**Facial recognition**

*Dokumentacija za predmet: Projektovanje složenih digitalnih sistema*

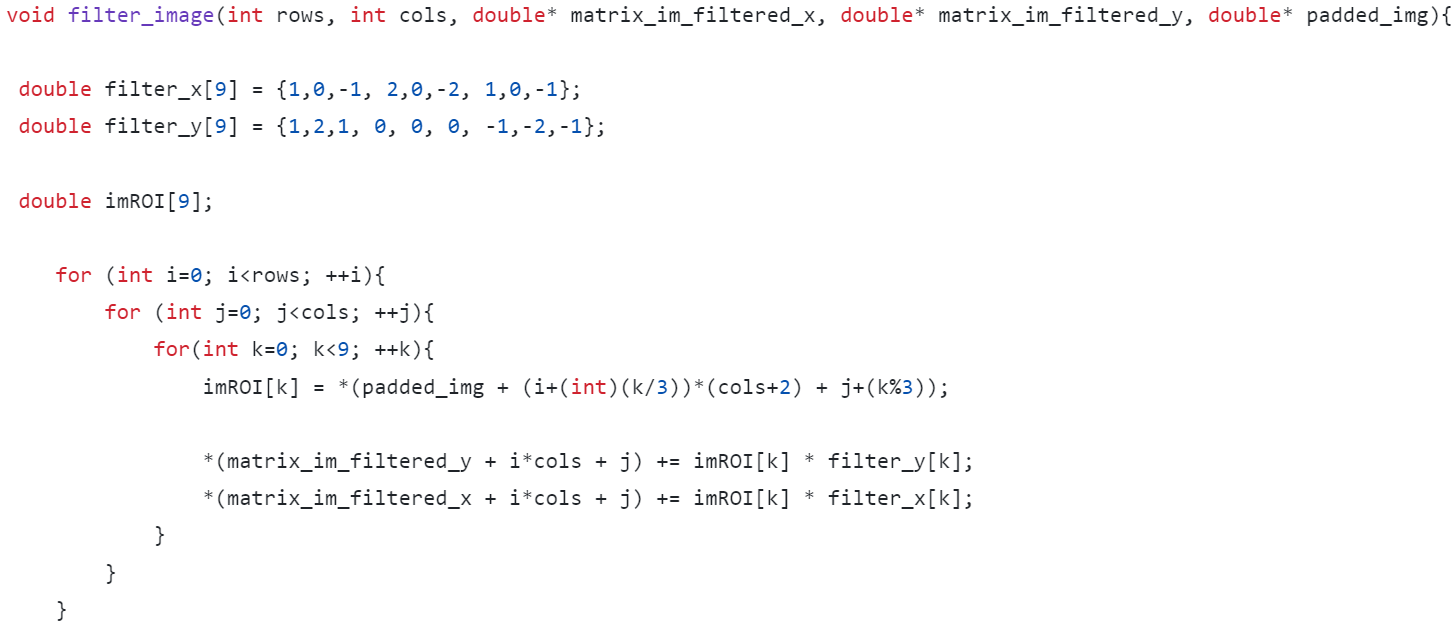
1. Opis algoritma

Ideja projekta je prepoznavanje ljudskog lica na fotografiji. Fotografija se prosleđuje programu, a on iscrtava okvir oko detektovanog lica. Centralni dio algoritma predstavlja filtriranje slike (po x-osi i y-osi) koje podrazumijeva množenje manjih dijelova slike sa filterima. Naime, dimenzije filtera su 3x3, te se svaki komad slike dimenzija 3x3 množi sa njime. Šablon se ponavlja sve dok se ne prođe kroz čitavu sliku pomjerajući se prvo po x-osi za 1 piksel, a nakon obrađenog reda za 1 piksel po y-osi. Ostatak algoritma sprovodi niz transformacija nad slikom čiji je krajnji rezultat skup HOG vektora. Zatim je potrebno pronaći skup HOG vektora na dijelu originalne slike koji je sličan skupu HOG vektora šablona slike. Algoritam ekstrahuje HOG šablona slike kojoj se mijenjaju dimenzije počevši od minimalne dimenzije pa do trećine minimalne dimenzije originalne slike da bi se omogućilo prepoznavanje lica bez obzira na njegovu veličinu na originalnoj slici. Sa različitim veličinama HOG-a šablona slike algoritam prolazi kroz originalnu sliku i pamti one koje su iznad određenog praga (threshold).

*Slika 1.1 Originalna slika Slika 1.2 Šablon slika Slika 1.3 Izlazna slika*

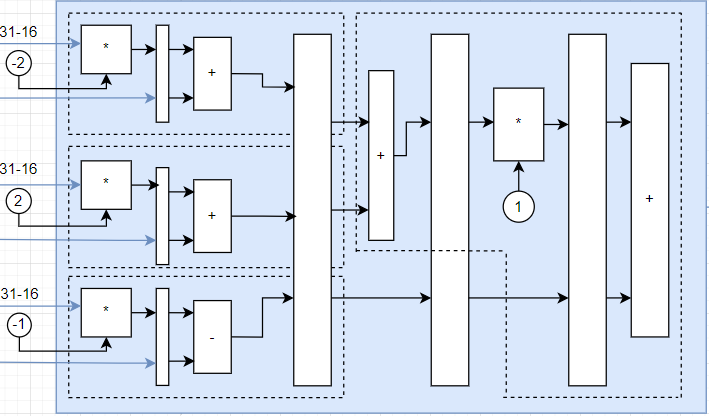
1. Implementacija algoritma u C jeziku

**

*Slika 2. Implementacija algoritma u C jeziku*

1. Optimizacija algoritma

Filter funkcija sadrži operacije množenja, sabiranja i oduzimanja, te je idealna za mapiranje na DSP module. Da bi se optimizovao algoritam, ideja je da se obrađuje 8 tačaka u paraleli što predstavlja unutrašnju k petlju. S obzirom da je kod oba filtera peti član niza nula, nema potrebe obrađivati devetu tačku jer neće uticati na rezultat. Takođe, za član filter niza koji je jednak nuli nije potrebno koristiti množač, već se piksel direktno vodi na sabirač. Da bi se izračunao jedan filter potrebno je 4 DSP bloka (*Slika 3.1*). A za dodatnu optimizaciju obrađivaćemo istovremeno 16 filtera (po 8 za oba filtera). Ovo ograničenje je posljedica nedostatka većeg broja DSP blokova na Zybo ploči.



