

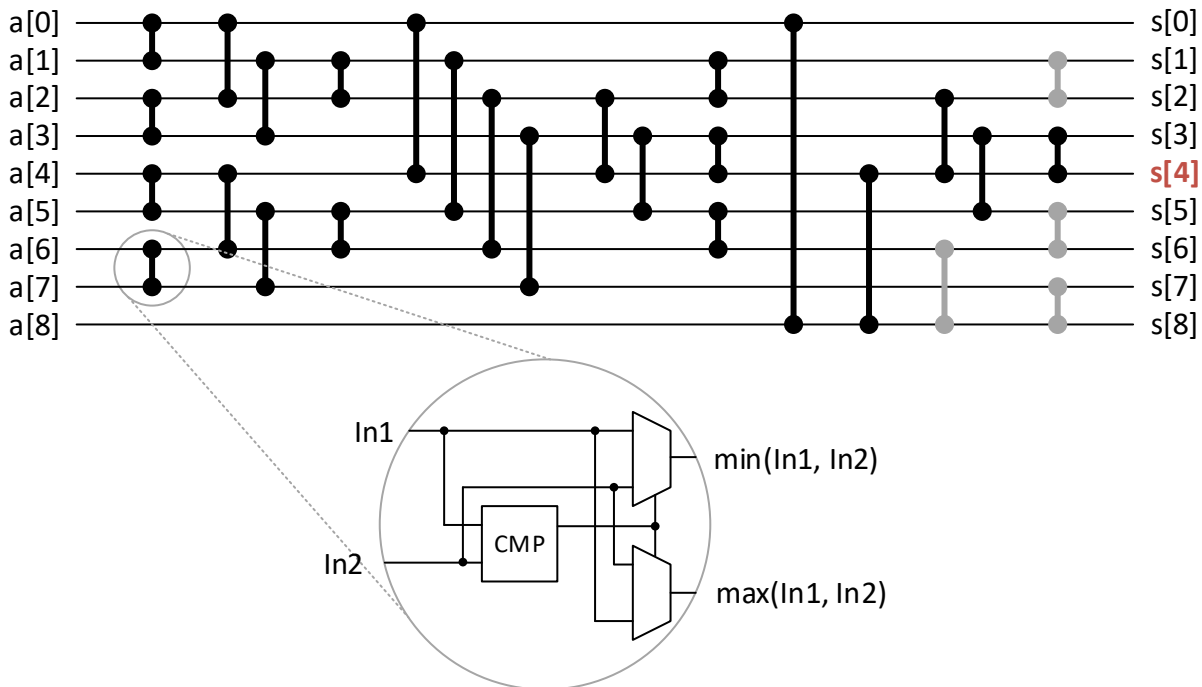
PRVA FAZA PROJEKTA

Medijan filtriranje slike

Potrebno je implementirati metodu za popravku kvaliteta slike na FPGA korišćenjem VHDL jezika i demonstrirati rad na FPGA razvojnom sistemu. U ovom dokumentu je opisana faza projekta koji se sastoji iz dve celine koje se nadovezuju jedna na drugu.

1. Mreža za izdvajanje medijane

Za potrebe filtriranja slike, potrebno je uraditi sortiranje devet susednih piksela i izdvojiti središnji element (medijanu). Mreža koja treba da vrši sortiranje se realizuje korišćenjem "Odd-Even Merge" arhitekture koja je prikazana na slici. Svaka vertikalna linija označava blok koji određuje manju i veću vrednost dva ulaza i prosleđuje ih na izlaz tako da manja vrednost bude bliža indeksu nula, a veća indeksu 8. Na izlazu cele šeme dobija se sortirani niz. Imajući u vidu da je za dalju obradu u ovom projektu potrebna samo medijana, tj. vrednost $s[4]$, blokove za poređenje označene sivom bojom nije neophodno realizovati. Čak i ako bi bili realizovani u VHDL kodu, nakon sinteze će automatski biti optimizovani, tj. eliminisani iz dizajna, jer nisu potrebni za generisanje izlaza.



