Univerzitet u Beogradu – Elektrotehnički fakultet
Katedra za elektroniku
Odsek za elektroniku i digitalne sisteme

Projektovanje VLSI sistema (19E043VLSI)

PROJEKAT

Popravka kontrasta slike ekvalizacijom histograma

Potrebno je implementirati metodu za popravku kvaliteta slike na FPGA korišćenjem VHDL jezika i demonstrirati rad na FPGA razvojnom sistemu. U ovom dokumentu je opisan zadatak koji se sastoji iz nekoliko celina koje se nadovezuju jedna na drugu.

1. deo – Ekvalizacija histograma (30 poena)

Memorija za čuvanje slike

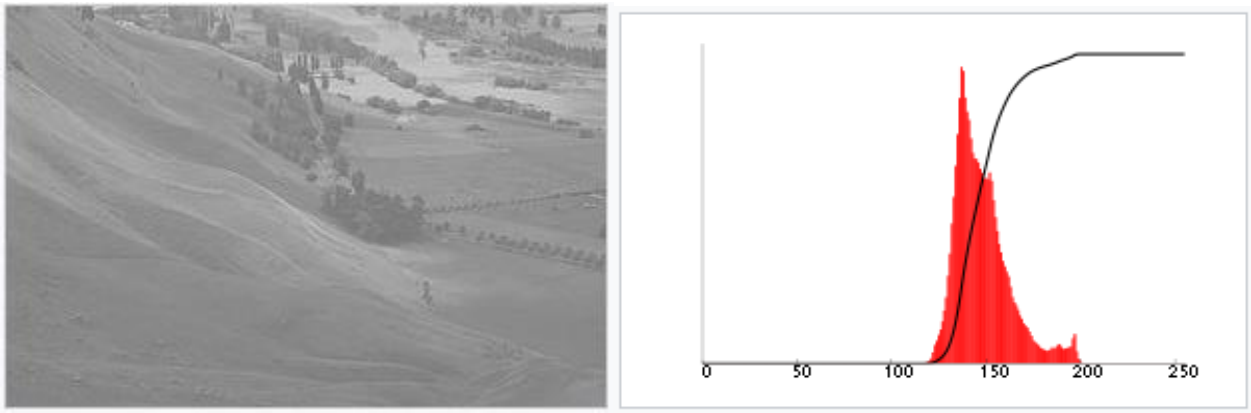
Grayscale slika veličine 256 × 256 8-bitnih piksela je smeštena u osam paralelno postavljenih inicijalizovanih BRAM memorija čija implementacije je data kao prateći fajl ovog dokumenta (*im_ram.vhd*). Za inicijalizaciju se koriste fajlovi *lenaCorruptedX.dat* (*X* = 0, 1, 2, …, 7) koje je potrebno smestiti na istu lokaciju kao i *im_ram.vhd*. Memorija je ralizovana kao "*Simple Dual Port*" memorija, s obzirom na to da će se u zadatku koristiti jedan port za čitanje, a drugi za upis u memoriju. Dubina jedne memorije je 256 × 256 / 8 = 8192, 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.

Uz ovaj fajl dat je i primer instanciranja svih 8 blokova memorije, pri čemu su ulazi i izlazi memorija mapirani na niz od 8 podataka. Za potrebe ovog projekta biće potrebno koristiti te podatke paralelno (svih 8 odjedanput) ili serijski (jedan po jedan).