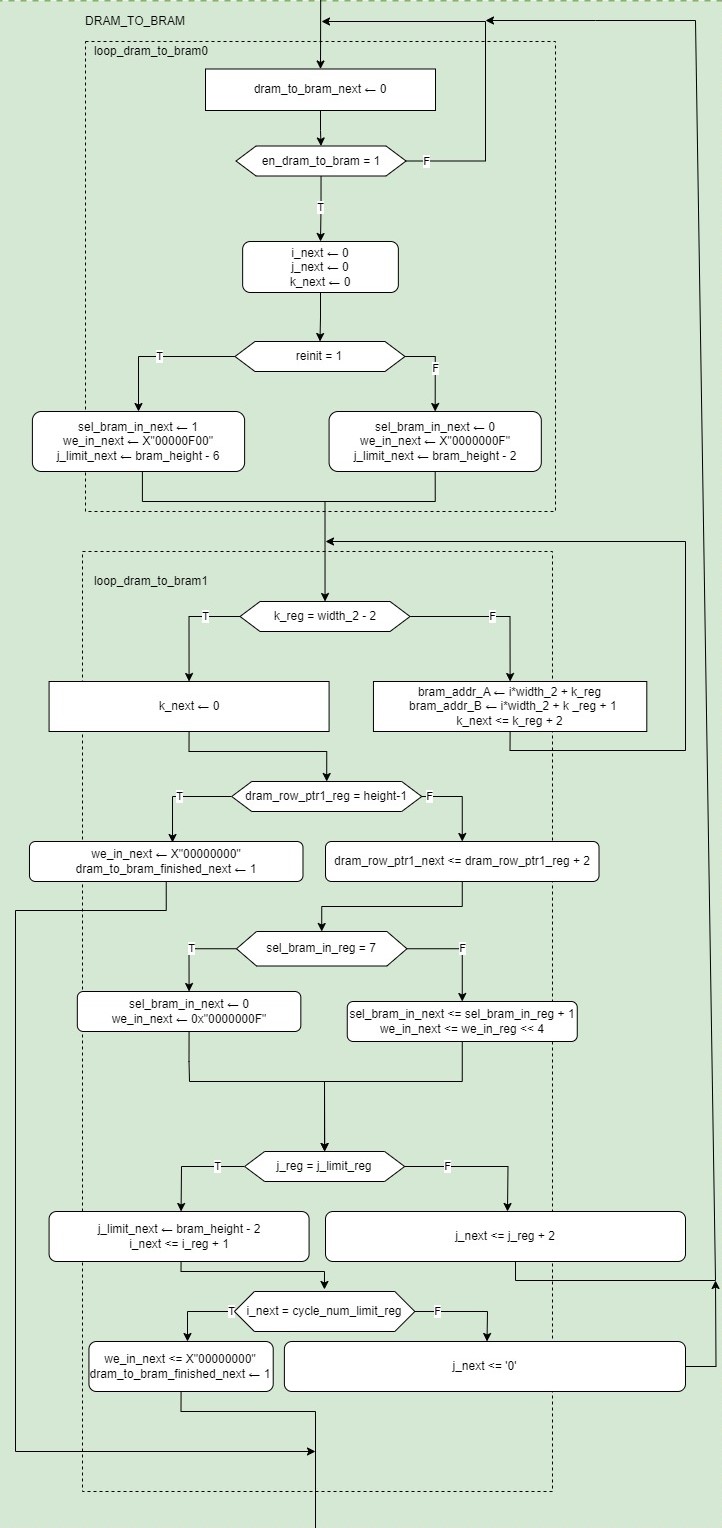
*Slika 3.1 Filter blok*

Dodatno, obrada podataka se vrši u 5 faza pipeline-a. U prvoj fazi se čitaju podaci iz BRAM blokova i uz pomoć rutirajuće mreže prosleđuju na odgovarajuće ulaze filter blokova. Zatim se filtriranje slike odvija u 3 faze, dok posljednja faza podrazumijeva upis u BRAM blokove, što je takođe kontrolisano rutirajućom mrežom.

