava s kamerama čiji je glavni cilj asistencija vozaču u vožnji i povećanje sigurnosti svih putnika u vozilu. Takvi sustava morali bi pouzdano raditi pri različitim uvjetima vožnje, poput vožnje danju i noću, po sunčanom i oblačnom vremenu, naglo promjenjivim uvjetima, poput izlaska iz tunela na danje svjetlo i slično. To upravo i predstavlja najveći izazov pri realizaciji takvih sustava.

FPGA sklopovi predstavljaju vrhunac tehnologije u pogledu fleksibilnosti sustava. Pružaju vrlo visok stupanj slobode pri izradi složenih sustava, te jednostavnu ponovljivost i mogućnost brze izmjene dizajna. Integracijom FPGA sklopa s fiksno ožičenim procesorskim sustavom na istom čipu ujedinjuju se sve njihove prednosti, što omogućuje realizaciju visoko složenih sustava visokih performansi. Zbog činjenice da se FPGA i procesorski sustav nalaze na istom čipu omogućena je velika propusnost i velike brzine prijenosa podataka među njima, za razliku od rješenja s dva odvojena čipa. Upravo iz tog razloga Xilinx Zynq SoC odabran je kao platforma za razvoj ovog sustava.

U drugom poglavlju dan je kratak pregled primjera primjene kamera u automobilskim sustavima te su objašnjene glavne karakteristike takvih kamera. U trećem poglavlju opisan je razvojni sustav i pojedine hardverske komponente koje se koriste u realizaciji sustava. U četvrtom poglavlju objašnjen je formirani model sustava po čijem se uzoru izrađuje konkretan sustav. U petom poglavlju opisan je proces razvoja sustava i njegova funkcionalnost.

2. Kamere u automobilskim sustavima

Suvremeni automobili imaju sve veći broj ugrađenih sustava s kamerama namijenjenim za različite primjene, sa svrhom pomaganja vozaču za vrijeme vožnje i povećanja sigurnosti svih putnika u vozilu. Takvi sustavi pripadaju ADAS (*eng. Advanced Drivers-Assistance Systems*) kategoriji sustava.
Sustavi s kamerama počinju sve više zamjenjivati zrcala omogućavajući bolju preglednost prilikom vožnje u svim uvjetima. Inteligentni *Surround View 360°* sistemi omogućuju pregled okoline automobila iz ptičje perspektive. Sustavi za detekciju umora i pospanosti vozača analiziraju snimku izraza lica u stvarnom vremenu, te tako sprječavaju velik broj potencijalnih nesreća prouzročenih upravo zbog vozača zaspalih u autu. Sustavi za detekciju prepreka analizom slike u stvarnom vremenu zajedno s drugim sustavima temelj su razvoja autonomnih i polu-autonomnih vozila, dok sustavi za snimanje vožnje omogućavaju pohranjivanje snimki s većeg broja kamera koje snimaju različite scene. To su samo neki od bitnijih sustava s kamerama koji se trenutno koriste ili su u fazi razvoja, a potencijalnih primjena ima i više.
Kamere korištene u takvim sustavima trebale bi zadovoljavati određene kriterije koje se tiču kvalitete slike u različitim uvjetima vožnje.

2.1. Visoki dinamički opseg

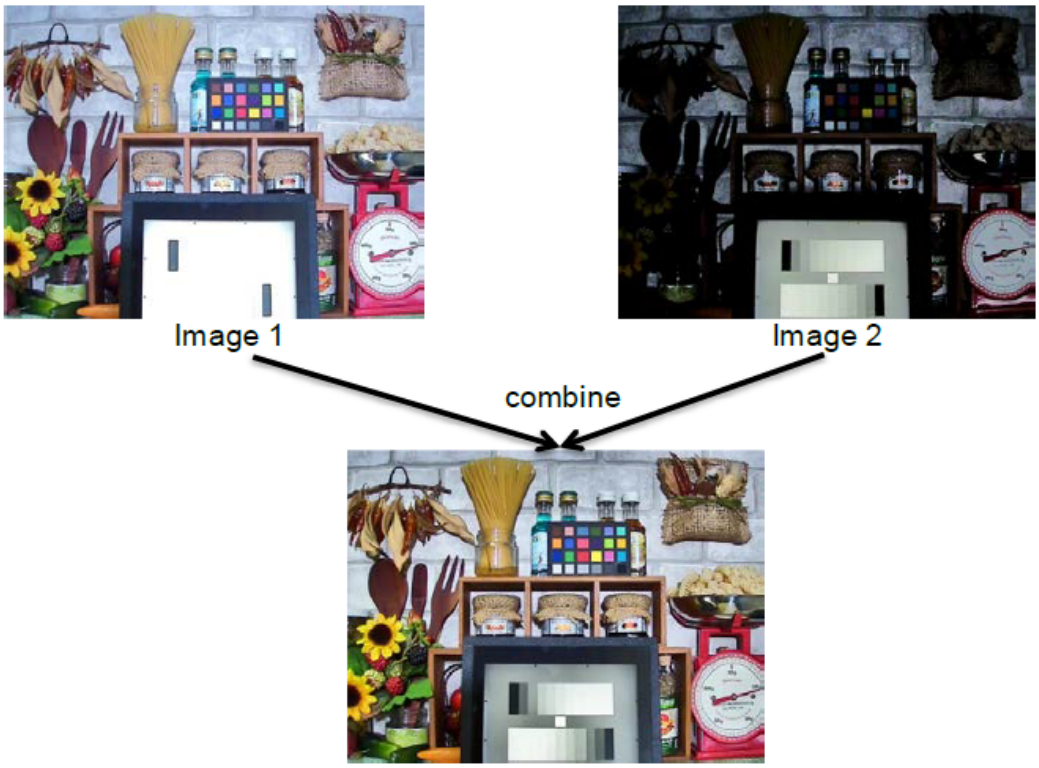
Poznato je da se informacija o pojedinom pikselu slike dobiva u obliku generiranog naboja koji je posljedica fotoelektričnog efekta. Količina generiranog naboja direktno ovisi o energiji predanoj senzoru u vremenu izloženosti svjetlu. Takva veličina često se naziva ekspozicija i izražava se u *lux* sekundama $[lx \cdot s]$. Svjetlosna energija po jedinici površine naziva se iluminacija i izražava se u *luxima* $\left[lx = \frac{lm}{m^2}\right]$. Na slici broj 5 može se vidjeti raspon vrijednosti iluminacija za različito doba dana i različite vremenske uvjete. Raspon vrijednosti razlikuje se u više redova veličine.

Uvjeti	Iluminacija (lx)
Dnevno svjetlo	**10000**
Oblačan dan	**1000**
Vrlo oblačan dan	**100**
Sumrak	**10**
Duboki sumrak	**1**
Puni mjesec	**0,1**
Prva četvrt	**0,01**
Bez mjesečine/zvijezde	**0,001**

Slika 2.1: Iluminacija

Osjetljivost senzora na svjetlo ne pokriva cijeli raspon. Maksimalni nivo senzora određen je kapacitetom pixela i predstavlja zasićenje, dok je minimalni nivo ograničen razinom šuma. Dinamički raspon predstavlja omjer maksimalnog i minimalnog nivoa izraženog u decibelima.

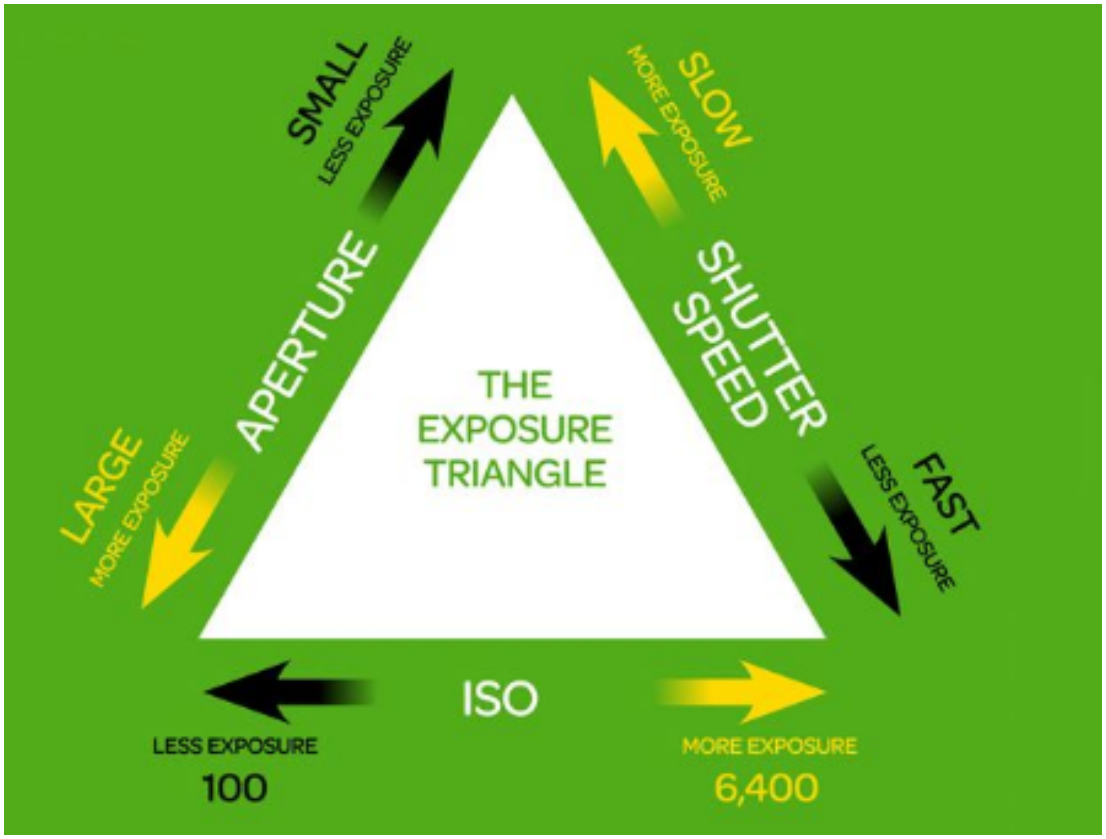
Proširenje dinamičkog raspona izvodi se HDR kompozicijom slika s dvaju ili više senzora različitih osjetljivosti. Većina senzora izrađuje se od silicija, a različite osjetljivosti postižu se različitim dopiranjem, koje utječe na efikasnost pretvorbe fotona u elektrone (kvantna efikasnost) i različitom realizacijom mikro-leća, koje utječu na postotak efikasne površine senzora, tj. površine senzora koja pretvara fotone u naboj (faktor ispune). Rezultat HDR kompozicije je slika koja snima visoko kvalitetne slike u širokom dinamičkom opsegu čak i u situacijama velikog kontrasta pojedinih dijelova slike.



Slika 2.2: HDR kompozicija

2.2. Kontrola ekspozicije

Bilo da se radi o običnom senzoru ili senzoru koji podržava oslikavanje u visoko dinamičkom rasponu, ne uspijeva se obuhvatiti cijeli željeni raspon iluminacije. Sliku je potrebno dobiti u svim uvjetima. To je moguće ostvariti kontroliranjem količine svjetlosne energije koja dopire do senzora, tj. kontroliranjem ekspozicije. Ekspozicija ovisi o tri čimbenika. Prvi čimbenik predstavlja vrijeme izloženosti svjetlosti što je direktno povezano s brzinom zatvarača (*eng.Shutter Speed*). Drugi čimbenik predstavlja otvorenost leće, što određuje kolika količina svjetlosti dopire do senzora (*eng. Aperature*). Treći čimbenik predstavlja osjetljivost senzora (*ISO*) koja je direktno povezana s pojačanjem signala u senzoru. Promijeni li se jedan od parametara potrebno je uravnotežiti bilo koji od druga dva parametra za ekvivalentan iznos kako bi ekspozicija ostala identična. Drugim riječima, ista ekspozicija može se dobiti na bezbroj načina različitim kombinacijama ova tri parametra.



Slika 2.3: Ekspozicijski trokut

Različite vrijednosti parametara utječu na različita svojstva slike. Tako otvor leće definira dubinsku oštrinu slike, dok brzina zatvarača definira duljinu trajanja ekspozicije, što utječe na zapis objekta u pokretu. Kraće trajanje ekspozicija dati će oštriju sliku objekta u pokretu. Veće pojačanje signala u senzoru negativno djeluje na odnos signal-šum, što rezultira više zašumljenom slikom.

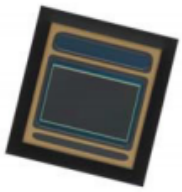
2.3. Potiskivanje LED treperenja

U današnjem prometnom okruženju LED izvori svjetla često su korišteni za prometne znakove i signale. To znači da automobilske kamere moraju moći precizno snimiti takve elemente. LED svjetla koja se napajaju izmjeničnom strujom gradske mreže trepere duplo većom frekvencijom od frekvencije napajanja jer se u jednom naponskom periodu dva puta postigne maksimum i minimum osvjetljenja. Takvo treperenje nije vidljivo ljudskom oku, no kada je snimano kamerom može doći do pojavljivanja efekta treperenja. Problem treperenja može se riješiti povećanjem vremena ekspozicije preko vrijednosti perioda treperenja. Tako se eliminira nastanak okvira snimljenih za vrijeme dok je LED svjetlo ugašeno.

Efekt treperenja, osim što može ometati rad sustava koji rade analizu slike, može i negativno utjecati na sposobnosti i zdravlje čovjeka. Treperenje može uzrokovati distrakciju, zamućenje vida, naprezanje oka, migrene, glavobolje, pa čak i epileptične napade.

2.4. Sony IMX390 senzor

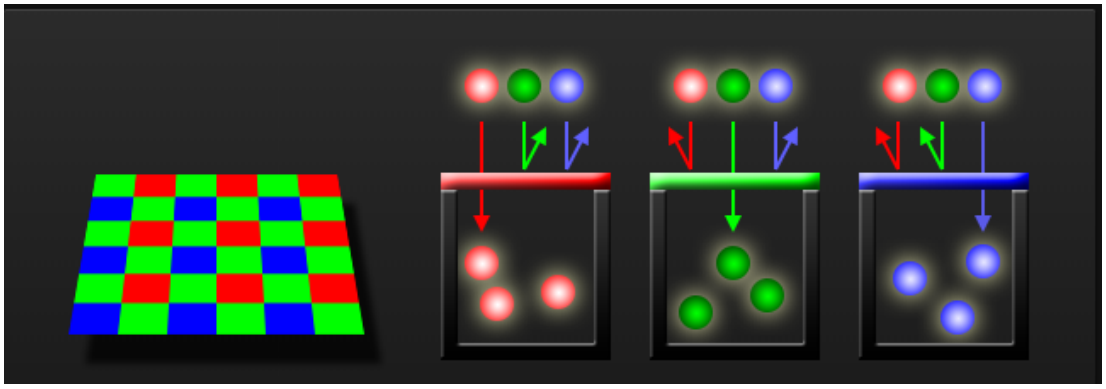
IMX390 novi je Sony-ev 2.45 megapikselni CMOS senzor namijenjen za automobilske kamere. Prvi je senzor na tržištu koji omogućuje istovremeno snimanje u visokom dinamičkom opsegu (120 dB) i potiskivanje LED treperenja. To se postiže odgovarajućom strukturom piksela i metodom ekspozicije. IMX390 senzor pruža mogućnost snimanja visoko kvalitetnih slika u boji i u uvjetima iluminacije od 0.1 *luxa*, koja je ekvivalentna mjesečini. Senzor koristi Bayer filtar te na izlazu daje sliku u RAW (12/20/24) Bayer formatu u Full HD (1920x1080) ili WUXGA (1920x1200) rezoluciji do 60 okvira po sekundi, koju zatim šalje MIPI CSI-2 sučeljem. Komunikacija s aplikacijskim sustavom može se odvijati putem I2C ili SPI sabirnice.



Slika 2.4: IMX390 senzor

2.4.1. Bayer filtar

Velika većina današnjih kamera koristi Bayer filtar za snimanje slika u boji. Bayer filtar ima oblik matrice gdje je svako polje veličine piksela i propušta jednu od tri boje, crvenu, plavu ili zelenu (RGB). Dobivena slika sastoji se od piksela koji sadrže informaciju o samo jednoj od tri boje. Takva slika je u RAW formatu i šalje se iz senzora prema sustavu za obradu. Zbog najveće osjetljivosti ljudskog oka na zelenu boju, matrica sadrži 50% polja koji propuštaju zelenu boju i po 25% polja koju propuštaju plavu i crvenu boju.



