DIGITALNI VLSI SISTEMI 2022/2023 – PROJEKTNI ZADATAK

UBRZANJE ALGORITMA ZA ODREĐIVANJE MAGNITUDE GRADIJENTA NA SLIKAMA

1. Kreiranje osnovnog sistema

Potrebno je realizovati sistem za određivanje magnitude gradijenta na slikama korišćenjem prostorne maske dimenzija 3x3. Ulazna slika je u binarnom formatu pri čemu prva 4 bajta predstavljaju širinu slike, druga 4 bajta visinu slike dok nakon toga slede pikseli slike u **uint8** formatu. Slika u binarnom formatu se nalazi na ***host*** računaru i učitava se u SDRAM memoriju korišćenjem ***host file system*** mehanizma. Po uključivanju ***hostfs*** opcije u okviru BSP, moguće je koristiti standardne C funkcije **fread** i **fwrite** za pristup binarnim fajlovima koji se nalaze na host računaru. ***Voditi računa da ova opcija radi samo u Debug modu.*** Smatrati da su ulazni podaci 8-bitni pikseli čija vrednost se nalazi u opsegu 0-255.

Kako bi se detektovale ivice potrebne su dve maske **coeffs_H** i **coeffs_V** koje odgovaraju horizontalnim i vertikalnim gradijentima. Moguće je koristiti 2 varijante računanja gradijenta: Standardni gradijent (indeks 0) i Sobel operator (indeks 1). Indeks trenutno korišćenog operatora se nalazi u **gradinet_operator** koji se može promeniti i sa host procesora. Koeficijenti filtra su definisani na sledeći način:

1) Standardni gradijent: $coeffs\_H = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ i $coeffs\_V = \frac{1}{2}\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
2) Sobel operator: $coeffs\_H = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ i $coeffs\_V = \frac{1}{8}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

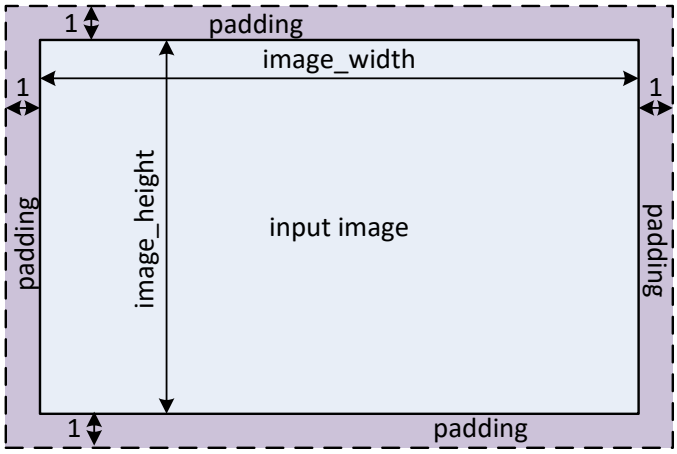
Rezultati fitriranja prostornim maskama **coeffs_H** i **coeffs_V** redom **gradient_H** i **gradient_V**

Magnituda gradijenta se određuje aproksimativno kao:
$\boxed{gradient_{magnitude} = abs(gradient_H) + abs(gradient_V)}$.

Izlazne vrednosti predstavljaju magnitude gradijenta pri čemu se ona predstavlja sa 16 bita, kao neoznačeni broj, od čega se 9 bita koristi za ceo deo i 7 bita za razlomljeni deo

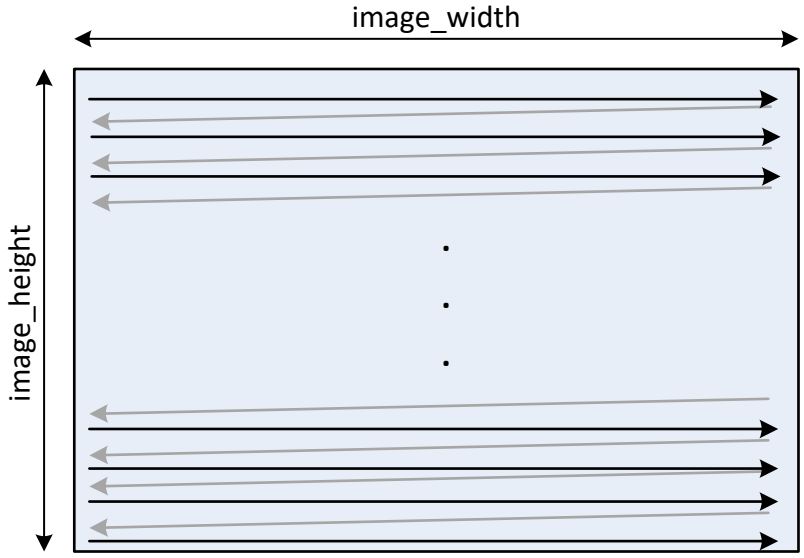
Procesiranje organizovati u okviru beskonačne ***while*** petlje gde se na početku procesiranja zahteva unos tipa gradijentnog operatora i imena fajla u kom se nalazi ulazna slika.