1. Kontrola sistema

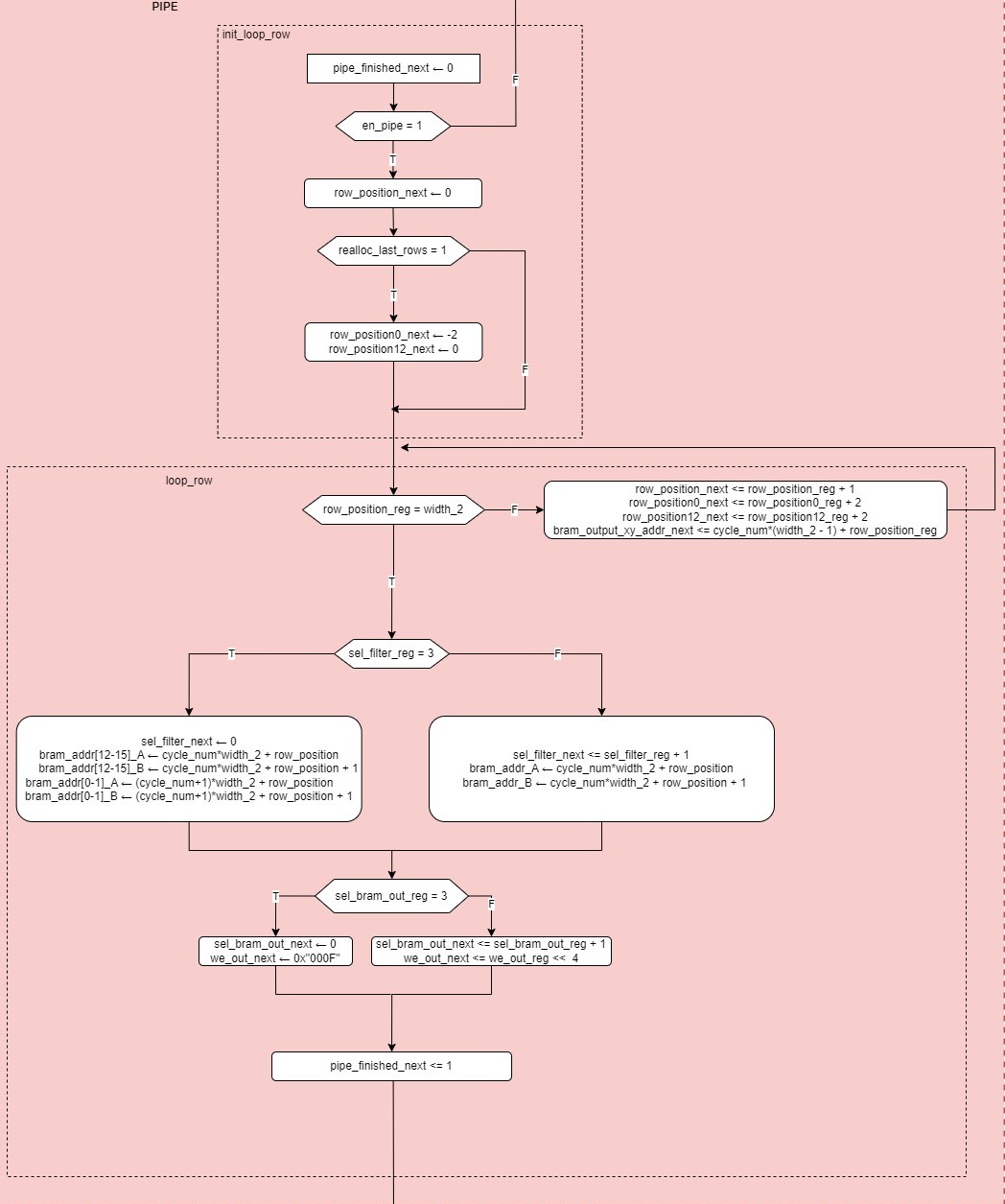
Unutar kontrole postoje 3 jedinice koje će biti usklađene od strane FSM-a.

Sa teorijske tačke gledišta DUT može da radi nad neograničenom slikom, preciznije, maksimalne dimenzije slike ograničene su veličinom DRAM-a. Ideja je da se prebacuju redom redovi slike u BRAM blokove (njih 16), sve dok se ne popuni njihov kapacitet. Istovremeno se vuče 8 podataka iz DRAM-a preko 2 linije od 64 bita. Zatim se oni vode na rutirajuću mrežu koja će odrediti BRAM blok u koji se upisuju odgovarajući pikseli. Ideja je da se nakon svakog upisanog reda prelazi na upis u naredni BRAM blok. Takođe, upis se vrši preko 2 porta BRAM bloka. Cjelokupnu kontrolu nad ovom fazon obrade ima modul DRAM\_TO\_BRAM. Kontroliše adrese sa kojih se čita iz DRAM-a, adrese na koje se upisuje u BRAM blokove kao i mrežu za rutiranje podataka. ASM dijagram ovog bloka prikazan je na *Slici 4.2*.



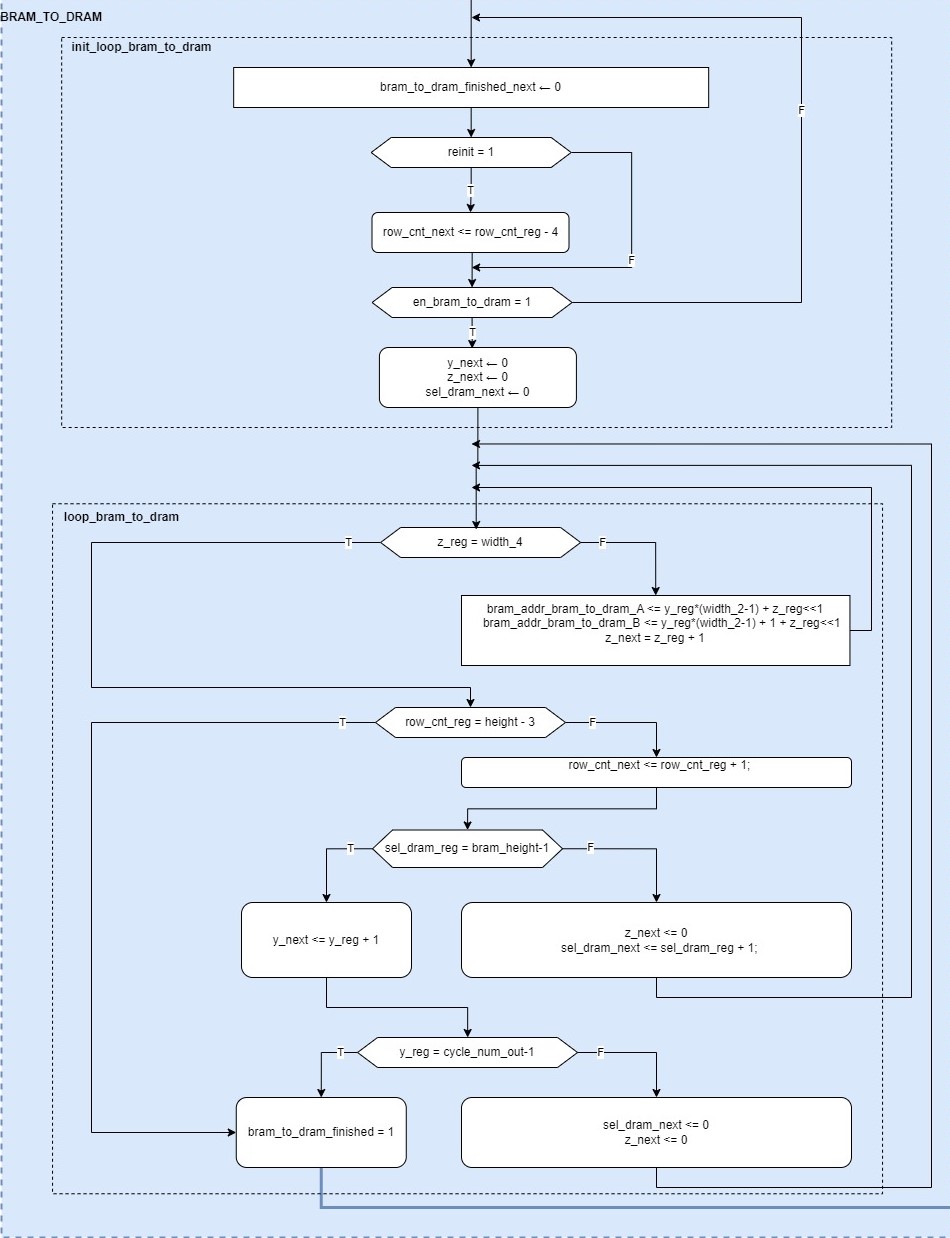
*Slika 4.2 ASM dijagram DRAM TO BRAM modula*