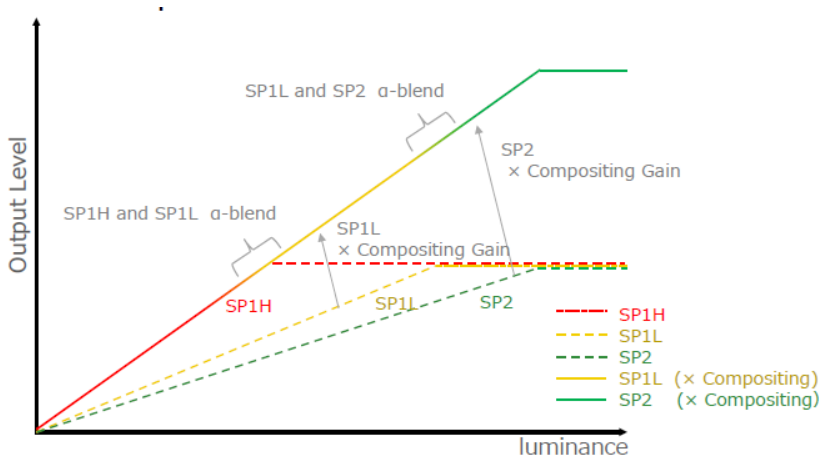
Slika 2.5: Bayer-ov filtar

2.4.2. HDR kompozicija

IMX390 senzor hvata tri slike različite osjetljivosti (SP1H/SP1L/SP2) i komponira ih u rezultantnu sliku visokog dinamičkog opsega. Ovisno o odabranom načinu rada (U20 ili U24) rezultantna slika može sadržavati 20 ili 24 bita po pikselu. Osim toga, senzor ima ugrađenu mogućnost obavljanja po dijelovima linearne kompresije koja podatke može komprimirati na 12 bita kako bi se mogli slati u RAW12 formatu. Tada je potrebno na prijemnoj strani odraditi dekompresiju podataka. U ovisnosti o vremenima ekspozicije, pojačanju i osjetljivosti pojedinih potpiksela, senzor računa kompozicijska pojačanja koja se primjenjuju da bi karakteristika ostala linearna. Na području prijelaza koristi se *alpha blending* za ostvarivanje glatkog prijelaza. Kompozicijska pojačanja računaju se po sljedećim formulama:

$$SP1L\ kompozicijsko\ pojacanje = \frac{SP1H\ vrijeme\ ekspozicije}{SP1L\ vrijeme\ ekspozicije} \times \frac{SP1H\ pojacanje}{SP1L\ pojacanje} \times \frac{SP1H\ osjetljivost}{SP1L\ osjetljivost}$$

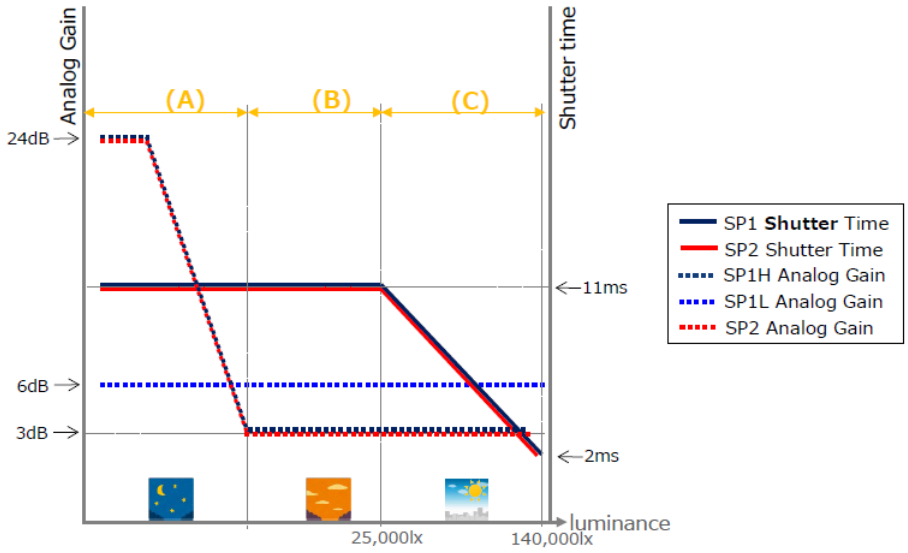
$$SP2\ kompozicijsko\ pojacanje = \frac{SP1H\ vrijeme\ ekspozicije}{SP2\ vrijeme\ ekspozicije} \times \frac{SP1H\ pojacanje}{SP2\ pojacanje} \times \frac{SP1H\ osjetljivost}{SP2\ osjetljivost}$$



Slika 2.6: IMX390 HDR kompozicija

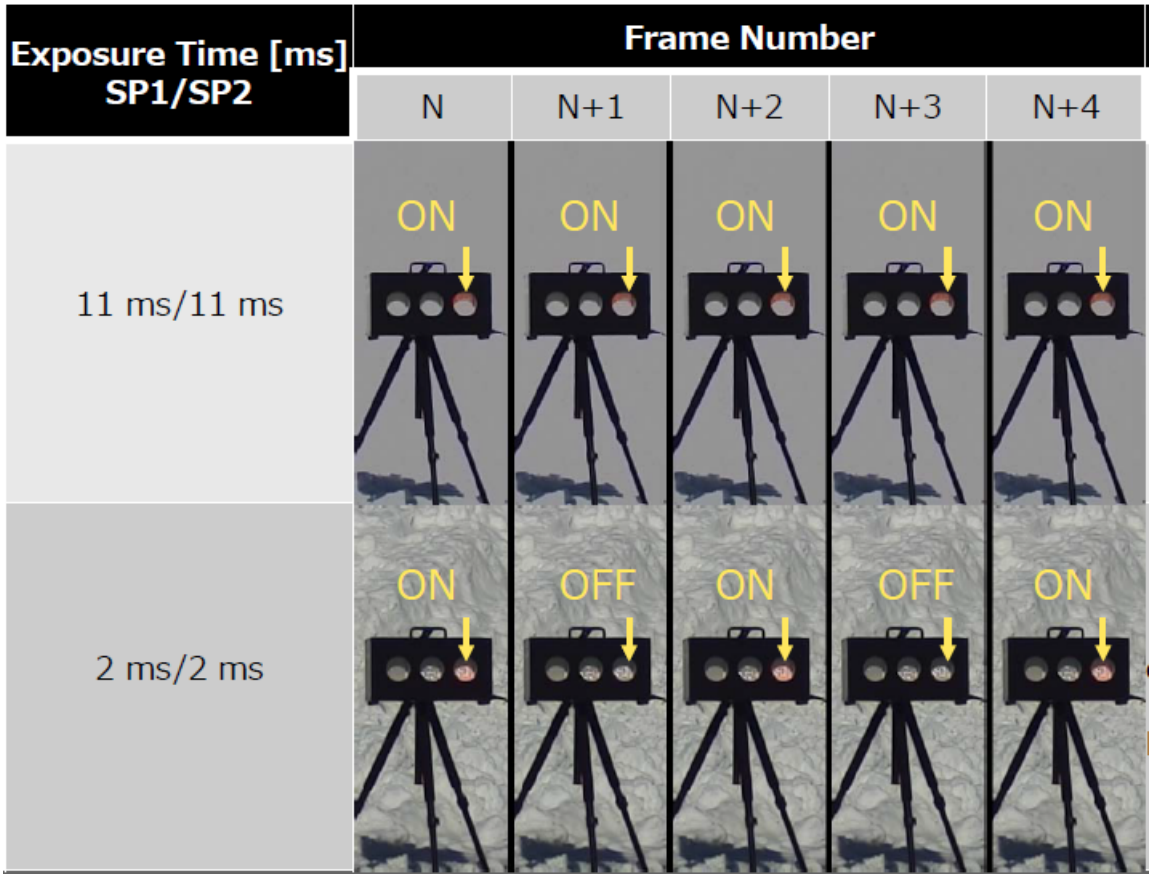
2.4.3. Kontrola ekspozicije i potiskivanje LED treperenja

Metoda kontrole ekspozicije ovisi o postavkama analognog pojačanja za SP2 i odabranom izlaznom formatu (U20 ili U24). Pojačanja za SP1L i SP1H su nezavisna i mogu se postavljati po potrebi, a pojačanje za SP2 može se odabrati da bude jednako ili pojačanju za SP1L, ili jednako pojačanju za SP1H. Na slici 2.7 prikazana je metoda kontrole pojačanja i vremena ekspozicije za U20 format s postavljenom vrijednosti pojačanja za SP2 jednakom pojačanju za SP1H.



Slika 2.7: Kontrola ekspozicije, SP2 AG= SP1H AG, U20

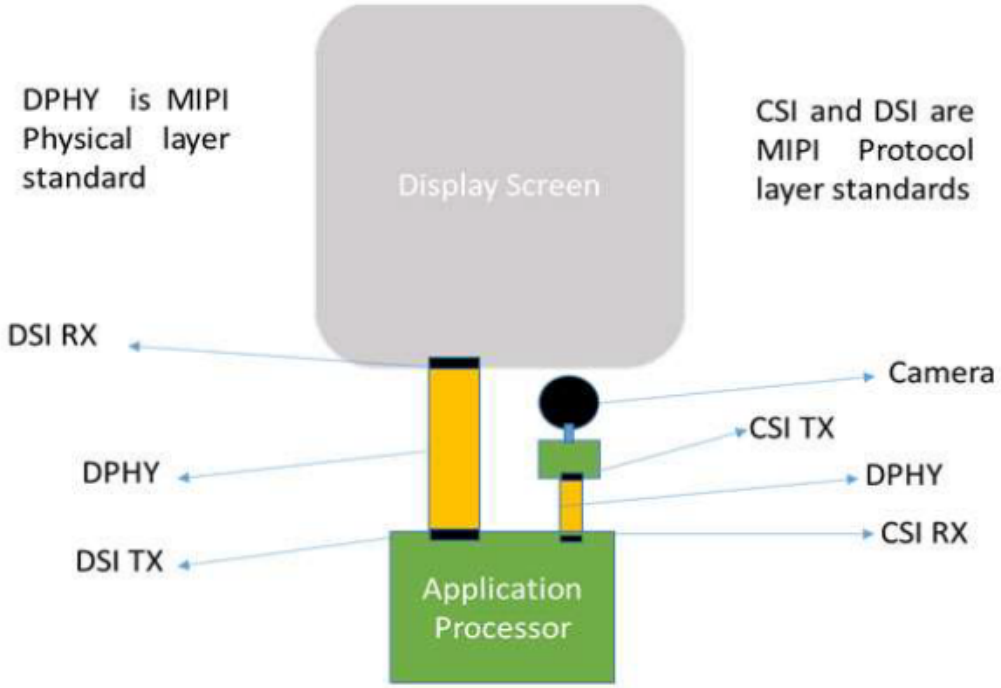
Postavljanjem jednakih vremena ekspozicije za sve tri slike izbjegavaju se mogući artefakti koji se mogu pojaviti snimanjem objekta u pokretu. Postavljanjem vremena ekspozicije na 11 ms uspješno se potiskuju treperenja LED izvora svjetla za frekvencije jednake ili veće od 90 Hz. Međutim, pri velikoj iluminaciji potrebno je smanjiti vrijeme ekspozicije kako bi se izbjeglo zasićenje piksela što negativno utječe na funkciju potiskivanja LED treperenja. Što je manje vrijeme ekspozicije, to je veća minimalna frekvencija treperenja LED svijetla koju senzor može potisnuti. Tako, ako se vrijeme ekspozicije spusti na 2 ms, senzor potiskuje treperenje na frekvencijama jednakim ili većih od 500 Hz. Pojačanje za SP1L fiksira se na 6 dB, a pojačanja za SP1H i SP2 pomiču se u granicama od 3 dB do 24 dB, ovisno o uvjetima osvjetljenja.



Slika 2.8: Vrijeme ekspozicije i potiskivanje LED treperenja

2.5. MIPI CSI-2 protokol

MIPI CSI-2 (*eng. Mobile Industry Processor Interface, Camera Serial Interface*) protokol specificira sučelje između kamere i aplikacijskog sustava. Definiran je od strane MIPI saveza čija je svrha uspostavljanje standarda za hardverska i softverska sučelja u mobilnim uređajima. MIPI CSI-2 protokol odvija se na MIPI D-PHY fizičkom sloju za serijsku komunikaciju. D-PHY fizički sloj uključuje modul za brzi prijenos serijskih podataka i modul za rad u načinu niske potrošnje, što pomaže u postizanju energetske učinkovitosti koja je posebno bitna kod mobilnih uređaja s ugrađenom baterijom. Korisni podatci (slika) šalju se brzim modulom, dok se kontrolni i statusni podatci šalju modulom niske potrošnje. D-PHY sadrži diferencijalnu liniju za takt i do 4 diferencijalne linije za prijenos podataka. Propusnost podataka može se povećati korištenjem više linija. Osim CSI, D-PHY podržava i DSI (eng. Display Serial Interface) protokol za povezivanje monitora.



Slika 2.9: MIPI D-PHY

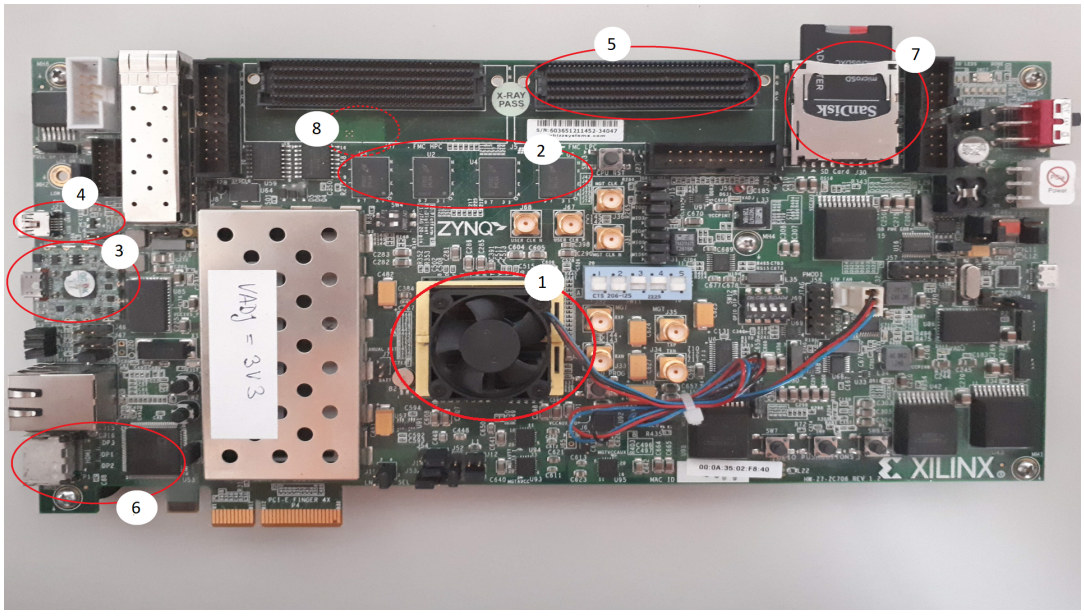
MIPI CSI je najkorištenije sučelje za kameru u mobilnoj industriji zbog jednostavnosti uporabe i sposobnosti za podršku širokog raspona aplikacija visokih zahtjeva, uključujući FHD, 4K, 8K i više. Svi pametni telefoni imaju barem jedno MIPI sučelje, a često se koriste u tabletima, laptopima, automobilskim i IoT (eng. *Internet of Things*) sustavima.