Nakon toga je potrebno alocirati prostor za proširenu ulaznu i izlaznu sliku (magnituda). Izlazna slika predstavlja matricu u kojoj će biti smeštene vrednosti magnitude gradijenta. Vodite računa o veličini pojedinačnih podataka prilikom alokacije memorije. Proširiti ulaznu sliku ponavljanjem linija sa svake strane kao što je prikazano na Slici 1 kako bi se obezbedilo potpuno lokalno susedstvo za svaki piksel.



Slika 1. Proširenje ulazne slike za filtriranje prostornom maskom 3x3

Nakon toga softverski realizovati određivanje magnitude gradijenta korišćenjem zadatih prostornih maski. Procesiranje se obavlja tako što se za svaki piksel ulazne slike izdvoji region 3x3 njegovih okolnih piksela. Oni se pomnože koeficijentima prostorne maske i saberu, čime se formira vrednost hoirzontalnog i vertikalnog gradijenta na toj poziciji. Nakon toga je potrebno odrediti magnitudu gradijenta korišćenjem navedene aproksimacije. Redosled procesiranja je prikazan na Slici 2.



Slika 2. Redolsed procesiranja piksela ulazne slike

Po završetku procesiranja, izlazne matrice magnitude i ugla gradijenta se šalju nazad na host računar i upisuju u izlazni binarni fajl. Prva 4 bajta izlaznih fajlova sadrže širinu isprocesirane slike, druga 4 bajta visinu isprocesirane slike, nakon toga slede podaci odgovarajuće širine. Kreiranje binarnih fajlova ulazne slike i učitavanje binarnih fajlova izlazne slike realizovati u Python-u.

Demonstrirati funkcionalnost sistema testiranjem za sve dostupne modove gradijenta. Pri evaluaciji rezultata izvršiti poređenje sa procesiranjem u Python-u.

Korišćenjem komponenti ***performance counter***-a izmeriti brzinu rada (procesiranja slike) ove realizacije na slici ***lena.bin***. Prilikom testiranja brzine koristiti Sobelov filtar. Odrediti brzinu procesiranja pojedinačnih delova algoritma i detektovati uska grla sistema.

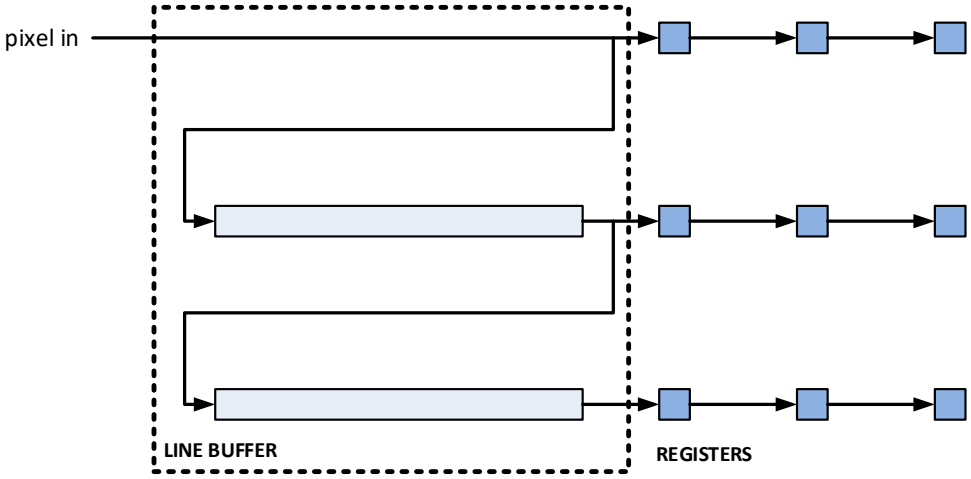
2. Ubrzanje korišćenjem hardverskih blokova

Kako bi se ubrzalo procesiranje potrebno je dodati sledeće hardverske blokove kojim se ubrzava procesiranje:

1) **acc_gradient_vectors** – prihvata piksele ulazne slike i određuje vrednosti horizontalnog i vertikalnog gradijenta. Potrebno je da ovaj blok ima jedan 8-bitni registar u kojem se čuva indeks gradijentnog operatora koji se trenutno koristi. Ovom registru se može pristupiti sa procesora korišćenjem Avalon-MM Slave interfejsa. Ulazni interfejs za podatke je Avalon-ST i povezuje se na ulazni DMA koji čita sliku iz memorije i šalje je u akcelerator. Izlazni interfejs predstavlja Avalon-ST interfejs širine 32 bita u kojem su zapakovane vrednosti horizontalnog i vertikalnog gradijenta.