Modul koji kontroliše protočnu obradu je CONTROL\_LOGIC. Svrha ovog bloka je kontrola rutirajuće mreže ulaznih piksela, obrađenih piksela te formiranje adrese za upis filtriranih piksela u izlazne BRAM blokove. ASM dijagram CONTROL\_LOGIC modula prikazan je na *Slici 4.3*.



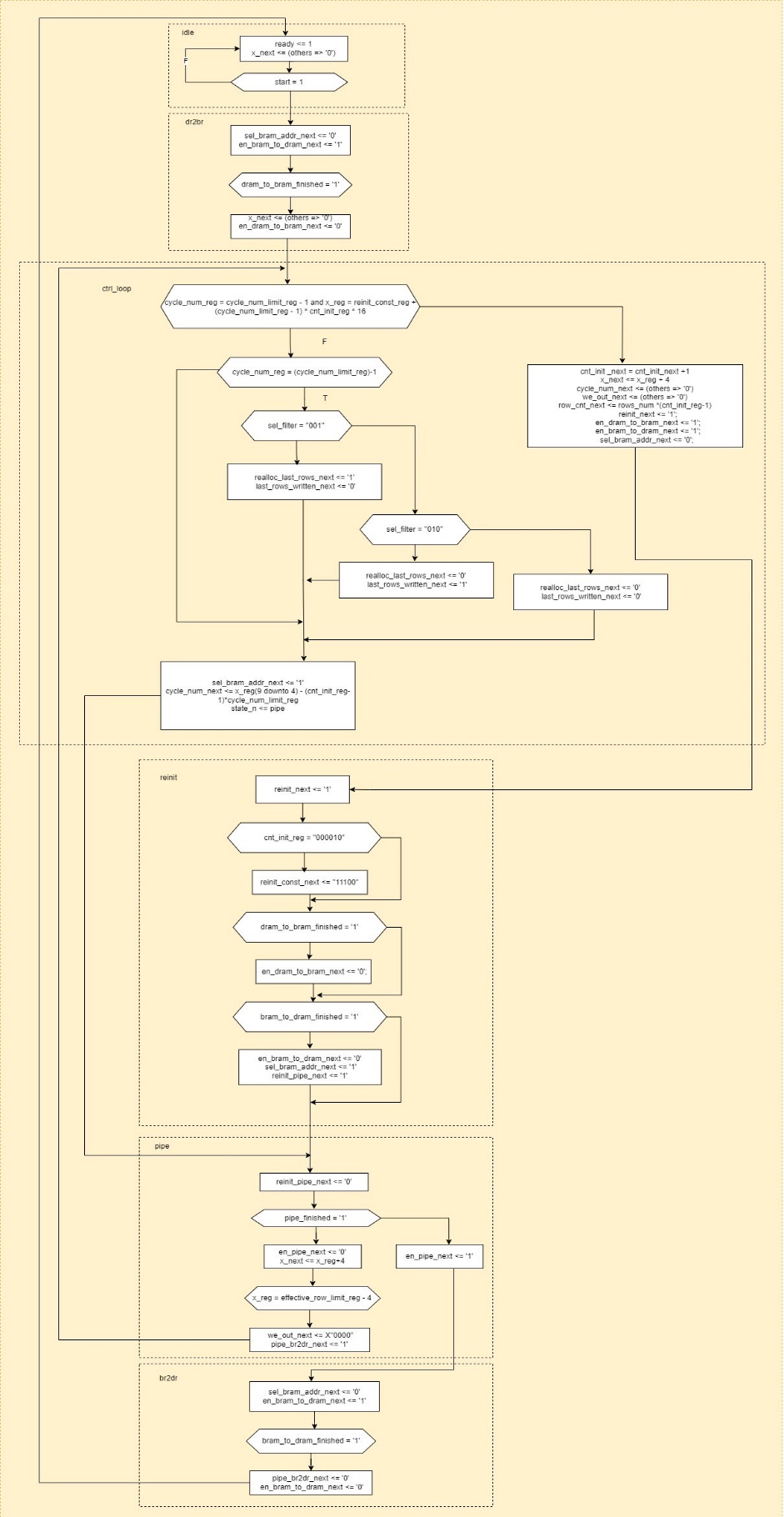
*Slici 4.3 bloka CONTROL LOGIC modula*

Modul koji kontroliše upis izlaznih piksela u dram je BRAM\_TO\_DRAM. Ovaj modul je odgovran za generisanje adrese sa koje će podaci biti pročitani iz BRAM blokova, adrese na koju će podaci biti upisani u DRAM, kao i za kontrolu rutirajuće mreže koja će odrediti iz kog BRAM bloka će podaci biti pročitani. ASM dijagram BRAM\_TO\_DRAM modula prikazan je na *Slici 4.4*.