3. Razvojni sustav

3.1. ZC706 razvojna pločica

ZC706 razvojna je pločica za Zynq-7000 XC7Z045 AP SoC(1) (eng. All Programmable System on Chip) koja pruža potrebno hardversko okruženje za razvoj i evaluaciju rješenja uobičajenih za mnoge ugradbene sustave. Bitnije hardverske komponente koje su sastavni dio razvijenog sustava ili su u nekoj fazi razvoja korištene su:

- 1GB DDR3(4x256MB) memorija(2)
- USB-JTAG modul i mikro-b USB konektor(3)
- USB-UART premosnik i mini-B USB konektor(4)
- FMC LPC konektor(5)
- ADV7511 HDMI kontroler i HDMI konektor(6)
- SD konektor(7)
- PCA9548 I2C prespojnik(8, donja strana)



Slika 3.1: Razvojni sustav

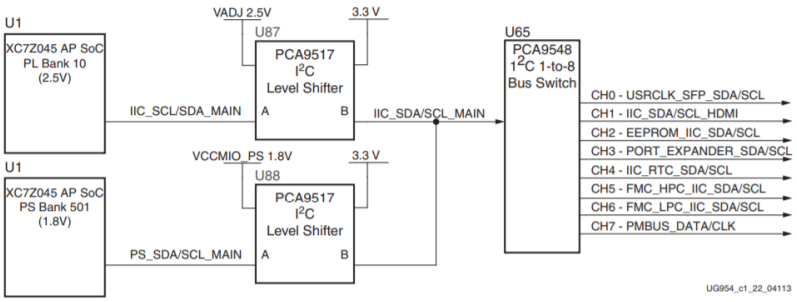
DDR memorija služi kao radna memorija za procesorski sustav. Programiranje procesorskog sustava PS i programabilne logike PL unutar Zynq SoC-a, za vrijeme faze razvoja, obavlja se preko JTAG sučelja. Kasnije, kada je prva završna verzija spremna, programiranje se izvršava pokretanjem kreiranog sadržaja sa SD kartice prilikom paljenja napajanja. U kasnijim poglavljima detaljnije će biti opisan postupak kreiranja sadržaja SD kartice.

Komunikacija sustava s korisnikom odvija se preko UART-USB sučelja. Za uspostavljanje komunikacije potrebno je na korisničkom računalu imati instaliran terminal program, no može se koristiti i ugrađeni terminal u sklopu Xilinx SDK-a. Prije komunikacije potrebno je odabrati naziv priključka na kojem je spojen drugi kraj USB kabela te definirati brzinu prijenosa podataka na 115200 bauda, da odgovara inicijalno postavljenoj brzini prijenosa od strane procesora.

Na 160 pinski FMC LPC konektor spaja se Xylon FMC pločica na kojem je spojen IMX390 modul. ADV7511 HDMI kontroler pretvara izlazni video signal u format pogodan za slanje HDMI sučeljem. Inicijalizacija IMX390 senzora i ADV7511 HDMI kontrolera izvodi se putem I2C sabirnice.

I2C sabirnica

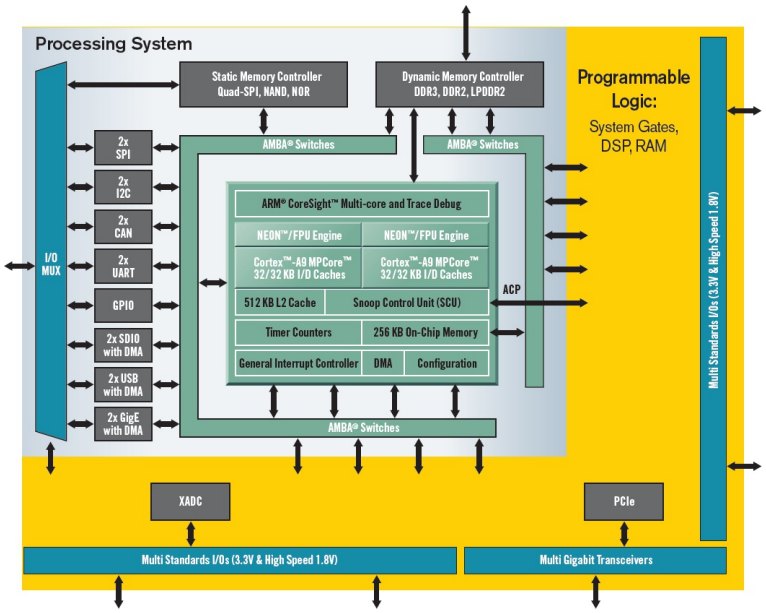
Na ZC706 razvojnoj pločici implementirana su dva I2C priključka, jedan od strane PL-a i jedan od strane PS-a. Oba priključka spajaju se na zasebni podizač naponskog nivoa tako da na izlazu imaju naponski nivo od 3.3 V. Izlazi se spajaju na zajednički priključak koji je spojen na 1-8 kanalni PCA9548 I2C prespojnik. Prije komunikacije s nekim od uređaja na jednom od kanala potrebno je inicijalizirati prespojnik da omogući komunikaciju na tom kanalu. Svaki pojedini kanal ili kombinacija više kanala može biti odabrana. Kada je kanal odabran, prespojnik postaje transparentan za komunikaciju. Sabirnica može raditi na brzinama do 400 kHz.



Slika 3.2: I2C sabirnica

3.1.1. Zynq-7000 XC7Z045 SoC

Zynq-7000 XC7Z045 pripada porodici 7 series Xilinx SoC All programabilnih sklopova. Sastoji se od procesorskog sustava s dvije ARM Cortex A9 procesorske jezgre i Xilinx Kintex-7 FPGA programabilne logike integrirane na istom čipu. Takva integracija procesorskog sustava i programabilne logike na jednom čipu omogućuje mnogo bolje performanse u odnosu na rješenja sa zasebnim čipovima. Uklanja se potreba za komunikacijom između čipova vezama ograničene propusnosti podataka, poput PCI-Express-a, te se eliminira zahtjev za serijalizacijom i deserijalizacijom podataka jer se koristi velik broj unutarnjih veza. Rezultati su veće brzine prijenosa podataka, manja latencija, smanjenje potrošnje do 50% i smanjenje ukupne cijene sustava.



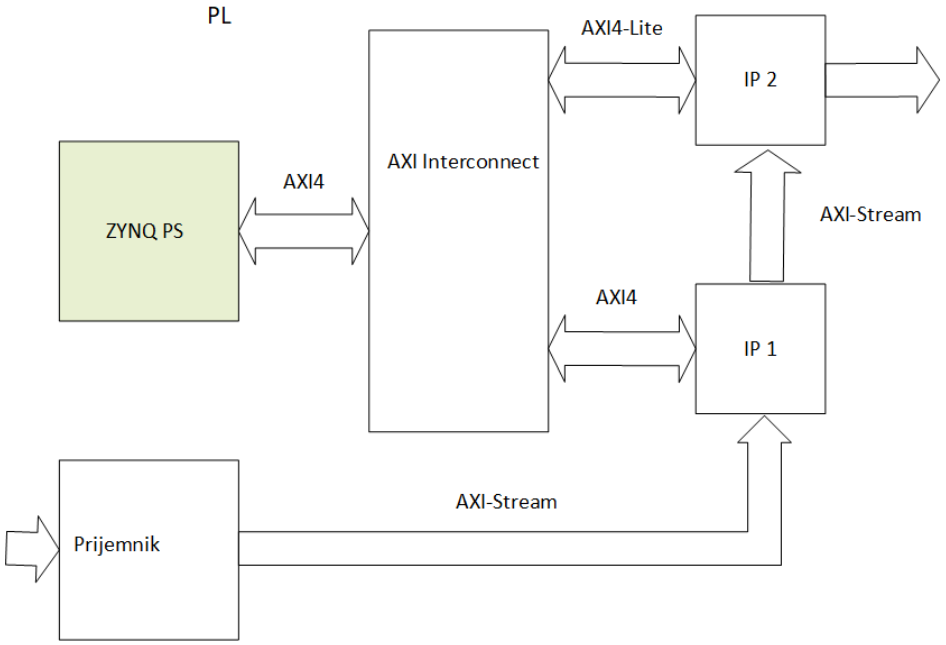
Slika 3.3: Zynq-7000 XC7Z045 SoC

Kombinacija procesorskih sustava i FPGA ujedinjuje njihove prednosti. Procesorski sustav optimiziran je za izvršavanje raznolikog broja operacija s mogućnošću brze promjene konteksta, dok FPGA realizira logičke blokove koji obavljaju ponovljive visoko paralelizirane zadatke koji se ponavljaju velik broj ciklusa i samo su povremeno redefinirani. Tako FPGA omogućuje korisnicima implementaciju algoritama u logici što znatno olakšava posao procesoru, a kao rezultat dobiju se puno brži sustavi. Osim toga FPGA može pružati puno veći broj operacija po jedinici energije od procesorskih rješenja, što rezultira i mnogo efikasnijim sustavima. Razvijeni logički blokovi često se nazivaju IP (eng. Intellectual Property) jezgrama ili skraćeno IP.

ARM AMBA AXI4 sabirnica

Prijenos podataka između logičkih blokova međusobno i procesorskog sustava i logičkih blokova odvija se preko ARM AMBA AXI4 sabirnice. Postoje tri tipa AXI4 sučelja:
- AXI4 - za memorijski mapirane blokove s visokim zahtjevima performansi
- AXI4-Lite - za memorijski mapirane blokove s niskim zahtjevima performansi
- AXI-Stream - za prijenos podataka velikim brzinama među memorijski ne mapiranim blokovima (npr. video stream)

Registri memorijski mapiranih blokova nalaze se u adresnom prostoru procesorskog sustava. Svi AXI *master* i slave uređaji povezuju se preko AXI Interconnect bloka koji je zadužen za dekodiranje adresa. AXI-stream sučelje služi za prijenos podataka između blokova koji nemaju memorijski mapirane registre. Kod AXI-Stream protokola, za razliku od AXI4 i AXI4-Lite protokola, *master* uređaji mogu samo slati tj. upisivati podatke na sabirnicu, a slave uređaji samo primati tj. čitati podatke koje šalje *master* uređaj. Adresiranje u tom slučaju nije potrebno.



Slika 3.4: Primjer povezivanja AXI4 sabirnicom

3.2. Xylon FMC pločica i IMX390 modul

Xylon FMC je tiskana pločica razvijena za izradu prototipa sustava za evaluaciju i testiranje IMX314 senzora. No, može se koristiti i za istu svrhu i kod IMX390 senzora

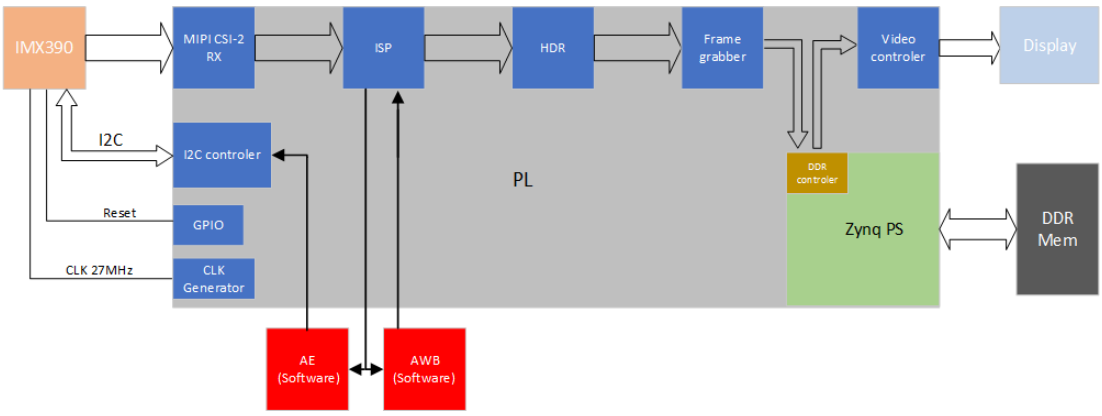
pravilnim spajanjem IMX390 modula. Preko FMC konektora priključuje se na ZC706 razvojnu pločicu.

Slika 3.5: Xylon FMC pločica i IMX390 modul

Xylon FMC pločica sadrži naponski podešivač nivoa za prilagodbu napona za I2C sabirnicu s 3.3V na 1.8V potrebnih za komunikaciju s IMX390 senzorom. Zynq-7000 nema I/O pinove koji nativno podržavaju D-PHY specifikaciju, pa je potrebno ugraditi dodatne komponente izvan FPGA kako bi se osigurala kompatibilnost. Svaka podatkovna linija rastavlja se na posebnu liniju za slanje podataka velike brzine, te na liniju niske potrošnje za slanje kontrolnih paketa.
Iz sheme IMX390 modula potrebno je odrediti što je spojeno na priključcima koji definiraju način komunikacije (I2C ili SPI) i adresu senzora na sabirnici. Slika sheme ne smije se priložiti zbog zaštite intelektualnih prava.

4. Model sustava

Na slici 4.1 nalazi se formalni model sustava za automobilsku kameru sa Sony IMX390 senzorom. Model je osmišljen po uzoru na sličan sustav namijenjen za prethodni Sony IMX290 senzor.



Slika 4.1: Model sustava

Za pravilan rad senzor zahtijeva takt od 27 MHz, a inicijalizacija senzora odvija se I2C sučeljem. Izvedena je i posebna linija za resetiranje senzora. Podaci sa senzora šalju se MIPI CSI-2 sučeljem do odgovarajuće IP jezgre prijemnika realiziranog u FPGA. Prijemnik šalje primljene podatke do ISP IP jezgre. U ISP IP-u odvija se obrada primljene slike. Prvo se odrađuju korekcije oštećenih piksela, zatim pretvorba slike iz Bayer RAW formata u RGB format, balansiranje boja i gama korekcija. Osim toga ISP sadrži poseban blok koji računa različitu statistiku slike koju koriste AE i AWB softverske biblioteke, čija je svrha kontrola ekspozicije na senzoru i balansiranje boja. Obrađena slika šalje se dalje do HDR IP jezgre. Glavni zadatak HDR IP-a je mapiranje slike visokog dinamičkog opsega (više od 8 bita po boji) u dinamički opseg pogodan za prikaz na zaslonu (8 bita po boji), uz očuvanje detalje slike. Sljedeća IP jezgra do koje se šalju video podatci ima ulogu spremanja okvira slike u memoriju iz koje zatim video kontroler dohvaća te okvire i šalje ih na zaslon. Hvatač okvira i video kontroler

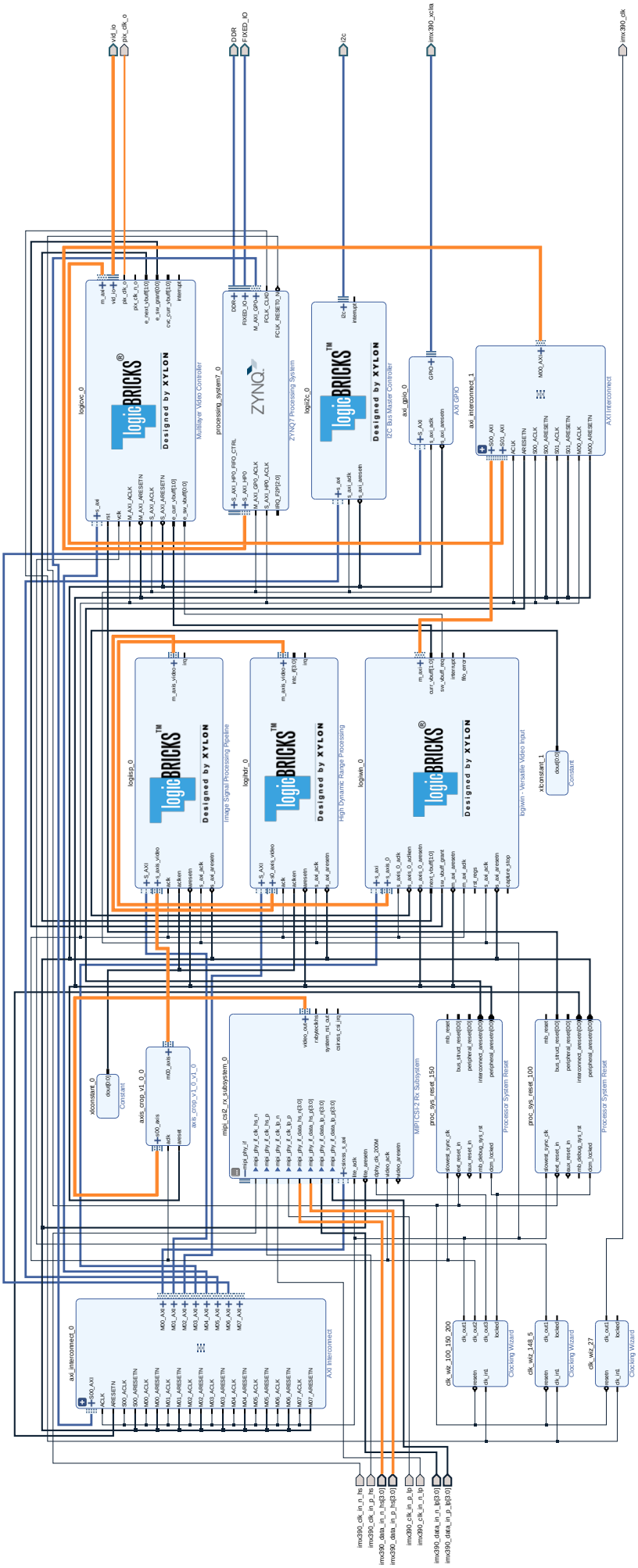
pristupaju DDR memoriji preko DDR kontrolera koji pripada procesorskom sustavu. Prilikom inicijalizacije IP jezgri, u memoriji se rezervira dio memorije namijenjen za spremanje okvira videa.