- Implementirati mrežu za izdvajanje medijane iz niza od 9 osmobitnih elemenata.
- Postaviti registre na ulaz i izlaz i proceniti maksimalnu učestanost rada ove mreže, nakon out_of_context implementacije.
- Na odgovarajuća mesta postaviti pajplajn registre kako bi se postigla veća maksimalna učestanost rada. Ispitati maksimalnu učestanost rada pajplajnovane komponente.
- Simulirati sve komponente i priložiti *testbench* fajlove.

2. Medijan filter

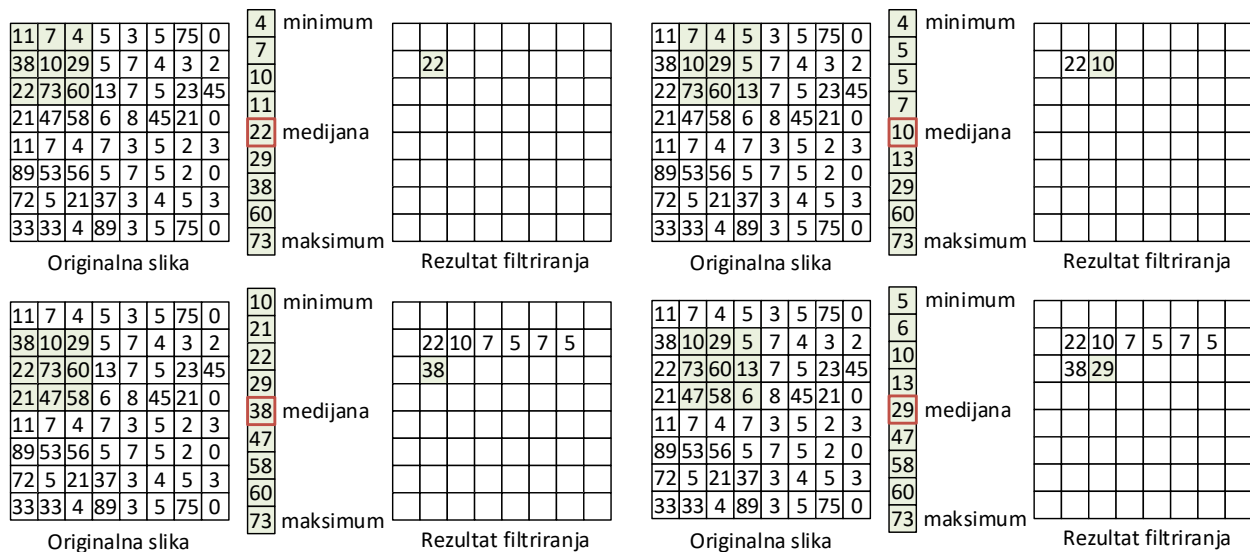
Memorija za čuvanje slike

Grayscale slika veličine 256×256 8-bitnih piksela je smeštena u inicijalizovanoj BRAM memoriji koja je data kao prateći fajl ovog dokumenta (*im_ram.vhd*). Za inicijalizaciju se koristi fajl *lenaCorrupted.dat* koji je potrebno smestiti na istu lokaciju kao i *im_ram.vhd*. Memorija je realizovana kao "Simple Dual Port" memorija, s obzirom na to da će se u kasnijim fazama zadatka koristiti jedan port za čitanje, a drugi za upis u memoriju. Dubina memorije je $256 \times 256 = 65536$, a na svakoj memorijskoj lokaciji se nalazi 8-bitna vrednost koja predstavlja intenzitet piksela. Pikseli su poređani po redovima, tj. u prvih 256 lokacija se nalazi 256 piksela prvog reda slike, u drugih 256 lokacija se nalaze pikseli drugog reda slike itd.

Sintetisati memoriju *im_ram* i uočiti da ona zauzima 16 blokova BRAM memorije, po dva za svaki bit. Dva bloka su kaskadirana na način opisan u odeljku "Cascadable Block RAM", na 23. strani dokumenta dostupnog na linku https://docs.xilinx.com/v/u/en-US/ug473_7Series_Memory_Resources.

Filtriranje

Slika u memoriji je pogođena impulsnim šumom koga je potrebno ukloniti medijan filtrom. Medijan filter je filter koji piksel na slici menja medijanom vrednosti svih piksela iz unapred definisanog susedstva, uključujući i vrednost piksela koja se menja. Preporučujemo za čitanje članak sa linka: https://en.wikipedia.org/wiki/Median_filter. Primer obrade nekoliko piksela prikazan je na slici ispod.