*Slika 4.4 ASM dijagram BRAM TO DRAM modula*

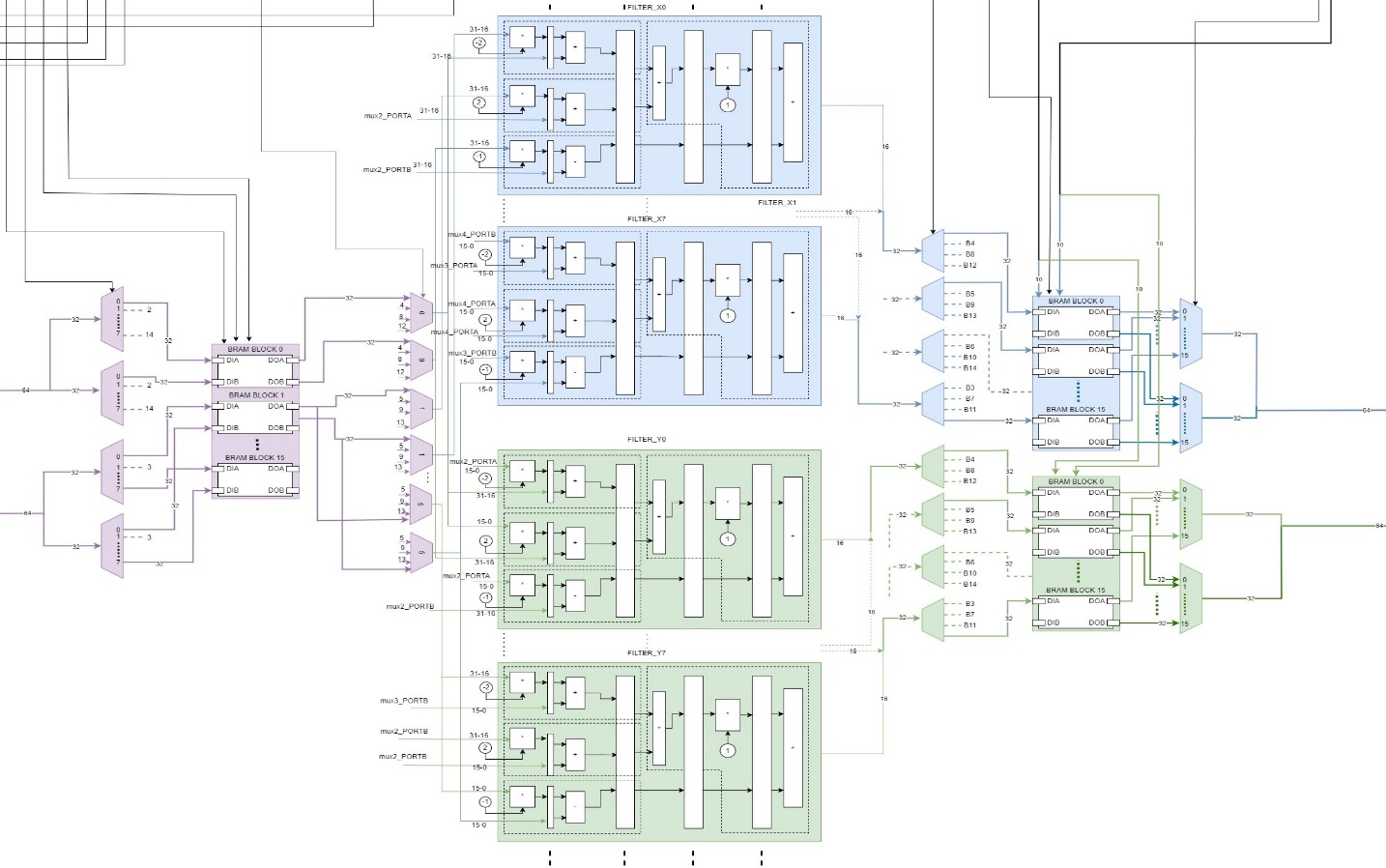
FSM je modul koji kontroliše redoslijed izvršavanja prethodnih modula. Ukoliko je slika većih dimenzija od kapaciteta BRAM-a potrebno je preostale neobrađene piksele ponovo smjestiti u BRAM i obraditi. Proces se ponavlja sve dok postoje neobrađeni pikseli u DRAM-u. Ova faza nosi naziv REINIT i podrazumijeva pokretanje DRAM\_TO\_BRAM-a da bi se smjestili novi podaci iz DRAM-a u BRAM i BRAM\_TO\_DRAM-a da bi se obrađeni podaci poslali iz BRAM-a u DRAM. Ukoliko nema potrebe za reinicijalizacijom faze se izvršavaju sekvencijalno. ASM dijagram FSM modula prikazan je na *Slici 4.5.*