5. Izrada sustava

Korišteno razvojno okruženje uključuje dva softverska alata, Xilinx Vivado Desing Suite 2017.4 i Xilinx SDK 2017.4. Vivado pruža potrebne alate za sintezu i analizu digitalnog dizajna, a SDK pruža potrebno okruženje za razvoj programske podrške.
Zbog improvizirane fizičke izvedbe sustava planirano je da se senzor postavi na način rada u kojem šalje 30 okvira po sekundi te da se koriste sve četiri linije na MIPI sučelju kako bi se smanjila brzina prijenosa, a s time i mogućnost greške u prijenosu na MIPI sučelju. Iako je početna želja bila raditi obradu slike na 24-bitnim ili 20-bitnim podatcima koje senzor šalje u RAW24 ili RAW20 formatu, zbog trenutnog ograničenja u samom dizajnu, odabran je izlazni format RAW12. Trenutno ograničenje predstavlja činjenica da raspoloživi ISP blok (logiISP) može primiti maksimalno 12 bita na ulazu. Zbog toga senzor šalje samo 12 najznačajnijih bitova što rezultira velikim gubitkom dinamičkog raspona slike. Cijeli potencijal senzora moći će biti iskorišten tek nakon planirane modifikacije logiISP IP-a.

5.1. Izrada dijagrama dizajna

Prilikom otvaranja novog projekta u Vivado-u odabran je RTL tip projekta. RTL projekt omogućava izradu i analizu dizajna te pokretanje sinteze, implementacije i generiranje konfiguracijske datoteke koja služi za konfiguraciju FPGA sklopa. U sljedećem koraku odabrana je Zynq-7 ZC706 evaluacijska pločica kao ciljana platforma. U lijevom dijelu prozora pod opcijom IP integrator klikom na "*Create Block Desing*" otvoren je novi prozor namijenjen izradi dijagrama sustava. Klikom na plus znak u alatnoj traci dodaju se IP jezgre iz biblioteke u trenutni dizajn. IP jezgre povezuju se jednostavnim spajanjem linijom od jednog do drugog priključka. Ulazni i izlazni priključci koji predstavljaju I/O pinove čipa, definiraju se desnim klikom i odabirom "*Create Port*". Odabir standarda I/O priključaka, kao i zadavanje vremenskih ograničenja, obavlja se u zasebnoj datoteci koja će biti objašnjena u poglavlju vezanom za implementaciju dizajna. Na slici 5.1 nalazi se gotov dijagram dizajna sustava.



Slika 5.1: Dijagram dizajna

Prijenos podataka između različitih dijelova sustava odvija se na AXI4 sabirnici. Za prijenos podataka među memorijski mapiranih blokova u sustavu koristi se AXI4 protokol, a za prijenos među memorijski ne mapiranih blokova, AXI-Stream protokol. Različiti dijelovi sustava rade na različitim taktovima. Takt za AXI4 sučelje iznosi 100 MHz, a za AXI-Stream sučelje 150 MHz. Osim toga, MIPI CSI-2 RX zahtijeva takt od 200 MHz te logiCVC takt od 148.5 MHz (za Full HD rezoluciju,60 fps). Dodatno, potrebno je generirati i takt za senzor koji iznosi 27 MHz. Za generiranje željenog takta koristi se *Clocking Generator* IP koji koristi ugrađene PLL i MMCM module. Jedan *Clocking Generator* može generirati do sedam taktova na izlazu, ali zato što svi izlazi dijele dio registara koji određuju frekvenciju takta, ne mogu se generirati sve željene kombinacije. Zbog toga koriste se tri *Clocking Generator* IP jezgre. Za generiranje sinkronog *reseta* na sabirnicama koriste se dvije *Proccessor System Reset* IP jezgre, jedan za 100 MHz i jedan za 150 MHz. Neke IP jezgre imaju priključak za omogućavanje ili ne omogućavanje takta zbog uštede energije u trenutcima kada se ne koriste. U ovom slučaju sve takve IP jezgre cijelo vrijeme će biti aktivne pa se na priključak dovodi konstantna vrijednost. Za generiranje *reseta* koristi se *Axi GPIO* IP jezgra, a za komunikaciju sa senzorom putem I2C sabirnice koristi se logiI2C kontroler.

5.1.1. MIPI CSI-2 RX

Podatke koje šalje senzor MIPI CSI-2 sučeljem prima MIPI CSI-2 RX Subsystem IP. Zynq-7000 ne podržava D-PHY specifikaciju pa se dodatnim sklopom izvan FPGA četiri diferencijalne linije odvajaju u osam diferencijalnih linija, od kojih su četiri namijenjene za slanje velikom brzinom i četiri za slanje u načinu rada niske potrošnje. Zbog toga MIPI CSI-2 RX ima posebne priključke za linije velike brzine i za linije niske potrošnje. U novijim porodicama XIlinx SoC-a, poput Ultrascale+ porodice, D-PHY specifikacija je nativno podržana i MIPI CSI-2 RX ima samo 4 priključka za četiri diferencijalne linije kojima se šalju i podatci visokom brzinom i niskom potrošnjom. Pristigli podaci spremaju se u privremeni spremnik iz kojeg se pripremaju za slanje AXI-Stream sabirnicom. Da bi se svi podatci uspješno poslali te da ne bi došlo do prepunjenja spremnika, mora vrijediti sljedeća jednakost :

$$video\_aclk(MHz) >= Brzina\ prijenosa(Mb/s) \times Broj\ podatkovnih\ linija/8/4$$

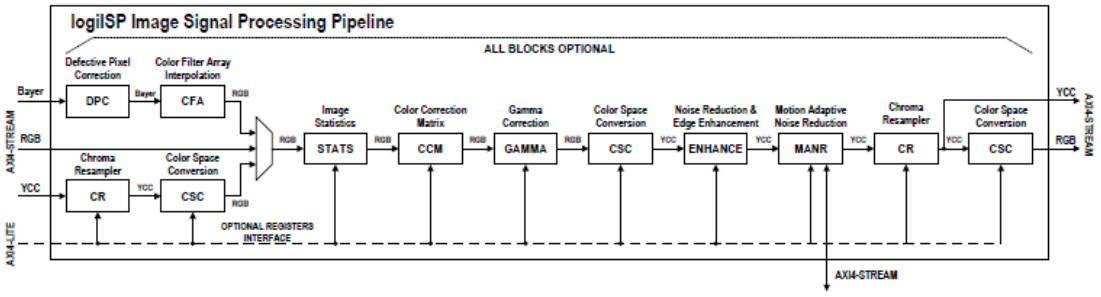
Brzina prijenosa na MIPI sučelju za 30 okvira po sekundi i uz korištenje četiri linije

iznosi 222.75 Mb/s na svakoj liniji. Odabrani takt od 150 MHz zadovoljava taj uvjet.

Trenutno maksimalno podržan format koji MIPI CSI-2 RX može primiti je RAW14, što znači da se podatci sa senzora ne mogu slati nu RAW20 ili RAW24 formatu. Tada bi bilo potrebno koristiti kompresiju unutar senzora i napraviti logički blok koji će raditi dekompresiju. Međutim, zbog daljnjih ograničenja u sustavu to nije potrebno raditi.
Nakon dekodiranja podataka koji se primaju MIPI CSI sučeljem, generira se AXI-stream format podataka koji se šalje do sljedećeg bloka u sustavu, axis_crop-a. Za sada se može zanemaritiAxiscropper blok i pretpostaviti da se video podatci šalju direktno na logiISP IP. Potreba za AxisCropper blokom i njegova funkcija bit će opisana u posljednjem poglavlju.

5.1.2. LogiISP

Slika koju šalje senzor u RAW formatu praktički je neupotrebljiva bez dodatne obrade. Obrada slike obavlja se u LogiISP IP-u koji predstavlja cjevovod za obradu slike. Podržane su rezolucije do čak 7680x7680 piksela te može primati sliku i u Bayer RAW, RGB i YCbCr formatu. U ovom slučaju primljena rezolucija slike je 1936x1096 piksela u RAW formatu. Senzor šalje dodatnih 8 piksela sa svake strane radi obrade na slici koja se temelji na susjednim pikselima. Tako i rubni pikseli željenog područja snimanja imaju susjedne piksele prema rubovima. LogiISP trenutno može primiti na ulazu samo piksele širine do 12 bita. Na slici 5.2 nalazi se cijeli cjevovod od kojeg se u ovom sustavu koriste samo neki blokovi odabrani u grafičkom sučelju koje pruža Vivado. Blokovi koji nisu odabrani ne implementiraju se u FPGA-u. Svaki pojedini implementirani blok može se softverskim putem postaviti u zaobilazni način rada u kojem se taj blok zaobilazi kao da ne postoji.



Slika 5.2: LogiISP cjevovod

DPC blok

DPC blok obavlja detekciju i korekciju oštećenih piksela senzora u stvarnom vremenu. Oštećeni pikseli mogu biti posljedica grešaka u proizvodnji ili grešaka tijekom normalnog rada. Oštećeni pikseli mogu se okarakterizirati kao mrtvi (uvijek niska razina), vrući (uvijek visoka razina) ili zaglavljeni (na neku vrijednost). Korekcija se radi interpolacijom vrijednosti susjednih piksela iste boje.

CFA blok

CFA blok obavlja interpolaciju preostalih dviju boja koje nedostaju u pojedinom pikselu. Originalni podatci za svaki piksel sadrže informacije o samo jednoj boji. Interpolacijom se za svaki piksel generiraju informacije za preostale dvije boje. Ne postoji metoda za potpunu rekonstrukciju boja koje nedostaju već se procjenjuju uzimajući u obzir susjedne piksele. U postavkama LogiISP-a bitno je točno postaviti fazu pristiglog Bayer uzorka. U dokumentaciji senzora može se pronaći taj podatak. U ovom slučaju faza je RG RG, što znači da je u prvoj liniji prvi piksel crvene boje, a drugi piksel zelene boje.

STATS blok

STATS blok generira različit set statističkih podataka za 64 softverski definirane zone na pojedinom okviru slike. Generiraju se podatci o osvijetljenosti zone, histogram boja, srednja vrijednost i varijanca, itd. Prikupljeni statistički podatci koriste se za *Auto-Exposure* (AE) i "*Auto-White-Balance* (AWB) softverske algoritme. Kod kamera s prilagodljivim fokusnim lećama podatci se mogu koristiti i za *Auto-Fokus* (AF) algoritam.

CCM blok

CCM blok koristi se za korekciju boja slike. Sastoji se od 3x3 matričnog množitelja i zbrajala za pomak. Podaci koje pojedini piksel generira za jednu boju rezultat su kombinacije fotona te boje, pomnoženo s odzivom filtra i pomnoženo s odzivom silicija. Odzivi filtra i silicija mogu se bitno razlikovati s odzivom ljudskog oka pa neku boju senzor i ljudsko oko mogu različito interpretirati. Te razlike mogu se ispraviti tako da se više poklapaju s percepcijom ljudskog oka. Koeficijenti na glavnoj dijagonali služe za pojačavanje pojedine komponente dok u isto vrijeme koeficijenti koji nisu na glavnoj dijagonali definiraju dodavanje mješavine ostale dvije komponente. Pomak

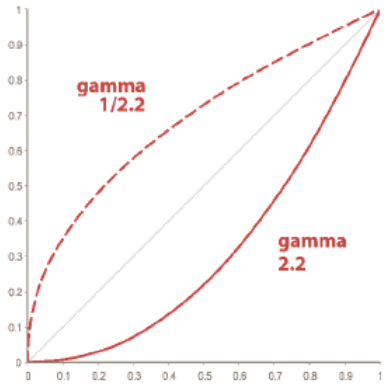
služi za postizanje crnog nivoa [0,0,0]. AWB algoritam mijenjat će koeficijente glavne dijagonale. Za podešavanje ostalih koeficijenata potrebno je raditi kalibraciju boja, ali to ne ulazi u obujam ovog rada.

$$\begin{bmatrix} Rc \\ Gc \\ Bc \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} O_1 \\ O_2 \\ O_3 \end{bmatrix}$$

GAMMA blok

GAMMA blok odrađuje pretvorbu linearnih RGB vrijednosti u vrijednosti koje se bolje poklapaju s nelinearnom karakteristikom ljudskog oka. Karakteristika osjetljivosti ljudskog oka na svjetlinu pri normalnim uvjetima osvjetljenja (ne skroz tamno ili skroz svijetlo) ima oblik eksponencijalne funkcije. Oko pokazuje veću osjetljivost na razlike u nijansama kod tamnijih tonova slike nego kod svjetlijih. Kod slika koje nemaju *gamma* korekciju previše bitova troši se na nijansne koje ljudsko oko ne može razlikovati, a premalo na nijanse koje ljudsko može raspoznati. *Gamma* korekcija upravo radi to da se više bitova dodjeljuje vrijednostima na koje je ljudsko oko osjetljivo i tako se efektivnije iskorištava ograničen dinamički raspon slike. Gamma korekcija je definirana sljedećom jednadžbom:

$$V_{out} = V_{in}^{y}, \qquad V_{out}, V_{in} \epsilon [0, 1]$$

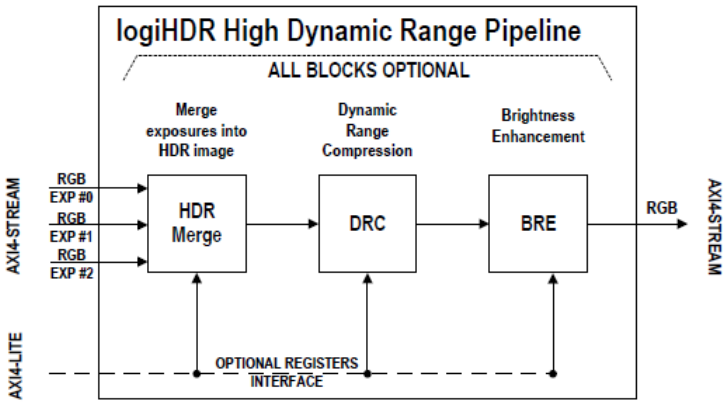


Slika 5.3: Gamma korekcija

5.1.3. LogiHDR

Obrada slike koja uključuje promjene u dinamičkom opsegu slike (broja bitova po pikselu) obavlja se u logiHDR IP-u. LogiHDR može primati slike rezolucija do 7680x7680

piksela u RGB formatu. Širina ulaznih podataka može iznositi do 20 bita po boji (60 bita po pikselu). Cjevovod se sastoji od 3 opcionalna bloka koji se također mogu postaviti u zaobilazni način rada, poput blokova u logiISP-u. Prvi blok, *HDR Merge*, može primati do 3 slike i raditi HDR kompoziciju. Budući da je u ovom sustavu HDR kompozicija već odrađena u samom senzora, taj blok neće se koristiti.



Slika 5.4: logiHDR

DRC blok

DRC blok radi kompresiju ulaznih podataka većeg dinamičkog opsega (broja bitova) u izlazne podatke manjeg dinamičkog opsega, pritom očuvajući detalje slike. U ovom slučaju ulazni podatci širine su 12 bita, a izlazni 8 bita, što odgovara korištenom monitoru. Ako DRC blok bude postavljen u zaobilazni način rada, najneznačajnija četiri bita samo se odbace, što rezultira gubljenjem dijela informacije tamnijih dijelova slike.

BRE blok

BRE blok odrađuje lokalno podešavanje svjetline gdje se korekcija svjetline pojedinog piksela računa s obzirom na susjedne piksele, sa svrhom povećanja vidljivosti detalja na slici. Može se primijeniti na bilo kojoj slici niskog dinamičkog raspona kojoj je potrebno dodatno poboljšanje vidljivosti detalja. U ovom slučaju poboljšanje se radi na slici koja je dobivena kompresijom iz DRC bloka.