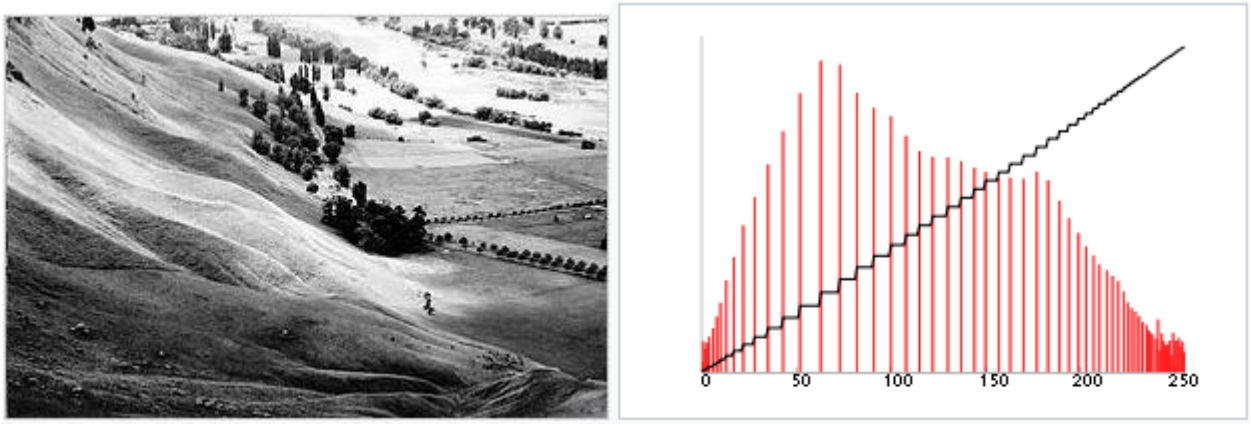
Ekvalizacija histograma

Sliku koja se nalazi u memorijama za čuvanje slike potrebno je obraditi i rezultat obrade upisati na iste memorijske lokacije na kojima se nalazila prethodna slika. Uz ovaj fajl, dostupna je i softverska realizacija ekvalizacije histograma koju potrebno pretočiti u hardver. Ona je data uz deo koda koji se odnosi na čitanje slike sa FPGA ploče. Ovde će biti dat opis tog koda, bez egzatnog matematičkog aparata. O ekvalizaciji histograma se može više čuti na predmetu Digitalna obrada slike. Za potrebe ovog projekta, ekvalizacija histograma je samo primenjena na sivu sliku (*grayscale*).

Ekvalizacija histograma predstavlja metod popravke kontrasta na slikama kojim se histogram slike menja tako da se na obrađenoj slici sve vrednosti piksela što uniformnije pojavljuju. Slika u ovom zadatku predstavlja matricu 8-bitnih vrednosti inteziteta svetla svakog piksela. Histogram slike predstavlja niz verovatnoća pojave svake od 256 vrednosti piksela na slici i može se dobiti prostim brojanjem koliko piksela ima svaku od vrednosti i deljenjem dobijenih brojeva sa ukupnim brojem piksela. Primer slike (levo) i histograma (desno) dat je na slici 1. Pored histograma, definiše se kumulativni histogram (prikazan crnom linijom na slici 1 - desno) koji predstavlja niz verovatnoća da je piksel na slici manji ili jednak odgovarajućoj vrednosti piksela (0 do 255). Kod slika koje imaju dobar kontrast, kumulativni histogram najčešće ima linearan rast sa vrednošću piksela, kao što je to prikazano na slici 2. Stoga je cilj ekvalizacije histograma da obezbedi izmenu piksela na slici tako da kumulativni histogram bude približno linearna funkcija.



Slika 1. Originalna slika i njen histogram i kumulativni histogram pre ekvalizacije



Slika 2. Obrađena slika i njen histogram i kumulativni histogram posle ekvalizacije