**

*Slika 5.5 ASM dijagram FSM modula*

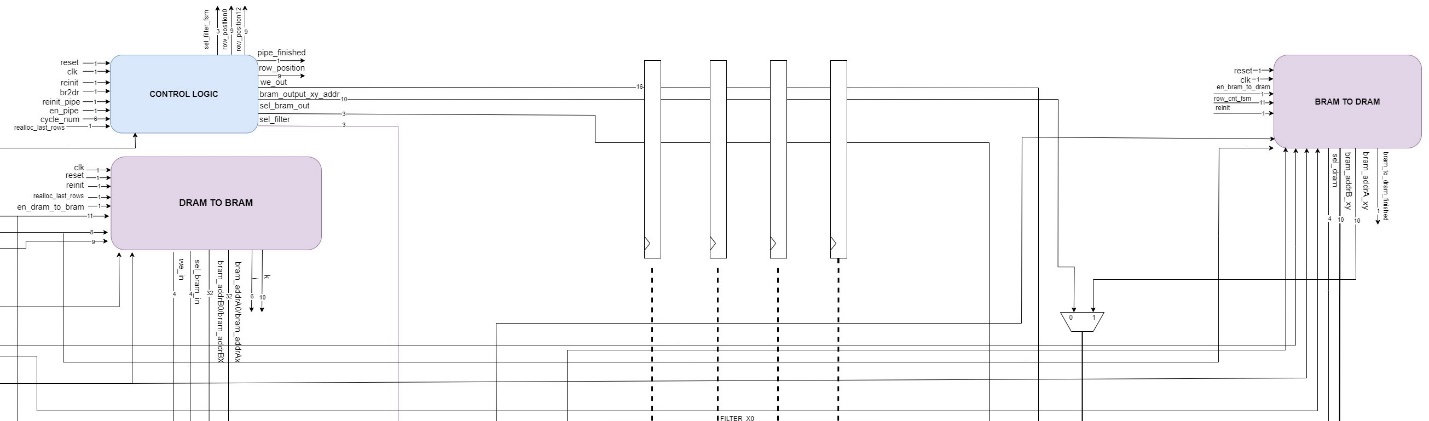
1. Izgled DUT-a

Izgled jedinice za obradu podataka prikazan je na *Slici 5.1*. Demultiplekseri na ulazu sistema služe za kontrolu izbora BRAM bloka u koji će podaci biti upisani. Nakon što su podaci upisani u BRAM blokove i pročitani iz istih, multiplekseri će iskontrolisati iz kog BRAM bloka će se podaci slati za obradu na filtere. Da bismo dobili 8 obrađenih tačaka u paraleli, potrebna su nam 24 ulazna piksela (na svaki filter blok se prosleđuju 3 piksela, ukupan broj filter blokova je 8). Analogno tome se obrađuje i drugi filter. Filter blokovi su sastavljeni od 4 DSP modula i izvršavaju neophodne operacije sabiranja i množenja. Nakon što su pikseli obrađeni multiplekseri će odrediti u koji bram blok se smještaju obrađeni podaci. Ideja je da se nakon smiještanja obrađenog reda slike prelazi u naredni BRAM blok. Nakon što se podaci pročitaju iz BRAM blokova, demultiplekser određuje koji obrađeni red se prosleđuje na izlaz *Data path*-a.



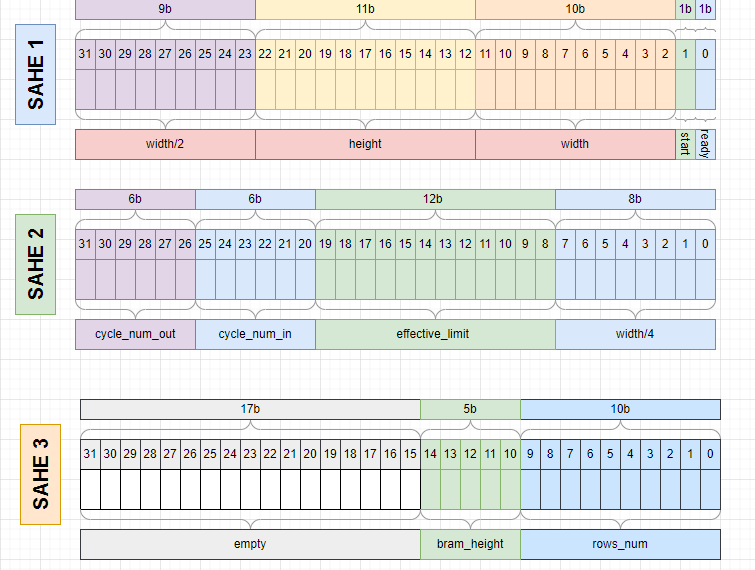
*Slika 5.1 Izgled jedinice za obradu podataka*

Kontrolna jedinica prikazana je na *Slici 5.2,* aopisana detaljno u sekciji 4.



*Slika 5.2 Izgled kontrolne jedinice*

Unutar sistema postoji registarska banka u kojoj su smještene konstante potrebne za izvršavanje algoritma. Ove konstante se šalju prema programabilnoj logici u vidu SAHE-a da bismo smanjili kašnjenje prilikom konfiguracije sistema. Izgled SAHE-a prikazan je na *Slici 5.2.*



*Slika 5.2 Izgled SAHE-a*

Na *Slici 5.1* uočavaju se uokvireni DSP blokovi unutar filter jedinica. Za jedan filter potrebna su 4 DSP bloka, a s obzirom da se u sistemu nalazi ukupno 16 filtera, za ovu fazu je potrebno 64 DSP bloka. Ukupan broj iskorištenih blokova je 76, preostalih 12 se koristi za izračunavanje granica i adresa u okviru *Control path* jedinice (*Tabela 5.1*).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Funkcionalna jedinica | Filter block | Datapath | Controlpath | Ukupno |
| DSP | 4 | 64 | 10 | 75 |

*Tabela 5.1. Pregled iskorišćenih DSP jedinica*

Memorija za fotografiju se mapira na BRAM blokove (veličine 4KB) . Ulazni pikseli se smiještaju u 16 BRAM blokova, dok se obrađeni pikseli po filteru x i y smiještaju takođe u po 16 BRAM blokova (*Tabela 5.2*).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Funkcionalna jedinica | Ulazni pikseli | Filter x | Filter y | Ukupno |
| BRAM blok | 16 | 16 | 16 | 48 |

*Tabela 5.2. Pregled iskorišćenih BRAM blokova*

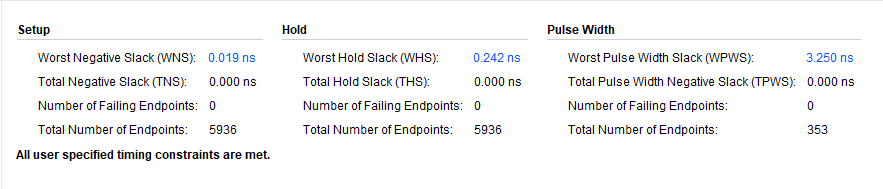
Teorijski gledano sistem može da radi nad neograničenom slikom, nakon što se obrade svi ulazni pikseli, vrši se dopremanje nove ture piksela iz DRAM-a.