5.1.4. LogiWIN

Glavna funkcija logiWIN IP-a je dohvat okvira slike i spremanje u video memoriju. Uz to, logiWIN posjeduje mogućnost skaliranja (povećavanje ili smanjenje rezolucije),

rezanja i pozicioniranja primljene slike. U ovom sustavu koristi se samo mogućnost rezanja okvira slike kako bi se uklonilo dodatnih 8 rubnih piksela koji služe za obradu nad slikom. Izlazna rezolucija iznosi točno 1920x1080 piksela. Podatci se šalju AXI4 sučeljem do procesorskog sustava, točnije do DDR kontrolera, preko kojeg se prosljeđuju do inicijaliziranog dijela u DDR memoriji koji služi kao video memorija. Budući da senzor šalje 30 okvira u sekundi, logiWIN također sprema 30 okvira po sekundi u memoriju.

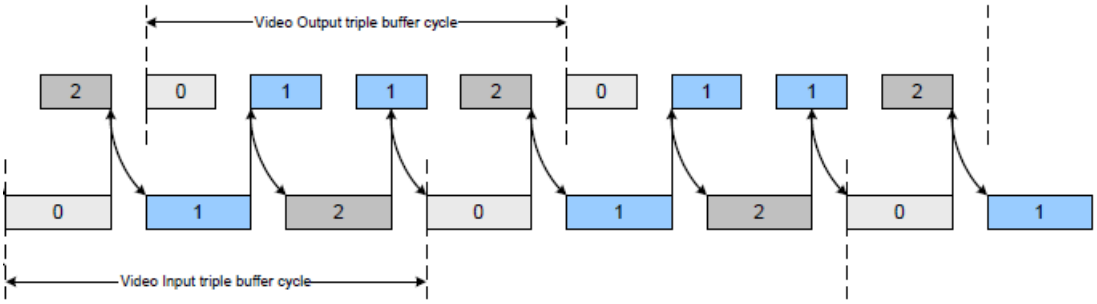
5.1.5. LogiCVC

LogiCVC je video *display* kontroler. Njegova funkcija je osvježavanje slike na monitoru čitajući video memoriju. Pročitani video podatci pretvaraju se u format pogodan za slanje sučeljem do monitora. Osim toga, LogiCVC generira potrebne kontrolne signale za monitor. Većina monitora predviđena je da prima 60 okvira po sekundi uz mala dozvoljena odstupanja. Monitor koji se koristi za razvoj sustava također podržava uzak skup frekvencija osvježivanja od oko 60 okvira po sekundi. Upravo zbog toga, logiCVC podešen je da 60 puta u sekundi osvježava monitor.

Triple buffering

Ako se svi video podatci spremaju u jedan memorijski spremnik te se iz njega čitaju (pogotovo ako se radi o različitim frekvencijama čitanja i pisanja), za očekivati je da će u nekom trenutku čitač vidjeti djelomično ažuriranu sliku u spremniku. To uzrokuje pojavu artefakta na slici prilikom reprodukcije. Dodatni problem predstavlja to što će upisivač morati čekati na pisanje sve dok čitač ne završi s čitanjem. Zbog toga je potrebno koristiti više memorijskih spremnika. U slučaju rješenja s 2 spremnika, čitač, nakon što pročita podatke iz jednog spremnika, mora čekati dok upisivač ne upiše do kraja podatke u drugi spremnik. Isto tako upisivač, nakon što upise podatke u jedan spremnik, mora čekati da čitač završi s čitanjem drugog spremnika. Rješenje s 2 spremnika zadovoljava za slučaj kada su čitač i upisivač sposobni čekati na takav događaj, poput procesora. No, budući da je u ovom sustavu čitač logiCVC koji radi fiksno na svojoj frekvenciji, a upisivač logiWIN koji također radi fiksno na svojoj frekvenciji, to rješenje nije dobro. Tada je potrebno koristiti 3 spremnika jer u tom slučaju logiCVC može odmah krenuti s čitanjem sljedećeg spremnika ili logiWIN s upisivanjem u sljedeći spremnik. Na slici 5.5 može se vidjeti primjer kada čitač radi na većoj frekvenciji od pisača. U takvoj situaciji dogodi se da čitač mora pročitati okvir, a upisivač nije stigao pripremiti novi okvir, pa je čitač prisiljen pročitati ponovno isti okvir. Ako

čitač radi na duplo većoj frekvenciji od pisača, kao što je slučaj (60Hz logiCVC i 30Hz logiWin) u ovom sustavu, čitač svaki okvir pročita točno 2 puta.



Slika 5.5: Triple buffering

5.2. Dodjeljivanje adresnog prostora IP blokovima

Nakon izrade dijagrama dizajna sustava potrebno je dodijeliti adresni prostor pojedinom IP bloku. Svakom bloku dodijeljen je adresni prostor kao višekratnik 64KB zbog bolje preglednosti adresa, unatoč tome što neki blokovi zahtijevaju znatno manje prostora. Na početku adresnog prostora mapirana je 1G DDR memorija. Budući da se radi o 32-bitnim procesorima, ukupan memorijski prostor iznosi 4GB(2^{32}). Dodjeljivanje adresa direktno utječe na realizaciju AXI Interconnect blokova koji su zaduženi za dekodiranje adresa.

Diagram | Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
logicvc_0					
videoData (32 address bits : 4G)					
processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	1G	0x3FFF_FFFF
logiwin_0					
winData (32 address bits : 4G)					
processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	1G	0x3FFF_FFFF
processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
axi_gpio_0	S_AXI	Reg	0x4007_0000	64K	0x4007_FFFF
logicvc_0	s_axi	reg0	0x4000_0000	64K	0x4000_FFFF
logihdr_0	S_AXI	reg0	0x4003_0000	64K	0x4003_FFFF
logii2c_0	s_axi	reg0	0x4006_0000	64K	0x4006_FFFF
logiisp_0	S_AXI	reg0	0x4002_0000	64K	0x4002_FFFF
logiwin_0	s_axi	reg0	0x4001_0000	64K	0x4001_FFFF
mipi_csi2_rx_subsystem_0	csirxss_s_axi	Reg	0x4004_0000	128K	0x4005_FFFF

Slika 5.6: Dodjeljivanje adresnog prostora

5.3. Sinteza i implementacija dizajna

Nakon izrade dijagrama dizajna sustava i dodjeljivanja adresnog prostora pojedinim IP blokovima, sljedeći korak je sinteza. Sinteza pretvara RTL model u listu povezanih standardnih logičkih sklopova iz UNISIM biblioteke. Model dobiven sintezom nije namijenjen za implementaciju već je pogodan za funkcijsku simulaciju te može ukazivati na neke greške i upozorenja koja nikako ne treba ignorirati.
Sljedeći korak poslije sinteze dizajna je implementacija dizajna na ciljanu platformu. Implementacija se sastoji od 3 faze: *translate*, *map* i *place and route*. *Translate* faza pretvara model dobiven sintezom koji se sastoji od komponenata iz UNISIM biblioteke u model koji se sastoji od komponenata iz SIMPRIM biblioteke. Komponente iz SIMPRIM biblioteke sadrže informacije o vremenima potrebnima za prebacivanje stanja. *Map* faza mapira komponente na specifične resurse FPGA sklopa, poput LUT-a, bistabila, blok RAM-a i slično. I dalje nije poznato vrijeme propagacije jer točan razmještaj još nije određen. *Place and Route* faza definira razmještaj i veze komponenata unutar FPGA čipa u odnosu na zahtjeve koje je postavio korisnik, poput odabira fizičkih priključaka i vremenskih ograničenja. U tom trenutku poznato je vrijeme propagacije. Ako jedan od putova ne uspije zadovoljiti postavljena vremenska ograničenja, kao rezultat vraća se greška.
Dakle, za pokretanje implementacije potrebno je definirati potrebne uvjete i ograničenja (eng. *Constraints*) o kojima će ovisiti razmještaj logičkih komponenti unutar čipa. Prvi bitan uvjet je odabir fizičkih priključaka čipa. Odabir fizičkih priključaka izvršava se po shemama ZC706 pločice i Xylon FMC pločice. Osim toga, potrebno je definirati standard I/O priključka i odabir opcija, poput korištenja priteznog otpornika ili diferencijalnog zaključenja za diferencijalne MIPI signale. Slijedi par primjera iz datoteke:

```
set_property IOSTANDARD LVDS_25 [get_ports {imx390_data_in_p_hs[0]}]
set_property DIFF_TERM FALSE [get_ports {imx390_data_in_p_hs[0]}]
set_property PACKAGE_PIN AH28 [get_ports {imx390_data_in_p_hs[0]}]

set_property -dict {PACKAGE_PIN AD25 IOSTANDARD LVCMOS25} [get_ports imx390_clk]

set_property PACKAGE_PIN AJ18 [get_ports i2c_sda_io]
set_property IOSTANDARD LVCMOS25 [get_ports i2c_sda_io]
set_property PULLUP true [get_ports i2c_sda_io]
```

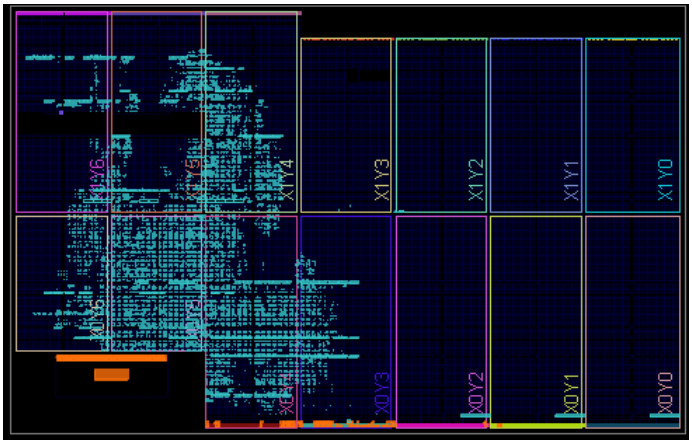
U sustavu postoji više domena koje rade na različitim taktovima. Za pravilan rad ovog sustava nije potrebno da su te domene sinkronizirane. To znači da se putovi koji povezuju te različite domene mogu izostaviti iz vremenske analize i optimizacije jer nije potrebno da se podatci na tom putu prihvate u nekom zadanom ograničenom

vremenu. Takvi putovi nazivaju se lažnima (eng. *false path*). Slijedi primjer definiranja lažnih putova između domena koje rade na 100 MHz i 150MHz.

```
create_generated_clock -name clk_100 [get_pins imx390_eval_i/clk_wiz_100_150_200/inst/mmcm_adv_inst/CLKOUT0]
create_generated_clock -name clk_150 [get_pins imx390_eval_i/clk_wiz_100_150_200/inst/mmcm_adv_inst/CLKOUT1]

set_false_path -from [get_clocks clk_100] -to [get_clocks clk_150]
set_false_path -from [get_clocks clk_150] -to [get_clocks clk_100]
```

Ako signal iz jednog logičkog bloka ide prema drugom logičkom bloku ili vanjskom digitalnom sklopu koji ima određene vremenske zahtjeve, potrebno je definirati i vremenska ograničenja. U našem slučaju video signal, koji izlazi iz čipa, ide do HDMI kontrolera koji zahtjeva da podatak bude stabilan minimalno 0.7 ns (eng. *Setup Time*) prije dolaska ruba takta, i minimalno 1 ns da podatak bude stabilan nakon dolaska ruba takta (eng. *Hold Time*). To je zadano na sljedeći način:

```
set_output_delay -clock [get_clocks clk_148_5] -reference_pin [get_ports pix_clk_o] -min -add_delay -0.700 [
    get_ports {vid_io_data[*]}]
set_output_delay -clock [get_clocks clk_148_5] -reference_pin [get_ports pix_clk_o] -max -add_delay 1.000 [
    get_ports {vid_io_data[*]}]
```



Slika 5.7: Rezultat implementacije

5.4. Izrada programske podrške sustava

Nakon uspješno obavljene sinteze, implementacije i generiranja konfiguracijske datoteke u Vivado-u, potrebno je klikom na "*File->Export->Export Hardware*" izvesti HDF datoteku koja sadrži informacije o dizajnu, poput korištenih IP jezgri i adrese na kojem su mapirane IP jezgre, postavke procesorskog sustava i slično. Te informacije

nužne su za rad softverskog dijela u SDK-u. Osim toga, odabrana je i opcija uključivanja konfiguracijske datoteke unutar HDF datoteke kako bi se iz SDK mogla obavljati konfiguracija FPGA dijela, i s time pokretanje cijelog sustava. Prvo što je potrebno napraviti nakon otvaranja radnog prostora (eng. *workspace*) u SDK-u je upravo uvoz spomenute HDF datoteke. Nakon toga, potrebno je klikom na "*File->New->Board Support Package*" generirati softverski sloj specifičan za sustav opisan u HDF datoteci. Taj softverski sloj sadrži upravljačke programe (eng. *drivers*) i rutine koji omogućavaju izradu aplikacija za odabranu platformu. Na kraju je potrebno klikom na "*File->New->Aplication Project->Empty Project*" napraviti projekt u kojem će se razvijati aplikacija. Za pokretanje aplikacije potrebno je klikom na "*Xilinx->Program FPGA*" konfigurirati FPGA i zatim klikom na "*Run*" pokrenuti sustav. U tom se trenutku pokreće skripta koja inicijalizira procesor na postavke koje su definirane u HDF datoteci, te se izvršna datoteka (dobivena kompajliranjem koda aplikacije) kopira u DDR memoriju iz koje se počne izvršavati. Sve se to odvija preko JTAG sučelja.

5.4.1. Inicijalizacija sustava

Inicijalizacija sustava odvija se redom od zadnjeg elementa u sustavu prema prvom elementu u sustavu. Tako se osigurava da je svaki element sustava inicijaliziran i spreman za prijem podataka prije nego podatci krenu stizati, što potencijalno sprječava razne probleme koji mogu nastati. Tako su redom inicijalizirani: logiCVC, logiWin, logiHDR, logiISP i na kraju sam senzor. MIPI CSI RX nije potrebno softverski inicijalizirati već je nakon konfiguracije FPGA spreman za rad. Prije samog inicijaliziranja komponenti sustava potrebno je inicijalizirati spomenuti I2C prespojnik i HDMI kontroler. To se obavlja pozivom funkcije *basePlatformInit*. Inicijalizacija pojedinih IP jezgri u sustavu obavljena je po uzoru na primjere koje proizvođač nudi zajedno uz *drivere*. Inicijalizaciju senzora provedena je po uputama proizvođača.

```
bool hwInit(HwDescrT * pHwDescr)
{
    bool bInit = false;
    basePlatformInit();

    if (!hwInitLogiCvc(pHwDescr))
        ERROR_PRINT("Error: failed to initialize logiCvc.\n");
    else if (!hwInitLogiWin(pHwDescr))
        ERROR_PRINT("Error: failed to initialize logiWin.\n");
    else if (!hwInitLogiHdr(pHwDescr))
        ERROR_PRINT("Error: failed to initialize logiHdr.\n");
    else if (!hwInitLogiIsp(pHwDescr))
        ERROR_PRINT("Error: failed to initialize logiIsp.\n");
    else if (!hwInitSensor(pHwDescr))
        ERROR_PRINT("Error: failed to initialize  sensor.\n");
    else
        bInit = true;
```

```
    if (!bInit)
        hwDeinit(pHwDescr);

    return bInit;
}
```

Inicijalizacija senzora

Inicijalizacija senzora odvija se putem I2C sabirnice. Nakon paljenja sustava potrebno je resetirati senzor postavljanjem *reset* linije u nisko stanje te je zadržati u tom stanju minimalno 100 us da se osigura uspješno resetiranje senzora. Nakon postavljanja linije za *reset* u visoko stanje senzor ulazi u *Initial1* stanje koje traje 30 ms nakon kojeg automatski pređe u *Standby* stanje. Za vrijeme dok je senzor u *Initial1* stanju nije moguće pisati u registre preko I2C sabirnice. Nakon upisivanja svih potrebnih registara u senzor (za vrijeme *Standby* stanja), upisivanjem 1 u kontrolni registar senzor ulazi u *Initial2* stanje nakon kojeg direktno prelazi u *Active* stanje. Ponovnim pisanjem 0 u kontrolni registar senzor ulazi u *Terminate* stanje nakon kojeg se automatski vraća u *Standby* stanje.