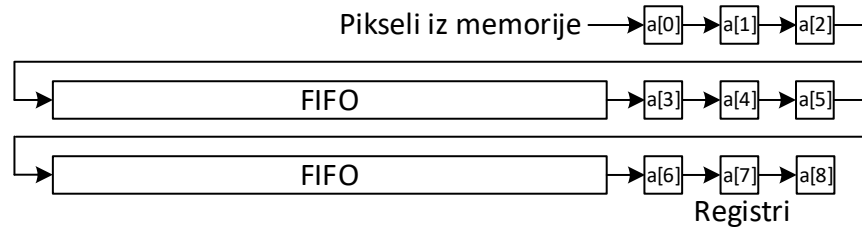
Treba obratiti pažnju da je uz ivice slike susedstvo piksela manje, tj. da neki pikseli nemaju sve susedne piksele sa svake strane. Zbog toga je slika u memoriji zapravo slika dimenzija 254×254 piksela, proširena nulama, što odgovara operaciji *medfilt2d* iz *Scipy* paketa. U Pajton skripti je dato učitavanje slike iz fajla, njeno filtriranje i prikaz rezultata.

Modul koji implementira medijan filtriranje treba da čita piksele iz RAM memorije. Rezultat filtriranja se mora upisati u istu memoriju, čime se originalni sadržaj memorije uništava. U drugoj fazi projekta, rezultat filtriranja će biti prikazan na računaru korišćenjem serijske komunikacije. Na UART se mogu slati svi pikseli, uključujući i proširene piksele, dakle svih 256×256 piksela. Nema potrebe raditi isecanje značajnih piksela slike u hardveru. Ipak, u ovoj, prvoj, fazi projekta, rezultat filtriranja, pored upisa u memoriju, treba sačuvati u fajl čiji sadržaj treba učitati u Pajtonu i uporediti sa softverski filtriranom slikom. Postoje dva načina: 1) upis u fajl direktno iz testbench-a u kom slučaju je neophodno otvoriti prilagoditi test upisu u tekstualni fajl (primer dat uz tekst

projekta u direktorijumu *hdl*; primer predstavlja testiranje FIR filtra čiji se ulazi čitaju iz fajla, a izlazi upisuju kada su podaci spremni) i 2) ispisivanjem odgovarajućih signala iz simulacije korišćenjem tcl komandi iz Vivado konzole (uputstvo je dato u direktorijumu *vivado_logger*). Pogledati oba načina i odabrati jedan.

Filtriranje se može realizovati na više načina, ali se preporučuje da se najpre dve linije slike baferuju u dva FIFO bafera i da se kreira struktura kao na slici. Pikseli iz memorije se čitaju na svaki takt po jedan. FIFO baferi su dubine $256 - 3 = 253$. Korišćenjem ovakve strukture postiže se da se lokalno susedstvo pomera na svaki takt i omogućava se obrada jednog piksela po taktu. Brzina obrade mora biti približna jednom pikselu po taktu. Preporučuje se studentima da najpre analiziraju tok podataka pre nastavka projekta.

Implementirati medijan filtriranje slike opisano u ovom odeljku, a rezultate uporediti sa filtriranjem iz Pajton skripte. Odrediti maksimalnu učestanost rada sistema. Priložiti sve simulacione fajlove.



(opciono) Fleksibilna veličina maske: Modul koji filtrira sliku može da bude generički u vreme sinteze. Modul treba da ima odgovarajući generik/parametar kojim se podešava veličina maske, i to pre sinteze. Ovo nije neophodno uraditi, ali grupe koje podrže ovu fleksibilnost mogu dobiti nagradne poene u slučaju da im je potrebno za veću ocenu. Ako se podržava fleksibilnost, treba podržati tri veličine maske: 3x3, 5x5 i 7x7. **Napomena:** Imati u vidu da je fleksibilnost dodatna komplikacija projekta, pa preporučujemo da se najpre realizuje filter za veličinu maske 3x3, a tek onda uradi nadgradnja. Za realizaciju generičke mreže za sortiranje preporučujemo sistoličku arhitekturu koja nije optimalna sa stanovišta hardverskih resursa, ali je jednostavna za realizaciju (primer u poglavlju 4.1.2 master rada sa [linka](#)).