Na slici su prikazani rezultati iskorišćenih resursa nakon implementacije

1. Frekvencija rada i kritična putanja filter\_core-a

Na ovom nivou dizajna sistem može da radi na frekvenciji od 133.333 MHz (*Slika 6.1*). Kada se u sistem dodaju druge jedinice, moguće je da će frekvencija pasti.





*Slika 6.1 Tajming analiza*

Sinteza je rađena u modu out\_of\_context, pa su podaci o kritičnoj putanji nepouzdani. Kritična putanja se nalazi na izlazu DSP bloka, preciznije prenošenje podatka na izlaz izlaznog registra.

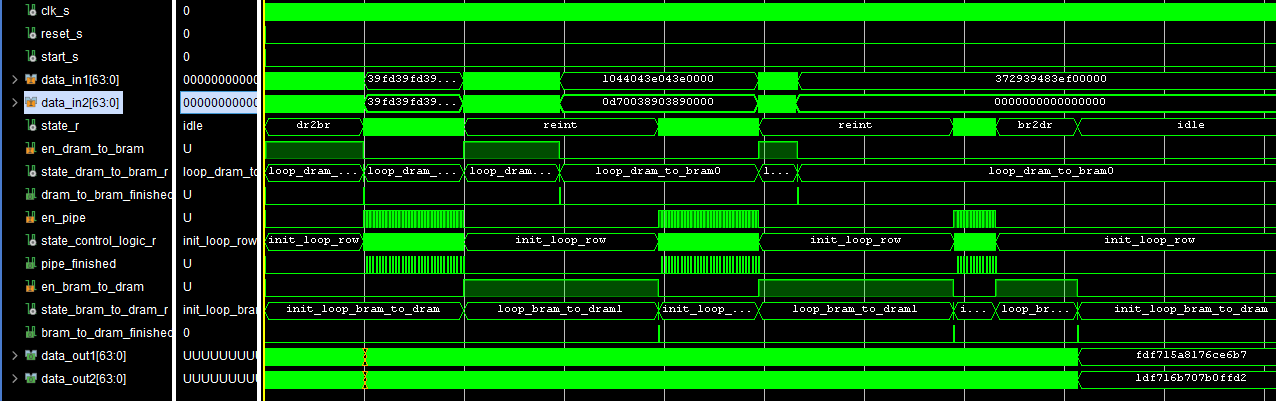


*Slika 6.2 Kritična putanja*

1. Testbenč

Da bi se provjerila funkcionalnog sistema napravljeno je jednostavno verifikaciono okruženje. Sistemu se prosljeđuju slike različitih dimenzija i uz odgovarajuće prilagođavanje parametara dobijaju rezultati filtriranih podataka.

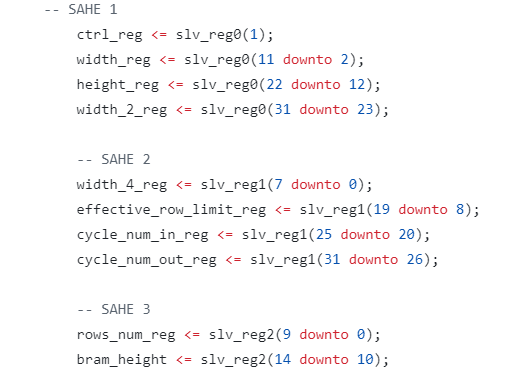
Očekivani rezultati iz virtuelne platforme se poklapaju sa dobijenim rezultatima testbenča. Na *Slici 7.1* može se uočiti da sistem prolazi kroz sva potrebna stanja prilikom obrađivanja slike za koju je potrebno da se reinicijalizacija izvrši dva puta.



*Slika 7.1 Testbenč slike 246x300*

1. Pakovanje IP jezgra

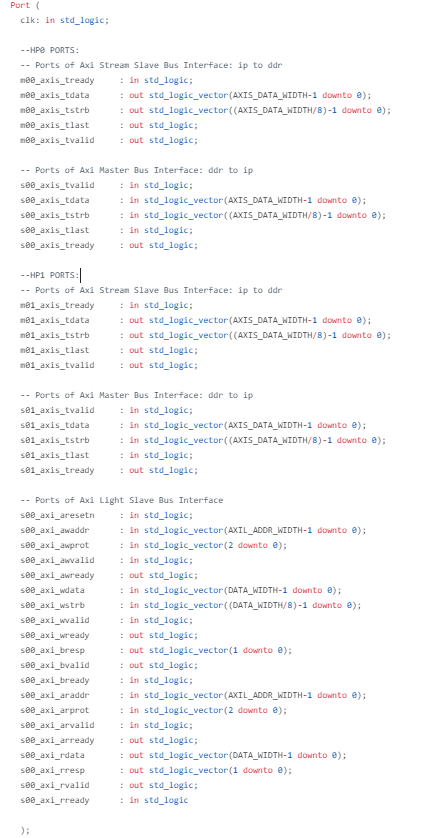
Da bi se omogućila komunikacija sa procesorom potrebno je koristiti odgovarajući protokol koji se nalazi na Zybo platformi. Za ovaj sistem iskorišćen je AXI Lite protkol koji služi za upis u registre i čitanje iz registara. Registri za čitanje i upis su: start, height, width, width\_2, width\_4, effective\_limit, bram\_height, rows\_num, cycle\_num\_in, cycle\_num\_out, dok se iz registra ready može samo čitati. Na *Slici 5.2* prikazan je izgled SAHE-a, dok se način na koji su ovi registri povezani za Filter\_core-om prikazan na *Slici 8.1*.



Slika 8.1 Povezivanje registara sa Filter\_core-om

Za prenos podataka koristi se AXI Stream protokol. Koriste se 2 HP porta na svakom po jedan AXI Stream, prenosi se istovremeno ulaznih 8 podataka dok se na izlazu prenosi po 4 podatka za svaki filter.