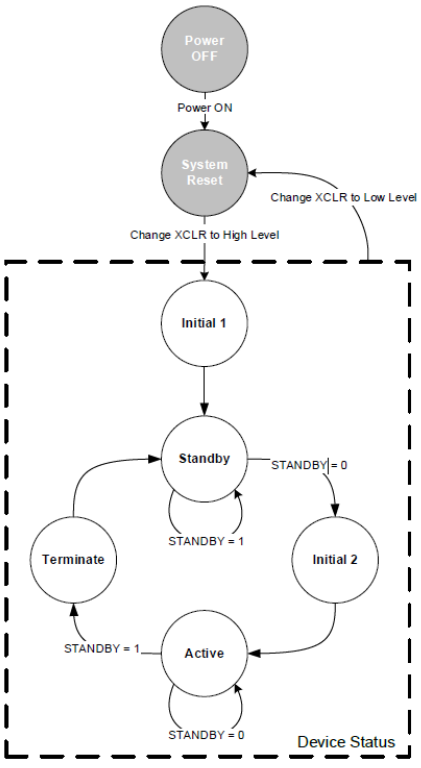


Figure 4-1 Transition of Device Status

Slika 5.8: Inicijalizacija senzora

Za inicijalizaciju senzora realizirane su 4 funkcije: *CAM_SW_Reset, CAM_Init, CAM_Init_Check* i *CAM_START_Active*. Posebno je napravljena struktura podataka koja sadrži parove I2C adrese i vrijednosti koje se upisuju na tu adresu. *CAM_SW_Reset* upisuje 1 u kontrolni registar i tako inicira prebacivanje senzora u *Standby* stanje. *CAM_START_Active* upisuje 0 u kontrolni registar i inicira prebacivanje senzora u *Active* stanje. Funkcija *CAM_Init*, koristeći spomenutu strukturu, upisuje inicijalne vrijednosti na odgovarajuće adrese registara u senzoru. *CAM_Init_Check* čita vrijednosti s istih adresa i uspoređuje ih s inicijalnim vrijednostima strukture. Ako se neka vrijednost razlikuje, to je znak da je možda došlo do greške prilikom inicijalizacije (greška može biti i samo prilikom čitanja). Realizacija funkcija dana je u prilogu.

```
static bool hwInitSensor(HwDescrT * pHwDescr)
{
  bool SInit=false;
  XGpio reset;
  logiI2C_PointerT logiI2C_instanceP;
  logiI2C_instanceP =  logiI2C_init(0, CAM_I2C_CLOCK);

  XGpio_Initialize(&reset,XPAR_GPIO_0_DEVICE_ID);

  XGpio_DiscreteWrite(&reset,1,0x00000000);
  OsMsDelay(1); // min 100us
  XGpio_DiscreteWrite(&reset,1,0x11111111);
  OsMsDelay(40);//min 30 ms

  if(logiI2C_instanceP)
  {
    if(CAM_SW_Reset(logiI2C_instanceP))
      ERROR_PRINT(" I2C fail 0!\n\r");
    else if(CAM_Init(logiI2C_instanceP))
      ERROR_PRINT(" I2C fail 1!\n\r");
    else if(CAM_Init_Check(logiI2C_instanceP))
      ERROR_PRINT(" CAM Init check fail!\n\r");
    else if (CAM_START_Active(logiI2C_instanceP))
      ERROR_PRINT(" I2C fail 2!\n\r");
     else
      SInit=true;
  }
  else
    ERROR_PRINT(" Error init i2c\n\r");

  return SInit;
}
```

5.4.2. AWB i AE

AWB i AE algoritmi izvode se u beskonačnoj petlji. Iz STATS logiISP neprestano se dohvaćaju potrebni podaci na temelju kojih se radi odgovarajuća korekcija slike.
Na boju pojedinog objekta utječu uvjeti osvjetljenja. Ljudsko oko i mozak automatski kompenziraju utjecaj različitih osvjetljenja i zbog toga objekt bijele boje uvijek je bijel bez obzira da li je osvijetljen sunčevim svjetlom, fluorescentnim svjetlom ili nekim drugim svjetlom. S druge strane, senzori trebaju pomoć da bi mogli emulirati taj

proces. Za to je zadužen AWB algoritam koji iz statistike slike prepoznaje kakvi su uvjeti osvjetljenja i na temelju toga radi korekciju slike. Korekcija se radi tako da se mijenjaju vrijednosti na dijagonali matrice CCM bloka u logiISP-u. Postavke za AWB postavljaju se ovisno o specifičnim parametrima senzora.

```
logiIspAwbConfigT  awbConfig;
awbConfig.maxGainRed          = 8.;     // sensor specific maximum red color gain
awbConfig.maxGainGreen        = 8.;     // sensor specific maximum green color gain
awbConfig.maxGainBlue         = 8.;     // sensor specific maximum blue color gain
awbConfig.greenChromaticity   = .39;    // sensor specific green chromaticity value
awbConfig.blackCorrectionRed  = BLC\_RED;  //!< sensor specific red channel black level correction
awbConfig.blackCorrectionGreen = BLC\_GREEN;//!< sensor specific green channel black level correction
awbConfig.blackCorrectionBlue  = BLC\_BLUE; //!< sensor specific blue channel black level correction
```

Iz statistike slike može se također raspoznati koliko je jako osvjetljenje snimanog kadra. Ako je osvjetljenje preveliko, potrebno je smanjivati ekspoziciju kako bi se izbjeglo zasićenje piksela, a ako je premalo, potrebno je povećati ekspoziciju da slika ne bi bila pretamna. Promjena ekspozicije radi se po uzoru na sliku 2.7. Za automatsku regulaciju ekspozicije koristi se programski PID regulator kojem je moguće podešavati parametre i tako dobiti različite brzine odziva. Moguće je postaviti razinu željene svjetline slike i vrijednosti koje definiraju do koje se granice primjenjuje promjena pojačanja, a od koje promjena vremena ekspozicije, kao metoda kontrole ekspozicije. Promjena pojačanja ili vremena eksplozije izvodi se pisanjem odgovarajućih vrijednosti u odgovarajuće registre senzora. Za to je potrebno napraviti funkcije *setGain* i *setExposure* koje će na temelju podataka koje PID regulator izgenerira, izračunati i upisati potrebne vrijednosti u senzor.

```
logiIspAeConfigT  aeConfig;
aeConfig.setpoint               = 0.21;         // constant output set-point
aeConfig.hysteresis             = 0.01;         // regulator hysteresis
aeConfig.kP                     = 87.5;         // regulator P constant value
aeConfig.kI                     = 0.0725;       // regulator I constant value
aeConfig.kD                     = 1250.0;       // regulator D constant value
aeConfig.logicalValueMaxErr     = 8.0;          // Max. proportional error step value
aeConfig.logicalValueMaxIntErr  = 16.0;         // Max. integral error value
aeConfig.logicalValueRange      = 300.0;        // range of the logical variable value
aeConfig.logicalValueExpoRange  = 40.0;        // part of the logicalValueRange that belongs to exposure
aeConfig.logicalValueGainRange  = 260.0;        // part of the logicalValueRange that belongs to
                                                  //gain(logicalValueRange - logicalValueExpoRange)
aeConfig.sensorMaxGain          = 24.0;         // sensor maximum gain
aeConfig.sensorMinGain          = 10.0;         // sensor minimum gain
aeConfig.sensorMaxExpo          = 11.0;           // sensor maximum exposure
aeConfig.sensorMinExpo          = 2.0;          // sensor minimum exposure
aeConfig.pCameraInstance        = NULL;         // pointer to the camera instance data
aeConfig.setGain                = &setGain;     // callback function for setting a gain
aeConfig.setExposure            = &setExposure; // callback function for setting a exposure
```

Funkcija *setGain* kao parametar prima vrijednost koju izračuna PID regulator i ta vrijednost predstavlja pojačanje u decibelima. U registar senzora potrebno je upisati vrijednost koja je jednaka vrijednosti pojačanja u decibelima podijeljenoj s 0.3.

```
static void setGain(float gain)
{
  logiI2C_PointerT logiI2C_instanceP;
  logiI2C_instanceP =  logiI2C_init(0, CAM_I2C_CLOCK);
  OsU8 aeGain=(OsU8) (gain/0.3);
  CAM_WriteReg(logiI2C_instanceP, 0x0018, aeGain);
}
```

Funkcija *setExposure* kao parametar prima vrijednost koja predstavlja vrijeme trajanja ekspozicije u milisekundama. S druge strane, senzor vrijeme ekspozicije računa kao broj poslanih linija prethodnog okvira pomnožen s periodom jedne linije. Taj period ovisi o frekvenciji slanja okvira, a u slučaju 30 okvira po sekundi iznosi 29.6 us. U funkciji je potrebno iz podataka o trajanju ekspozicije i perioda linije izračunati broj linija na koje senzor mora čekati do početka ekspozicije. Veći broj linja znači kraće vrijeme ekspozicije. Senzor šalje ukupno 1114 linija od kojih 1096 predstavlja piksele slike sa senzora, a ostale linije služe za prijenos dodatnih informacija koje mogu poslužiti prilikom obrade. Za zapis podatka o broju linija potrebna su dva 8-bitna registra, pa je potrebno zapisati najznačajnije bitove na donju, a manje značajne na gornju adresu. Budući da se sada upisuju vrijednosti dva registra, potrebno je osigurati da senzor prihvati njihovu promjenu u istom trenutku. Zato se prije upisivanja u te registre upiše vrijednost 1 u REG_HOLD registar koji blokira prihvat promjena registara. Tek kada se upišu vrijednosti u oba registra može se omogućiti prihvat promjena upisivanjem 0 u REG_HOLD registar.

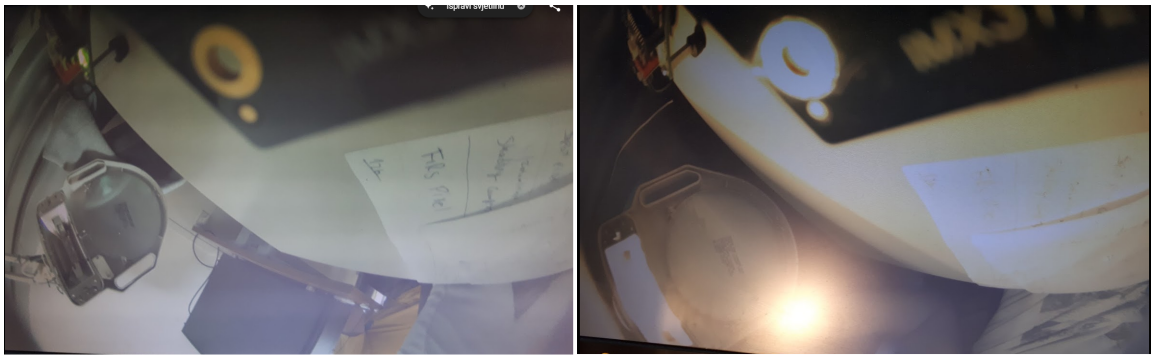
```
static void setExposure(float expo)
{
  OsU16  aeExpo ;
  OsU8  data_buf[2];
  float ExpLines;
  logiI2C_PointerT logiI2C_instanceP;
  logiI2C_instanceP =  logiI2C_init(0, CAM_I2C_CLOCK);

  ExpLines= expo / 0.0296;
  aeExpo= 1114-(OsU16) ExpLines;

  data_buf[0] = (aeExpo >> 8);
  data_buf[1] = aeExpo & 0xff;
  CAM_WriteReg(logiI2C_instanceP, 0x0008, 1); //reg_hold on
  CAM_WriteReg(logiI2C_instanceP, 0x000D, data_buf[0]);
  CAM_WriteReg(logiI2C_instanceP, 0x000C, data_buf[1]);
  CAM_WriteReg(logiI2C_instanceP, 0x0008, 0); //reg_hold of
}
```

5.4.3. Demonstracijska aplikacija

Za demonstraciju funkcionalnosti sustava napisana je sljedeća aplikacija. Nakon inicijalizacije sustava svi logički blokovi koji utječu na sliku postavljani su u zaobilazni

način rada, osim CFA bloka iz logiISP-a koji odrađuje pretvorbu iz RAW formata u RGB format. Nakon slanja znaka UART-om, aktiviraju se pojedini blokovi jedan po jedan. Nakon što se aktiviraju svi blokovi, posljednja se aktivira funkcija koja u petlji odrađuje AE i AWB algoritme.

```
int main(){
  if (!hwInit(&gHwDescr))
      {
          OsPrintfError("Error: failed to initialize hardware!\n");
          while(1);
      }
    getchar();
    logiISP_BypassModules(gHwDescr.aLogiIspPtr[0], GAMMA , CLEAR);
    getchar();
    logiHdr_BypassModules(gHwDescr.aLogiHdrPtr[0], HDR_DRC, HDR_CLEAR);
    getchar();
    logiHdr_BypassModules(gHwDescr.aLogiHdrPtr[0], HDR_BRE, HDR_CLEAR);
    getchar();
  awbAeThread(&gHwDescr);
  return 0;
}
```



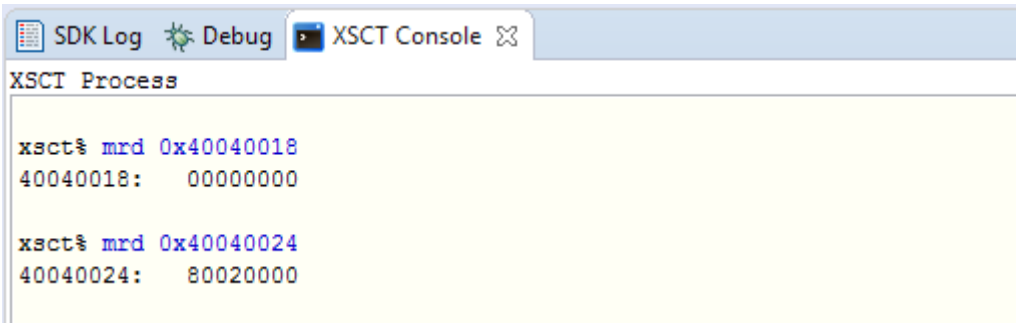
Slika 5.9: Slika sa senzora nakon paljenja svih modula pri različitim uvjetima osvjetljenja

5.4.4. Pokretanje sustava sa SD kartice

Do sada se konfiguracija FPGA, inicijalizacija procesorskog sustava, kopiranje izvršnog koda u memoriju i pokretanje sustava radilo iz SDK preko JTAG sučelja. Da bi se pokretanje sustava moglo izvršavati sa SD kartice, potrebno je imati program koji će sve te poslove sam odraditi pokretanjem iz SD kartice. Takav program naziva se FSBL (eng. *First Stage Bootloader*). Klikom na "*File->New->Aplication Project-> Zynq FSBL*" generira se takav program. Zatim, desnim klikom na aplikacijski projekt i odabirom "*Create Boot Image*" otvori se prozor u kojem je potrebno redom navesti putove datoteka: FSBL-a, konfiguracijske datoteke za FPGA i izvršne datoteke. Klikom na "*Create Image*" generira se datoteka koja se prebacuje na SD karticu. SD kartica se umetne u SD utor i nakon paljenja napajanja pokreće se sustav.