U ovom zadatku, vrednost histograma za određenu vrednost piksela treba da predstavlja samo ukupan broj piksela koji imaju tu vrednost. Nije potrebno računati verovatnoće. Histogram se dakle može predstaviti nizom brojeva gde se na i-toj poziciji nalazi sledeći broj:

$$h[i] = \sum_{r=0}^{255} \sum_{c=0}^{255} (im[r][c] == i)$$

gde su sa r i c označeni red i kolona piksela. Ako bi se histogram izrazio preko verovatnoća, tada bi bilo potrebno podeliti svaku vrednost ukupnim brojem piskela:

$$p[i] = h[i]/65536$$

Nakon izračunatog histograma, kumulativni histogram (kumulativna verovatnoća) se može izračunati sa:

$$cp[i] = \sum_{j=0}^{i} p[j].$$

Ekvalizacija histograma predstavlja transformaciju piksela ulazne slike $y = T(x)$ tako da se ova funkcija cp linearizuje. Konkretno, transformacija je:

$$T[i] = 255cp[i] = \sum_{j=0}^{i} p[j] = 255 \sum_{j=0}^{i} \frac{h[j]}{65536} \approx 256 \sum_{j=0}^{i} \frac{h[j]}{65536} = \frac{1}{256} \sum_{j=0}^{i} h[j].$$

Zaokruživanje normalizacione konstante na 256 ima mali uticaj na rezultat u konkretnom primeru, ali značajno utiče na kompleksnost hardvera, tj. njime se izbegava množenje.

Dakle, za potrebe ovog zadatka, treba implementirati izračunavanje histograma, izračunavanje skaliranog kumulativnog histograma i samu ekvalizaciju.

Hardverska realizacija ekvalizacije histograma

Imajući u vidu da se slika nalazi podeljena u 8 nezavisnih memorija, izračunavanje histograma se može paralelizovati i to tako što se instancira 8 modula koji izračunavaju histograme za svaki od delova slike. Histograme treba čuvati u *HIGH_PERFORMANCE* memorijama. Dakle, modul za računanje histograma treba da čita jedan po jedan piksel i da uvećava podatke u memoriji za histogram. Memorija za histogram je dubine 256 lokacija – po jedna za svaku vrednost histograma. Nakon što svaki od modula završi izračunavanje histograma svog dela slike potrebno je da signalizira glavnoj kontroli koja će započeti čitanje svih 8 histograma, sabiranje pojedinačnih vrednosti stablom sabirača, a zatim i izračunavanje kumulativnog histograma. Memorija za kumulativni histogram, radi daljeg izračunavanja, ne treba da bude odvojena, već je potrebno skalirani kumulativni histogram slike upisati nazad u svaku od 8 memorija u kojima su ranije računati pojedinačni histogrami. Ovim se omogućava da se ekvalizacija histograma radi ponovo sa paralelizmom 8. Nakon što je završen upis kumulativnog histograma, započinje proces ekvalizacije koja predstavlja prosto čitanje memorije u kojoj je sada skalirani kumulativni histogram pri čemu je adresa stara vrednost piksela, a podatak koji se dobija nova vrednost piksela. Rezultat ekvalizacije se upisuje nazad u memorije za čuvanje slike. Dizajn hardvera treba da bude takav da učestanost rada sistema bude što je moguće veća i da je za računanje potrebno što je moguće manje taktova.

Sve memorije treba da budu *HIGH_PERFORMANCE.* Uvođenjem ovog ograničenja, učestanost signala takta će biti veća. Ipak, mora se voditi računa o hazardima podataka koji mogu da nastanu ako je potrebno pročitati podatak koji još uvek nije upisan u memoriju. Ovo se može desiti pri izračunavanja histograma kada su vrednosti uzastopnih susednih piksela iste ili kada su iste vrednosti razmaknute samo za jedan takt. U ovim situacijama je neophodno uvesti cikluse pauze ili omogućiti baferisanje međurezultata kako se ne bi izgubili podaci. Uvođenje ciklusa pauze, iako najlakše rešenje, dovodi do toga da će svaki od osam modula za računanje histograma delova slike završiti računanje u različitom trenutku, pa se u proces sabiranja finalnog histograma ne sme ići dok svaki od modula ne završi računanje svog histograma. Dakle, vreme izvršavanja zavisi od podataka.