Na *Slici 8.2* prikazan je port na najvećem nivou hijerarhije. On se sastoji iz portova AXI Lite i AXI Stream protokola.

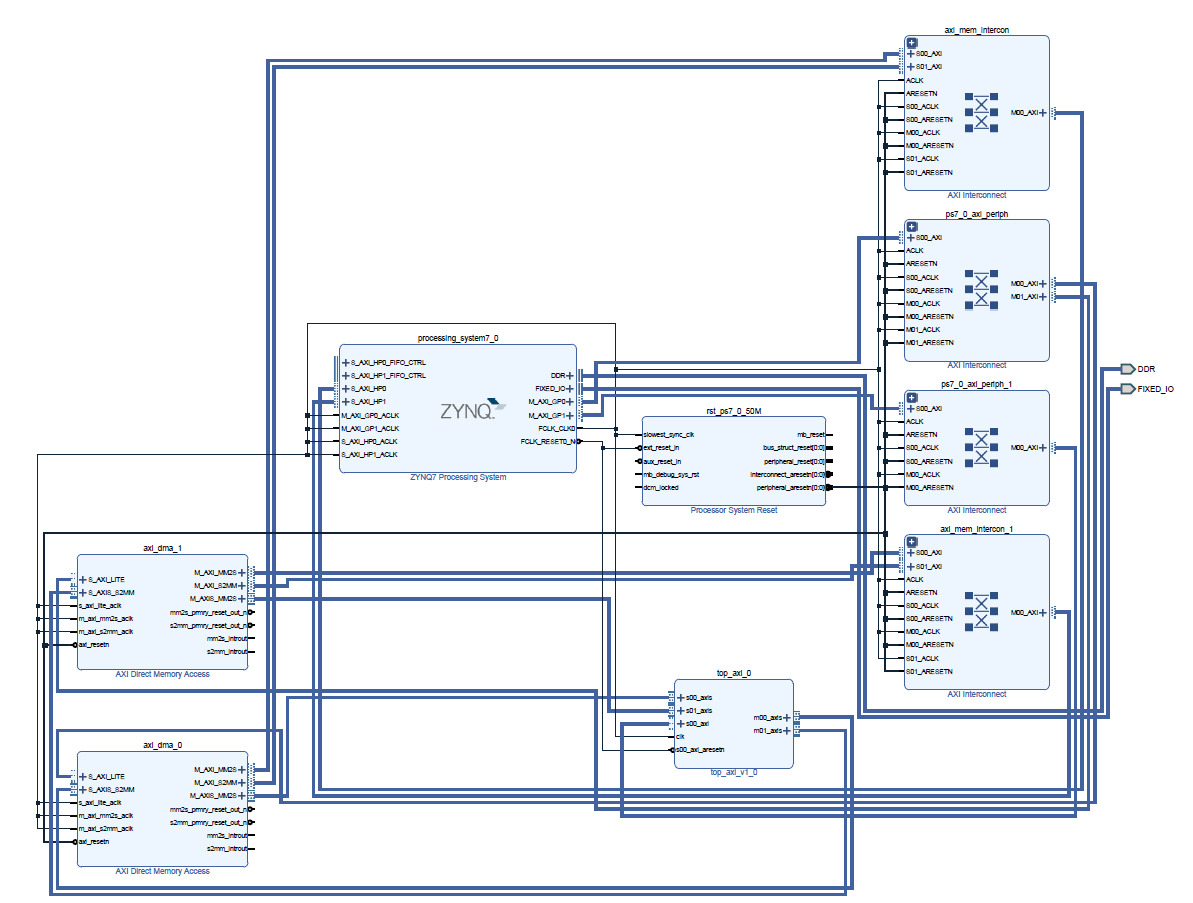


*Slika 8.2 Port na najvećem nivou hijerarhije*

1. Blok dizajn

Izgled blok dizajna prikazan je na *Slici 9.1.* U dizajn je ubačena kompoenenza Zynq processing\_system7\_0 koja predstavlja procesor. Zatim su izvršena odgovarajuća podešavanja frekvencije rada jezgra MHz, a nakon implementacije je utvrđeno da ona odgovara sistemu.

Pored njega dodata je jedinica filter\_core i sva AXI-DMA od kojih je svaki namijenjen jednom AXI Stream-u. Svaka DMA jedinica podešena je na isti način, izbačeni nepotrebni potrovi, proširena je linija za podatke na 64 bita. Nakon toga izvršeno je automatsko povezivanje jedinica unutar sistema. Dodatno je bilo potrebno povezati ip\_core sa DMA.



*Slika 9.1 Blok dizajn*

10.Frekvencija rada i i kritična putanja cijelog sistema

Nakon izvršene sinteze i implementacije pokazalo se da sistem može da radi na MHz. Kritična utanja se nalazi između na *Slici 10.*.

*Slika 10. Kritična putanja*

11. Testiranje rada u *Vitis*-u

Nakon sinteze i implementacije, generisan je Bitstreamo sistem je spušten na uređaj bez oprativnog sistema. Takođe, napisana je *bare-metal* aplikacija.

Upis u registre se izvršava se putem funkcije Xil\_Out32 u zavisnosti od dužine podataka koji se proleđuje, dok se iščitavanje vrši funkcijom Xil\_Out32. Na početku koda generiani su makroi za adrese registara. Na *Slici 11.1* prikazani su maroi za adrese registara.

*Slika 11.1 Registarski makroi*

Popunjavanje BRAM blokova i iščitavanja BRAM-ova se vrši pomoću funkcija DMA\_Simple\_Write() i DMA\_Simple\_Read().

12. Skripta

Da bi se omogućilo automatizovanje realizacije sistema, kreirana je TCL kripta. Skripta učitava odgovarajuće fajlove, pakuje jezgro, kreira blok dizajn, konfiguriše i povezuje odgovarajuće komponente. Takođe, vrši sintezu, implementaciju, generisanje Bitstream-a i njegovo export-ovanje.