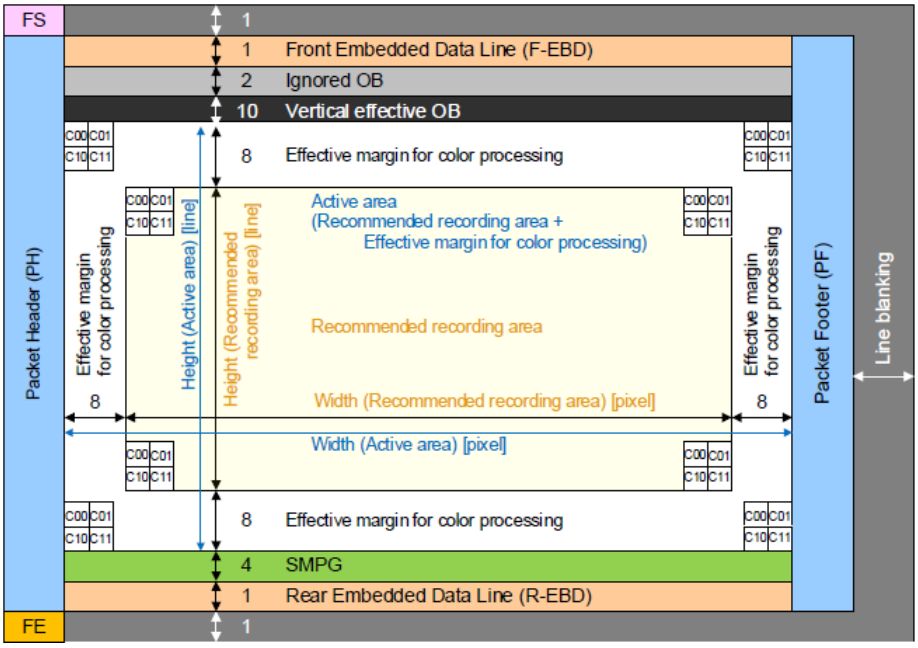
5.5. Otkrivanje i uklanjanje grešaka u radu sustava

Ako sustav nakon prvog paljenja ne radi kako bi trebao, potrebno je započeti proces otkrivanja grešaka u sustavu. Suprotno od inicijalizacije sustava, otkrivanje grešaka započinje od prvog elementa u sustavu. Svaki malo kompliciraniji IP sadrži dio statusnih i kontrolnih registara koji služe za otkrivanje grešaka u sustavu. Koristeći XSCT konzolu, koja je sastavni dio SDK, mogu se čitati i upisivati vrijednosti registara za vrijeme rada sustava. To se odvija preko JTAG sučelja. Za upisivanje neke vrijednosti u željeni registar koristi se naredba *mwr*, a za čitanje *mrd* naredba.



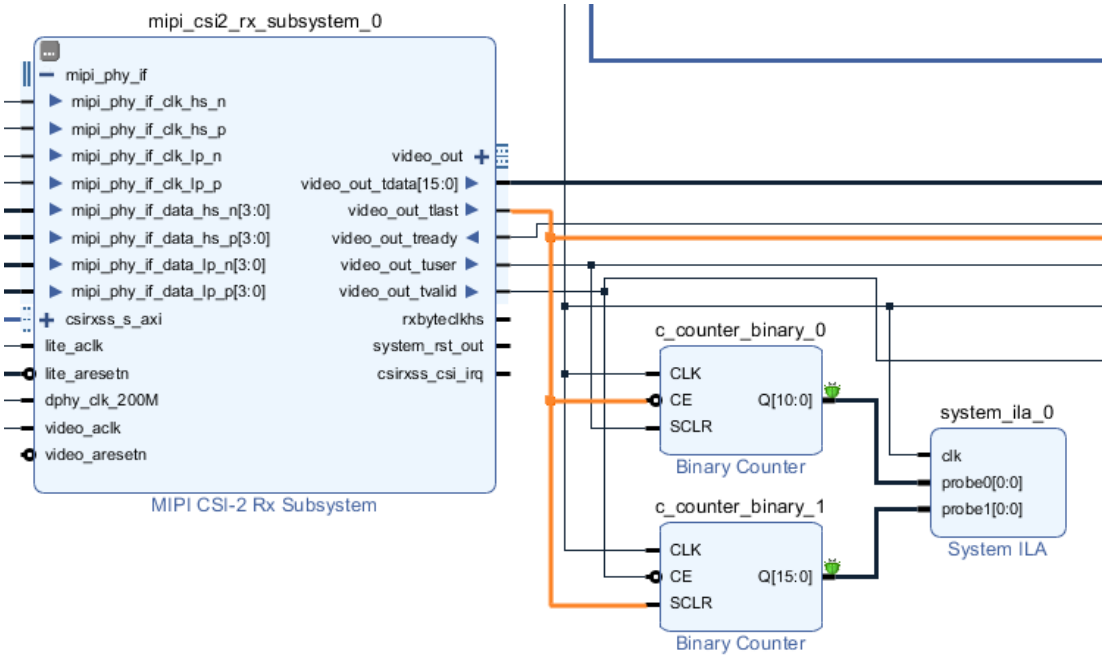
Slika 5.10: Korištenje XSCT konzole

Zbog nepravilnog rada sustava nakon prvog paljenja, započelo se s procesom otkrivanja grešaka od MIPI CSI-2 prijemnika. Čitanjem registara prijemnika zaključeno je da nema nikakvih grešaka u prijenosu paketa MIPI sučeljem te da podatci uredno stižu od senzora do prijemnika. Čitanjem registara sljedećeg elementa u sustavu, logiISP-a, otkrivena je greška u sustavu. Detektirana je greška koja označuje da je signal kraja linije (*tlast*) stigao pre kasno, što bi značilo da je došao veći broj piksela u liniji nego što je očekivano. Naime, prilikom inicijalizacije logiISP-a potrebno je definirati veličinu okvira. Po dokumentaciji senzora može se zaključiti da se okvir sastoji od 1114 linija gdje svaka linija ima 1936 piksela. Osim samih podataka koji predstavljaju snimljenu sliku (1096 linija), šalju se i neki dodatni podatci koje logiISP ne podržava i nisu bitni za njegov rad. Radi se o linijama podataka koji mogu poslužiti kao referenca za crnu boju (pikseli skriveni od vanjske svjetlosti) ili mogu sadržavati podatke o rednom broju okvira i slično.



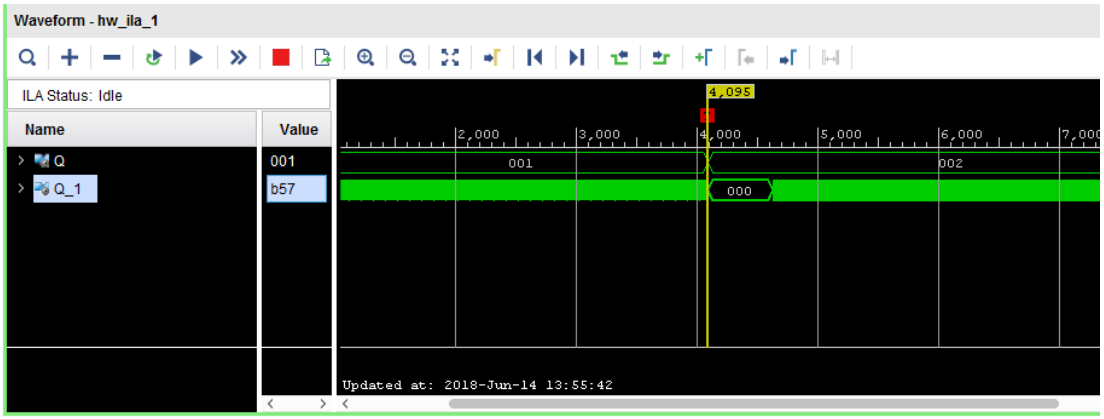
Slika 5.11: Struktura okvira

Budući da je detektirano da su pristigle linije duže od 1936 piksela, kako bi se provjerilo radi li se o svim linijama ili samo o dijelu linija, sustav je ponovno pokrenut, ali ovaj put je prilikom inicijalizacije logiISP-a zadana veličina okvira od 1114x1937. Tada je detektirana greška koje označuje da je signal kraja linije došao pre kasno, ali i greška koja označuje da je signal linije došao pre rano. Iz toga se može zaključiti da je dio linija koje senzor šalje zaista duljine 1936 piksela,a dio linija koje senzor šalje duljine veće od 1937 piksela. Da bi točno provjerili strukturu okvira koji dolaze, potrebno je analizirati unutarnje signale korištenjem ILA (eng. *Integrated Logic Analyzer*) IP jezgre. ILA IP koristi unutarnju raspoloživu memoriju od FPGA za spremanje snimljenih uzoraka. Po zahtjevu korisnika snimljeni uzorci mogu se putem JTAG sučelja poslati do korisnikovog računala i prikazati u Vivado prozoru. Kako bi točno provjerili broj linija i broj piksela u pojedinim linijama, iskorištena su 2 brojača spojena kao na slici 5.10. Brojač koji broji broj piksela u liniji inkrementira se svaki put kada se pojavi signal koji označava da je piksel poslan (*tvalid*), a resetira se svaki put kada se pojavi signal koji označava kraj linije (*tlast*). Brojač koji broji broj linija u okviru inkrementira se pojavom signala koji označava da je došao kraj linije, a restira se pojavom signala koji označava da je došao početak novog okvira (*tuser*).



Slika 5.12: Debug

Budući da ILA IP može spremati ograničen broj podataka, nije moguće prikazati sve linije jednog okvira u jednom prozoru, ali s različitim postavkama okidača moguće je prikazati svaku liniju zasebno. Tako na primjer, za prikaz prve i druge linije postavljen je okidač na pojavu vrijednosti 2 kod brojača koji broji linije. Analizom rezultata brojača piksela za svaku liniju zaključeno je da prvih 13 i zadnjih 5 linija sadrži 2903 piksela (b57 heksadecimalno), dok sve ostale linije sadrže 1936 piksela.



Slika 5.13: ILA prozor

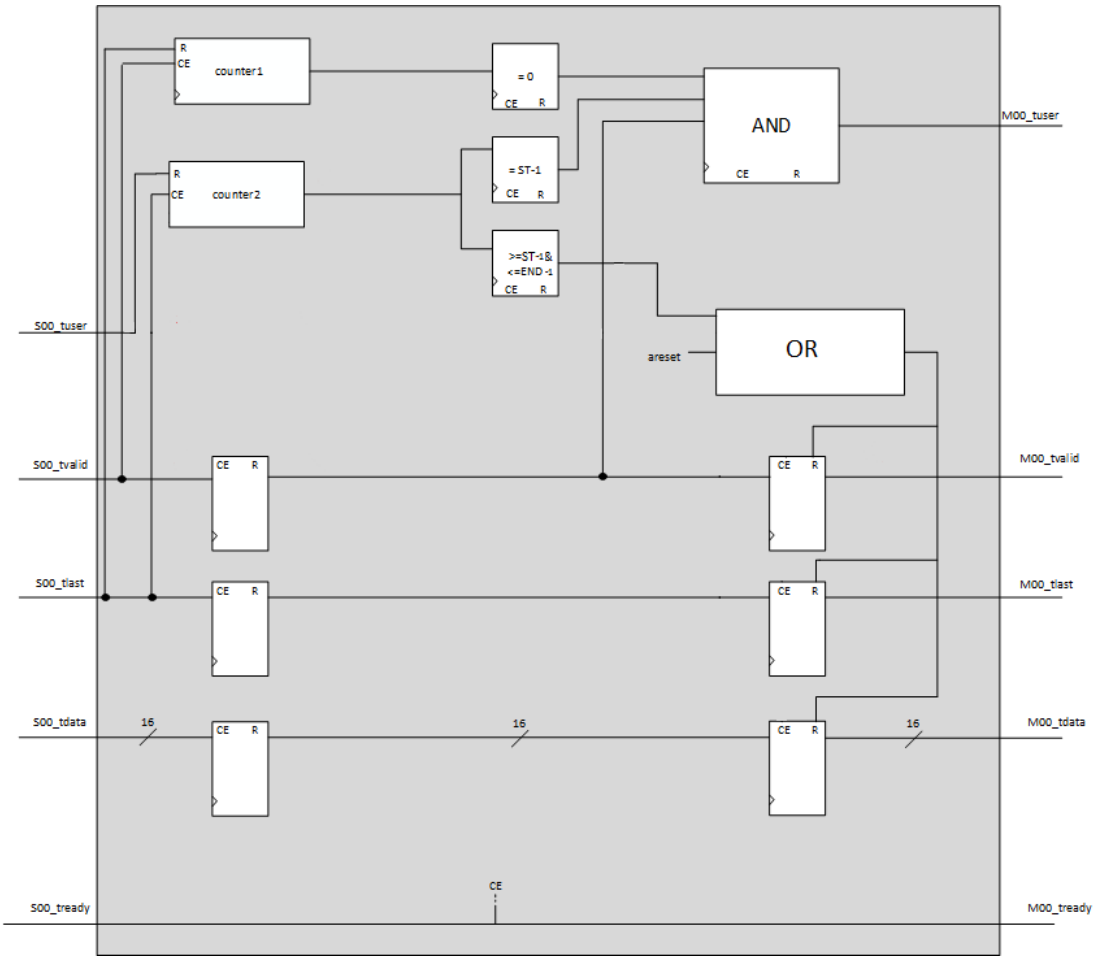
Linije od 1936 piksela sadrže korisnu informaciju o slici, dok ostale linije duže od 1936 piksela sadrže dodatne informacije koje ionako logiISP-u ne podržava. Budući da je logiISP dizajniran tako da zahtjeva stabilne podatke na ulazu (sve linije imaju

isti broj piksela), potrebno je na neki način ukloniti višak linija kako bi logiISP mogao normalno raditi. Za tu svrhu razvijen je poseban logički blok.

5.5.1. AxisCropper

AxisCropper logički je blok koji prima podatke slike u AXI-Stream formatu i prosljeđuje ih, uz mogućnost odbacivanja proizvoljnog broja linja s početka i kraja okvira odabirom generičkih vrijednosti ST i END. Axi-Stream sučelje sastoji se od 4 kontrolne linije i određenog broja podatkovnih linija *tdata* (u ovom slučaju 16). Kontrolni signal *tlast* označava kraj linije, a signal *tuser* označava početak novog okvira. Ti signali dolaze istovremeno s prvim odnosnom zadnjim pikselom. U trenutku slanja piksela također je aktivan i kontrolni signal *tvalid*. Signal *tready* jedini je ulazni signal kojeg aktivira prijemni blok kada je spreman za prijem podataka.
Unutar logičkog bloka postoji brojač koji broji pristigle linije unutar jednog okvira. Brojač se inkrementira na signal kraja linije, a resetira na signal početka novog okvira. Dok je vrijednost brojača veća ili jednaka vrijednosti ST i manja ili jednaka vrijednosti END, podatci s ulaza propuštaju se na izlaz. Kada je vrijednost brojača izvan tih granica, podatci se ne propuštaju na izlaz.
Kontrolni signal na zadnjem bistabilu, koji definira da li se podatci propuštaju ili ne, kasni za jedan korak od samog podatka. To mora biti tako jer bi inače blokirao posljednji piksel iz posljednje linije okvira. Na ovaj način točno po izlasku zadnjeg piksela postaje zabranjeno prosljeđivanje na zadnjem bistabilu. Međutim, sada kada na ulazu stigne prva linija sljedećeg okvira (*tuser* resetira brojač), zbog kašnjenja kontrolnog signala prvi piksel ne bi mogao izaći, iako se možda ne želi prva linija odbaciti. Zbog jednostavnosti postavljeno je da je generička konstanta ST minimalno vrijednosti 2, tj. da se prva linija mora odsjeći, što je ionako slučaj. Budući da je prva linija odbačena, potrebno je generirati novi *tuser* signal za vrijeme slanja prvog piksela prve željene linije. Da bi se omogućila detekcija tog uvjeta potreban je još jedan brojač koji broji piksele u liniji. Taj brojač inkrementira se na *tvalid* signal, a resetira na *tlast* signal. Potreban je još jedan komparator da bi se detektirala prva željena linija. Komparator se zatim spaja na brojač koji broji linije u okviru. Kada je zadovoljen uvjet da je vrijednost prvog brojača jednaka rednom broju prve željene linije (minus 1), te da je vrijednost drugog brojača jednaka 0 (prvi piksel) i da je *tvalid* signal zakašnjen za jedan korak aktivan, u sljedećem koraku generira se *tuser* signal koji predstavlja novi početak okvira.
U trenutku kada *tready* prestane biti aktivan mora se sačuvati stanje bloka. Zato se

tready signal spaja na CE (eng. *Clock enable*) registara.