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

2. deo – Slanje slike na PC računar (20 poena)

UART predajnik

Na raspolaganju je i fajl *uart_tx.vhd*, u okviru koga je implemenitan UART predajnik. Ova komponenta prima 8-bitni podatak sa porta *tx_data*, kada je signal *tx_dvalid* na “1”, koji označava da je podatak validan. Ako slanje prethodnog podatka nije u toku, započinje se slanje podatka *tx_data* i prelazi u zauzeto stanje, tokom koga je signal *tx_busy* na “1”. Tokom zauzetog stanja, na izlaznom pinu *tx* se jedan po jedan šalju biti brzinom koja je podešena generikom *SER_FREQ*. Prijemnik će za potrebe ovog projekta biti USB-UART konvertor koji treba povezati sa FPGA pločom. Preporučuje se brzina od 115 200 bps. Kada signal *tx_busy* padne na “0”, korsnik

može postaviti novi podatak i *tx_dvalid* na “1”, čime se započinje novo slanje. Da bi se ostvarila brzina definisana generikom *SER_FREQ*, neophodno je podesiti generik *CLK_FREQ* na vrednost koja odgovara učestanosti signala takta koji se koristi u sistemu. Signal *par_en* se može postaviti na “0”.

Zadatak

Korišćenjem navedenih komponenti potrebno je implementirati čitanje slike iz memorije i slanje svih piskela slike na PC računar. TX port UART-a je mapiran na GPIO port A0 na FPGA ploči. Ovo je podešeno u pratećoj .xdc skripti koju treba po potrebi izmeniti. Na raspolaganju je Pajton skripta koja prima podatke na strani računara i iscrtava sliku. Pajton skripta se može pokrenuti iz PyCharm-a ili bilo kog drugog okruženja, samo je neophodno instalirati paket *pyserial*.

Slanje se startuje pritiskom na taster i traje sve dok se ne pošalju svi pikseli slike. Svaki put kada se pritisne taster započinje se novo slanje slike, a ako se taster pritisne u toku slanja, pritisak se ignoriše.

- Implementirati komponente potrebne za detekciju pritiska tastera.
- Implementirati komponentu za slanje podataka na PC računar.
- Simulirati sve komponente i priložiti *testbench* fajlove.
- Testirati slanje slike na računar na FPGA platformi. Potrebno je najpre pokrenuti Pajton skriptu, a zatim startovati transfer. Imajući u vidu da se šalje 65536 piskela, slanje bi trebalo da traje nešto duže od pet sekundi.
- Uporediti rezultate ekvalizacije histograma u hardveru sa rezultatima ekvalizacije u Pajtonu. Razlike ne bi trebalo da postoje.

Povezivanje USB-UART modula sa FPGA pločom i računarom, kao i instrukcije za instalaciju drajvera, prikazani su u nastavku.

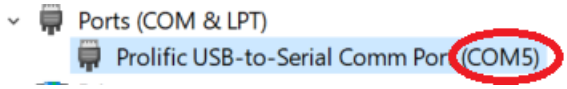
Povezivanje i instalacija

USB – UART drajver

Konvertor USB na UART koji ste dobili u svojim paketićima služi za komunikaciju PC računara i FPGA ploče.

Ukoliko koristite Windows operativni sistem, za njega je potrebno instalirati drajver i to u sledećim koracima:

1. Raspakovati arhivu *Prolific PL2303 driver v3.3.2.102 (2008-24-09) Win8_x64_x86.7z.*
2. Windows + X i odabrati Device Manager;
3. U okviru *Ports (COM & LPT)* pronaći *pl2303hxa* uređaj i kliknuti desni klik na njega i odabrati *Properties*;
4. Odabrati *Driver* tab pa *Update Driver*, pa *Browse My Computer for Drivers*, pa *Allow me to choose from a list of available drivers on my computer*.
5. Kliknuti na "*Have disk ...*" dugme.
6. Pronaći folder u kome se nalaze fajlovi koji su raspakovani u tački 1 i odabrati fajl "*ser2pl*".
7. Kliknuti *OK* i nastaviti sa instalacijom.
8. Nakon instalacije u *Device Manager*-u bi trebalo da se pojavi uspešno povezan uređaj, kao na slici ispod. Zabeležiti njegov COM port, u primeru sa slike, to je COM5.