2) **acc_gradient_magnitude** – prihvata vrednosti horizontalnog i vertikalnog gradijenta preko ulaznog Avalon-ST interfejsa, određuje vrednost magnitude i šalje je preko izlaznog Avalon-ST interfejsa na DMA koji prosleđuje podatke u SDRAM u prostor alociran za magnitudu gradijenta.

Kako bi se postiglo ubrzanje sistema potrebno je dodati dva navedena akceleratora kao i dva DMA kontrolera kojim se obezbeđuje prenos podataka između memorije i akceleratora. Akceleratore konfigurisati tako da procesiraju proširena ulazna slika ***lena.bin***.



Slika 3. Struktura za izdvajanje lokalnog susedstva dimenzija 3x3

Komponente akceleratora je potrebno projektovati što fleksibilnije, odnosno potrebno je obezbediti parametere za širinu i visinu slike. Ovi parametri mogu da se menjaju samo prilikom instanciranja komponente dok su u toku rada nepromenljivi. **Rešenje koje omogućava promenu veličine slike koje se procesira u *run-time*-u će doneti do 10% bonus poena.**

Na Slici 3. je prikazano izdvajanje lokalnog susedstva dimenzija 3x3 kojim se obavlja računanje vrednosti gradijenata. Kako su za izračunavanje jednog izlaznog piksela potrebni podaci iz 3 susedna reda na ulazu u filtar se nalazi linijski bafer kojim se baferišu dve prethodne linije ulazne slike. Linijski bafer ima 1 ulaz i 3 izlaza koji predstavljaju piksele iz 3 susedna reda slike. Na izlazu iz linijskog bafera nalazi se banka od 9 registara u kojima se nalaze pikseli koji pripadaju nekom lokalnom susedstvu veličine 3x3 i za koje je potrebno odrediti vrednost odgovarajućeg gradijenta. Pikseli iz lokalnog susedstva se množe sa 3x3 koeficijenatima filtra i zbir ovih proizvoda predstavlja vrednost odgovarajućeg gradijenta u toj tački. Kako bi se dobila vrednost drugog gradijenta potrebno je isti set od 9 piksela pomnožiti sa drugim setom koeficijenata. Na ulaz filtra se šalju pikseli iz proširene slike koja je prikazana na Slici 1, dok se sa izlaza

prosleđuju samo validni rezultati. Validni rezultati predstavljaju rezultate filtriranja za piksele koji imaju potpuno definisano lokalno susedstvo (označeno plavom bojom na Slici 1.).

Preporuka je da se za linijski bafer koristi komponenta **altshift_taps** koja predstavlja pomerački registar sa višestrukim izlazima. Za sabiranje se može koristi operator + u VHDL kodu samo voditi računa da podaci budu odgovarajućeg tipa signed ili unsigned i odgovarajuće širine. Množenje i deljenje stepenom broja 2 se može postići aritmetičkim pomeranjem. **Voditi računa o poziciji decimalne tačke nakon operacije množenja!**

Demonstrirati funkcionalnost sistema primenom svih zadatih prostornih maski na zadatoj slici. Uporediti rezultate sa softverskom realizacijom.

Korišćenjem komponenti ***performance counter***-a izmeriti brzinu rada (procesiranja slike) ove realizacije na slici lena.bin. Prilikom testiranja brzine koristiti Sobelov filtar.

Uporediti brzinu obrade u odnosu na softversku realizaciju.

NAPOMENA: Voditi računa o alociranju i dealociranju memorije. Sve bafere koji više neće biti potrebni u sistemu je potrebno dealocirati, čim prestane potreba za njihovim korišćenjem. **Vodite računa da ne dealocirate memoriju koju koristite kasnije!!!** Na primer nakon proširenja slike potrebno je dealocirati bafer u kom se čuvala originalna slika kao i memoriju koja je alocirana za lanac deskriptora.

NAPOMENA: Potrebno je da svaka grupa kreira blog na koji će postavljati vesti o svom trenutnom progresu. **Uniformisati ime bloga: DVS22_GX.** Portali na kojima se može besplatno pokrenuti blog su https://www.blogger.com/ i https://wordpress.org/ . Po otvaranju bloga potrebno je poslati link predmetnim asistentima kako bi mogli da prate progres projekta. **Važno je da redovno osvežavate vaš blog!**

Na kraju projekta je potrebno da kreirati izveštaj koji predstavlja dokumentaciju projekta i opisuje arhitekturu sistema, realizaciju hardvera i softvera za svaki od delova projekta. Izveštaj predstavlja dokumentaciju projektovanog sistema i potrebno je da bude dovoljno detaljan kako bi neko uz izveštaj i kod mogao potpuno da reprodukuje rezultate i iskoristi delove projekta u većem sistemu.

Za pun broj poena potrebno je da pored potpune funkcionalnosti sistem bude smisleno particionisan i kod uredno napisan sa dovoljnim brojem komentara.