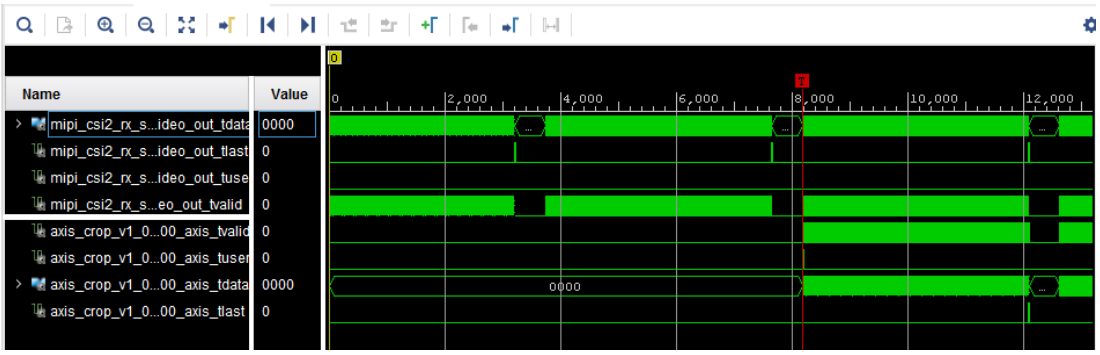


Slika 5.14: Axis_cropper

AxisCropper izrađen je korištenjem VHDL programskog jezika. Slijedi odsječak iz koda koji generira *tuser* signal. Cijeli kod nalazi se u prilogu.

```
process (aclk) is
begin
  if rising_edge(aclk) then
     if (areset='0') then
      m00_axis_tuser  <='0';
    elsif (m00_axis_tready = '1')  then
      if (compare2='1' and compare3='1' and m_tvalid1='1') then
      m00_axis_tuser  <='1';
      else
      m00_axis_tuser  <= '0';
    end if;
      end if;
  end if;
end process;
```

Na slici 5.14 može se vidjeti izlaz iz AxisCroppera oko mjesta gdje počne propuštanje linija okvira.



Slika 5.15: Izlaz iz Axis-cropper-a

Cijeli postupak izrade nove IP jezgre izvodi se u zasebnom RTL projektu. Po završetku opisa dizanja potrebno je klikom na "*Tools->Create and Package New IP*" kreirati i zapakirati dizajn u novu IP jezgru koja će se moći koristiti u drugim projektima.

6. Zaključak

Za realizaciju sustava za automobilsku kameru korišten je Xilinx Zynq-7000 SoC koji se sastoji od procesorskog sustava i FPGA sklopa integriranih na istom čipu. FPGA pogodan je za realizaciju logičkih blokova koji obavljaju visoko paralelizirane zadatke koji se ponavljaju velik broj ciklusa i samo su povremeno redefinirani. Upravo zbog toga, unutar FPGA sklopa realiziran je cijeli cjevovod za obradu sirove slike koja pristiže sa senzora te sustav za prikaz slike na monitoru. Cjevovod za obradu slike čine logiISP i logiHDR IP blokovi, a sustav za prikaz slike logiWIN i logiCVC IP blokovi. Procesorski sustav optimiziran je za izvršavanje velikog broja raznolikih operacija s mogućnošću brze promjene konteksta, što je pogodno za pokretanje korisničke aplikacije. Iz korisničke aplikacije mogu se paliti i gasiti pojedini moduli IP jezgri koji rade različitu obradu nad slikom, što utječe na konačan rezultat. Osim toga, na procesoru se izvršavaju AWB i AE algoritmi koji rezultiraju promjenom koeficijenata cjevovoda za obradu slike ili samog senzora, što utječe na rezultat snimanja.

Problem koji nastaje u sustavu zbog nekompatibilnosti logiISP-a i podataka koje šalje senzor lako je rješiv upravo zbog fleksibilnosti FPGA sklopa. U FPGA sklopu realiziran je logički blok koji je spojen prije logiISP-a i uklanja nekompatibilne linije što omogućava pravilan rad sustava. Da je sustav bio realiziran kao ASIC čip, sustav bi bio beskoristan te bi se trebao izraditi potpuno novi čip.

Zbog ograničenja logiISP IP jezgre koja može primiti maksimalno 12 bita na ulazu, odbačen je relativno velik dio bitova što rezultira smanjenjem dinamičkog raspona slike.

LITERATURA

Automotive camera solutions, 2018. URL https://www.ambarella.com/products/automotive-cameras.

Aldec. Introduction to axi protocol. URL https://www.aldec.com/en/company/blog/122--introduction-to-axi-protocol.

Love Gupta. All you need to know about mipi d'phy rx. 2015.

Marcus Hawkins. The exposure triangle: aperture, shutter speed and iso explained, 2017. URL https://www.techradar.com/how-to/photography-video-capture/cameras.

Texas Instruments. *PCA9548A Low Voltage 8-Channel I 2C Switch With Reset*, 2015. URL http://www.ti.com/lit/ds/symlink/pca9548a.pdf.

J.Ivanović. *logiISP user's manual*, 2016.

R. Končurat. *logiWIN user's manual*, 2017.

Miroslav Rožić. *Senzori slike*. Mikroprojekt, 2018.

LINDSAY SILVERMAN. Setting white balance. URL https://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/setting-white-balance.html.

Sony. Sony commercializes the industry's first automotive 2mp high-sensitivity cmos image sensor with led flicker mitigation and hdr. 2017a. URL https://blog.sony.com/2017/04/.

Sony. Sony commercializes the industry's first*1 high-sensitivity cmos image sensor for automotive cameras, delivering simultaneous led flicker mitigation and high-quality hdr shooting. 2017b. URL https://www.sony.net/SonyInfo/News/Press/201704/17-034E/index.html.

Sony. *IMX390CQV-W-technical_datasheet_E_Rev.0.1.3*, 2018a.

Sony. *IMX390_Exposure_Control_Guide20170510*, 2018b.

Sony. *IMX390_EVB-SDB-C_Schematics_E_Rev.0.1*, 2018c.

Sony. *IMX390_ApplicationNote-E-Rev.0.1.0*, 2018d.

S.Opačić. *logiHDR user's manual*, 2016.

Xilinx wiki. *FSBL*. URL http://www.wiki.xilinx.com/FSBL.

Wikipedia. High-dynamic-range imaging, 2018. URL https://en.wikipedia.org/wiki/High-dynamic-range_imaging.

Xilinx. *ZC706 Evaluation Board for the Zynq-7000 XC7Z045 All Programmable SoC User Guide*, 2016. URL https://www.xilinx.com/support/documentation/boards_and_kits/zc706/ug954-zc706-eval-board-xc7z045-ap-soc.pdf.

Xilinx. *Zynq-7000 All Programmable SoC Data Sheet*, 2017. URL https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.

Xilinx. *MIPI CSI-2 Receiver Subsystem v3.0*, 2018. URL https://www.xilinx.com/support/documentation/ip_documentation/mipi_csi2_rx_subsystem/v3_0/pg232-mipi-csi2-rx.pdf.

Xylon. Face detection and tracking based on xilinx zynq-7000 all programmable soc. URL https://www.logicbricks.com/Solutions/Xylon-Face-Detection-Tracing-on-Xilinx-Zynq-SoC.aspx.

Dodatak A
AxisCropper VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity axis_crop_v1_0 is
  generic (
    START_LINE           : integer range 2 to 1300    :=2;
    END_LINE             : integer range 3 to 1300    := 1111;
    -- Parameters of Axi Bus Interface
    C_AXIS_TDATA_WIDTH   : integer := 16
  );
  port (
        aclk         : in std_logic;
        areset         : in std_logic;
    -- Ports of Axi Slave Bus Interface S00_AXIS
    s00_axis_tready      : out std_logic;
    s00_axis_tdata     : in std_logic_vector(C_AXIS_TDATA_WIDTH-1 downto 0);
    s00_axis_tuser     : in std_logic;
    s00_axis_tlast     : in std_logic;
    s00_axis_tvalid      : in std_logic;
    -- Ports of Axi Master Bus Interface M00_AXIS
    m00_axis_tvalid      : out std_logic;
    m00_axis_tdata     : out std_logic_vector(C_AXIS_TDATA_WIDTH-1 downto 0);
    m00_axis_tuser     : out std_logic;
    m00_axis_tlast     : out std_logic;
    m00_axis_tready      : in std_logic
  );
end axis_crop_v1_0;

architecture arch_imp of axis_crop_v1_0 is

signal counter     : std_logic_vector(11 downto 0):=(others=>'0');
signal counter2    : std_logic_vector(11 downto 0):=(others=>'0');
signal compare     : std_logic:='0';
signal compare2    : std_logic:='0';
signal compare3    : std_logic:='0';
signal m_tdata1      : std_logic_vector(C_AXIS_TDATA_WIDTH-1 downto 0);
signal m_tvalid1   : std_logic;
signal m_tuser1    : std_logic;
signal m_tlast1    : std_logic;


begin
s00_axis_tready <= m00_axis_tready;

process (aclk) is
begin
  if rising_edge(aclk) then
    if (areset='0' or s00_axis_tuser='1') then
      counter <= (others => '0');
    elsif (s00_axis_tlast= '1' and m00_axis_tready = '1') then
```

```
      counter <= counter +1;
    end if;
  end if;
end process;

process (aclk) is
begin
  if rising_edge(aclk) then
    if (areset='0') then
      compare <= '0';
    elsif (m00_axis_tready = '1')  then
      if ((counter >= (START_LINE -1)) and (counter <= (END_LINE - 1)))then
        compare<='1';
      else
        compare<='0';
      end if;
    end if;
  end if;
end process;

process (aclk) is
begin
  if rising_edge(aclk) then
    if (areset='0') then
      m_tdata1  <= (others=>'0');
      m_tvalid1 <= '0';
      m_tuser1  <='0';
      m_tlast1  <= '0';
      elsif (m00_axis_tready = '1') then
         m_tdata1       <= s00_axis_tdata;
      m_tvalid1 <= s00_axis_tvalid;
      m_tuser1  <= s00_axis_tuser;
      m_tlast1  <= s00_axis_tlast;
      end if;
  end if;
end process;



process (aclk) is
begin
  if rising_edge(aclk) then
    if (areset='0' or compare = '0') then
      m00_axis_tdata  <= (others=>'0');
      m00_axis_tvalid <= '0';
      m00_axis_tlast  <= '0';
      elsif (compare = '1' and m00_axis_tready = '1') then
             m00_axis_tdata <= m_tdata1;
      m00_axis_tvalid <= m_tvalid1;
      m00_axis_tlast  <= m_tlast1;
      end if;
  end if;
end process;

-- generating new start of frame


process (aclk) is
begin
  if rising_edge(aclk) then
    if (areset='0') then
      compare2 <= '0';
    elsif (m00_axis_tready = '1')  then
      if (counter = START_LINE-1)then
        compare2 <= '1';
      else
        compare2 <= '0';
      end if;
    end if;
  end if;
end process;

process (aclk) is
```

```
begin
  if rising_edge(aclk) then
    if (areset='0' or s00_axis_tlast='1') then
      counter2 <= (others => '0');
    elsif (s00_axis_tvalid= '1' and m00_axis_tready = '1') then
      counter2 <= counter2 +1;
    end if;
  end if;
end process;

process (aclk) is
begin
  if rising_edge(aclk) then
    if (areset='0') then
      compare3 <= '0';
    elsif (m00_axis_tready = '1')  then
      if (counter2 = 0)then
        compare3 <= '1';
      else
        compare3 <= '0';
      end if;
    end if;
  end if;
end process;



process (aclk) is
begin
  if rising_edge(aclk) then
     if (areset='0') then
      m00_axis_tuser  <='0';
    elsif (m00_axis_tready = '1')  then
      if (compare2='1' and compare3='1' and m_tvalid1='1') then
      m00_axis_tuser  <='1';
      else
      m00_axis_tuser  <= '0';
    end if;
      end if;
  end if;
end process;

end arch_imp;
```

Dodatak B

imx390_init.c

```
#include<stdio.h>
#include<stdint.h>
#include "imx390_init.h"


int IMX390ES_FHD_NML_SP1_H_12_M4_30_27M_V126[][2] = {
  ...
  ...
};


int CAM_ReadReg(logiI2C_PointerT logiI2C_instanceP, OsU16 addr, OsU8 *data)
{
  OsU8    data_buf[3];

  data_buf[0] = (addr >> 8);
  data_buf[1] = addr & 0xff;
  return logiI2C_writeRead(logiI2C_instanceP, CAM_I2C_SLAVE_ADDR, data_buf, 2, data, 1,LOGII2C_I2C_DATA_STOP_MSK_E
      );
}

int CAM_WriteReg(logiI2C_PointerT logiI2C_instanceP, OsU16 addr, OsU8 data)
{
  OsU8    data_buf[3];

  data_buf[0] = (addr >> 8);
  data_buf[1] = addr & 0xff;
  data_buf[2] = data;
  return logiI2C_write(logiI2C_instanceP, CAM_I2C_SLAVE_ADDR, data_buf, 3, LOGII2C_I2C_DATA_STOP_MSK_E);
}

int CAM_SWReset(logiI2C_PointerT logiI2C_instanceP)
{
  // reg: 0000 - bit0 -> 0 - operating, 1 - reset
  OsU16 addr = 0x0000;
  OsU8  data = 0x01;
  OsU8  data_buf[3];

  data_buf[0] = (addr >> 8);
  data_buf[1] = addr & 0xff;
  data_buf[2] = data;
  return logiI2C_write(logiI2C_instanceP, CAM_I2C_SLAVE_ADDR, data_buf, 3, LOGII2C_I2C_DATA_STOP_MSK_E);
}

int CAM_Init(logiI2C_PointerT logiI2C_instanceP)
{
  int i, ret=0;
  OsU8 i2c_data;

  for(i=0;;i++)
  {
    if(IMX390ES_FHD_NML_SP1_H_12_M4_30_27M_V126[i][0] != 0xDEAD)
    {
```

```
      i2c_data = IMX390ES_FHD_NML_SP1_H_12_M4_30_27M_V126[i][1];
      ret |= CAM_WriteReg(logiI2C_instanceP, IMX390ES_FHD_NML_SP1_H_12_M4_30_27M_V126[i][0], i2c_data);
    }
    else break;
  }

  return ret;
}

int CAM_Init_Check(logiI2C_PointerT logiI2C_instanceP)
{
  int i, ret=0;
  OsU8 data;
  for(i=0;;i++)
  {
    if(IMX390ES_FHD_NML_SP1_H_12_M4_30_27M_V126[i][0] != 0xDEAD)
        {
          ret |= CAM_ReadReg(logiI2C_instanceP, IMX390ES_FHD_NML_SP1_H_12_M4_30_27M_V126[i][0], &data);
          if (data != IMX390ES_FHD_NML_SP1_H_12_M4_30_27M_V126[i][1])
            return 1;

        }
        else break;
  }
  return ret;
}

int CAM_START_Active(logiI2C_PointerT logiI2C_instanceP)
{
  // reg: 0000 - bit0 -> 0 - operating, 1 - reset
  OsU16 addr = 0x0000;
  OsU8  data = 0x00;
  OsU8  data_buf[3];

  data_buf[0] = (addr >> 8);
  data_buf[1] = addr & 0xff;
  data_buf[2] = data;
  return logiI2C_write(logiI2C_instanceP, CAM_I2C_SLAVE_ADDR, data_buf, 3, LOGII2C_I2C_DATA_STOP_MSK_E);
}
```

Sustav za automobilsku kameru visokog dinamičkog opsega temeljen na FPGA sklopu

Sažetak

Izrađen je sustav za prihvat i obradu digitalne slike s dvodimenzionalnog senzora Sony IMX390 koji podržava oslikavanje u visokom dinamičkom opsegu. Za razvoj sustava korištena je ZC706 razvojna pločica za Zynq-7000 SoC porodicu sklopova. Zynq-7000 sastoji se od FPGA sklopa i ARM procesorskog sustava integriranih na istom čipu. U FPGA dijelu realiziran je cijeli cjevovod za obradu slike i sustav za prikaz slike na monitoru. Na procesorskom sustavu izvršava se demonstracijska aplikacija koja sekvencijalno uključuje pojedine module cjevovoda u svrhu prikazivanja utjecaja pojedinog modula. Objašnjen je proces otkrivanja greški u radu sustava. Prikazana je mogućnost jednostavne modifikacije sustava zbog fleksibilnosti koje pruža FPGA sklop.

Ključne riječi: FPGA, Zynq, IMX390, automobilska kamera, visoki dinamički opseg

FPGA Based System for High Dynamic Range Automotive Camera

Abstract

A system for receiving and processing digital images from the two-dimensional sensor Sony IMX390, that supports high-dynamic imaging, is developed. System is developed using Zc706 development board for Zynq-7000 SoC family of integrated circuits. The Zynq-7000 consists of FPGAs and ARM processors integrated in the same chip. In the FPGA section, the entire image processing pipeline and display system are realized. The processor system executes a demonstration application that sequentially turns on individual piping modules, for the purpose of displaying the influence of a particular modules. The error detection process in the system operation is explained. The possibility of simple system modification due to the flexibility provided by the FPGA circuit is demonstrated.

Keywords: FPGA, Zynq, IMX390, automotive camera, high dynamic range