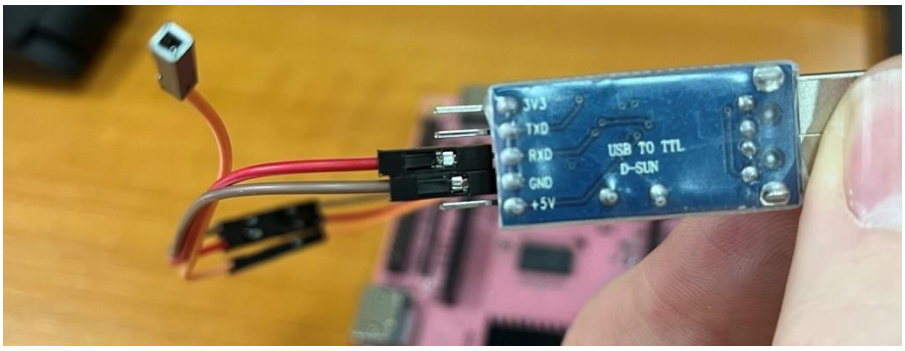
Ukoliko koristite Linux operativni sistem, instalacija drajvera nije potrebna, ali ime porta nije COM5, već je potrebno pronaći ga. Najlakše je to uraditi nakon što priključite USB-UART konvertor u računar pogledom u direktorijum */dev/serial/by-path*. Najverovatnije će se videti dva porta, ako je i ploča priključena. Uočiti oznaku ttyUSB0, ttyUSB1 i slično. Oznaka koja NE stoji uz Xilinx, a uz koji najverovatnije stoji pl2303, oznaka je USB-UART konvertora, zabeležiti je. Potrebno je dozvoliti i upis i čitanje u /dev/ttyUSBx fajl, i to sa:

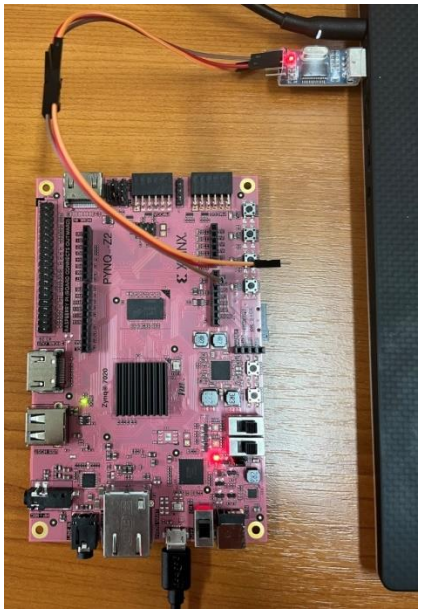
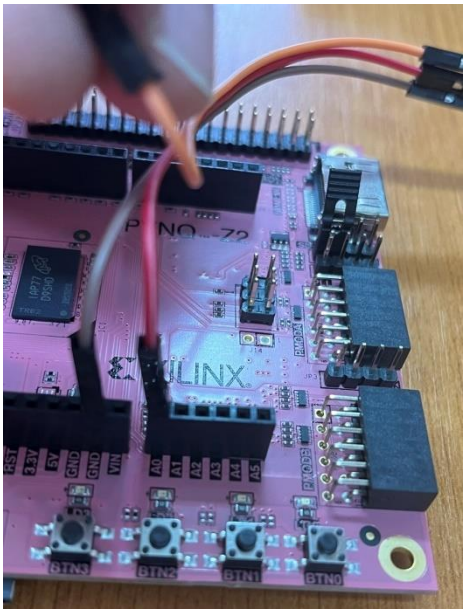
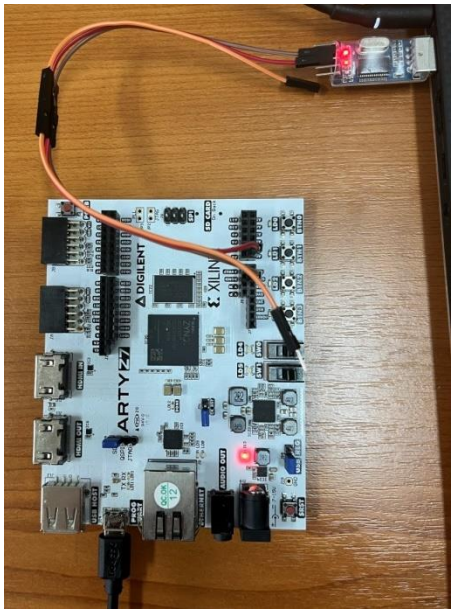
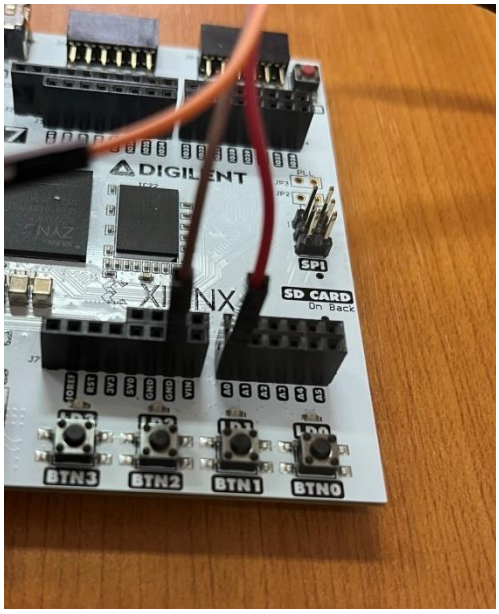
sudo chmod 666 /dev/ttyUSBx

gde je x odgovarajući broj. Ovo je potrebno ponoviti svaki put kada se USB ponovo priključi u računar.

USB – UART Povezivanje sa FPGA pločom

FPGA ploča šalje podatke, pa se na USB – UART pločici koristi pin za prijem (RX). Takođe, neophodno je povezati i masu jedne i druge ploče kako bi referentni potencijal bio isti. Primeri .xdc skripti za obe ploče su dati kao prateći fajlovi ovog dokumenta, a fotografije u nastavku prikazuju povezivanje. Čip na obe ploče je isti, i to xc7z020clg400-1.



Dodatne specifičnosti za korisnike Linux OS

Pre korišćenja Vivado alata na linux operativnim sistemima, potrebno je ručno instalirati drajvere za konekciju sa pločom. Uputstvo je dostupno na linku: https://docs.xilinx.com/r/2021.2-English/ug973-vivado-release-notes-install-license/Install-Cable-Drivers. Putanja instalacije je najverovatnije */tools/Xilinx/Vivado/2021.1/*. Stoga je potrebno skriptu pokrenuti sa *sudo* permisijama. Programiranje ploče je isto kao i u laboratoriji.

Python skriptu za učitavanje slike možete pokrenuti iz bilo kog okruženja. Ipak, najbolje je napraviti poseban venv. Nekoliko komandi kojima se instaliraju potrebni paketi sledi:

```
mkdir imeDirektorijuma
cd imeDirektorijuma
python -m venv venv
source venv/bin/activate
```

```
pip install pyserial
pip install numpy
pip install scipy
pip install matplotlib
pip install pyqt5
```

Ako koristite Visual Studio Code, dovoljno je da pokrenete `code .` i da u `imeDirektorijuma` iskopirate *main.py* i *lenaCorrupted.bmp*.