Mihailo Grbić i Filip Parag

Protočna FPGA arhitektura za filtriranje slike linearnim i adaptivnim medijanskim filtrom

U ovom radu je opisana digitalna protočna arhitektura na FPGA čipu za filtriranje slike, koristeći linearno i adaptivno medijansko filtriranje. Linearno filtriranje je metod obrade slike koji se najčešće koristi za izoštravanje slike, izdvajanje ivica i otklanjanje Gausovog šuma, dok se medijansko i adaptivno medijansko filtriranje uglavnom koriste radi otklanjanja impulsnog šuma i očuvanja kontura na slici. Opisana protočna arhitektura ima značajnu primenu u vremenski kritičnim, integrisanim sistemima, pošto otklanja potrebu za višestrukim pristupima istim podacima u memoriji i minimizuje vreme izvršavanja, po ceni većeg zauzeća logičkih i memorijskih jedinica. U slučaju adaptivnog medijanskog filtriranja, radi pronalaženja medijane skupa vrednosti piksela, implementirane su i upoređene dve arhitekture: bitoničko sortiranje i kumulativni histogram. Sintezom je potvrđeno da arhitektura za bitoničko sortiranje zauzima duplo manje registara i logičkih jedinica i da postiže 20% višu maksimalnu dozvoljenu učestanost signala takta.

Uvod

Sa rastućom popularnošću pametnih uređaja i Internet of Things tehnologija postoji sve veća težnja da se digitalnim uređajima pruži mogućnost da „vide" svoju okolinu, kako bi bolje interagovali sa njom. Kamere danas možemo naći u automobilima, televizorima, kompjuterima, pametnim asistentima, sigurnosnim sistemima i u mnogim drugim uređajima. Masovno korisćenje kamera podrazumeva i veliku potrebu za digitalnom obradom slike u vidu otklanjanja šuma prilikom fotografisanja i naknadno izvlačenje bitnih odlika sa fotografije.

Digitalnu obradu slike moguće je izvršiti na procesorima opšte namene koji su već prisutni u većini uređaja. Međutim, značajan broj algoritama za digitalnu obradu slike podrazumeva određeni broj koraka tokom kojih se više puta pristupa istim podacima, tako da implementacija ovih algoritama na procesorima zahteva višestruko čitanje istih podataka iz memorije, što je vremenski skupo. Samim tim digitalna obrada slike na procesoru može da bude spora, da zauzima puno računarskih resursa i usporava izvršavanje drugih procesa. Ovi efekti su najizraženiji kod uređaja koji obrađuju video zapise u realnom vremenu, gde je potrebno korišćenje brze i skupe memorije ili smanjenje rezolucije snimka kako bi se ispunili vremenski zahtevi.

Kako bi se izbegao ovaj kompromis, većina pametnih uređaja sa kamerama koriste specijalizovane module koji brzo i efikasno izvršavaju potrebne algoritme obrade slike. U ovom radu je opisan jedan takav modul koji obrađuje sliku linearnim i adaptivnim medijanskim filtrom, sa minimalnim opterećenjem sistemske memorije.

Filtriranje slike u prostornom domenu

Filtriranje slike u prostornom domenu je grupa algoritama za digitalnu obradu slike koji menjaju vrednosti jednog po jednog piksela na slici na osnovu njegovih susednih piksela. Ovi algoritmi se najčešće koriste odmah nakon akvizicije slike sa senzora kamere, radi otkla-

Mihailo Grbić (1999), Beograd, učenik 4. razreda Matematičke gimnazije u Beogradu

Filip Parag (1999), Subotica, učenik 4. razreda Gimnazije „Jovan Jovanović Zmaj" u Novom Sadu

Mentori:
Stefan Krsmanović, Draexlmaier, Beograd

Đorđe Marjanović, student Elektrotehničkog fakulteta Univerziteta u Beogradu

njanja šuma. Pored toga, upotrebu imaju u izoštravanju i zamućivanju slike kao i izdvajanju ivica sa slike.

Linearno filtriranje

Linearno filtriranje je algoritam filtriranja u prostornom domenu koji vrednost rezultujućeg piksela računa kao linearnu kombinaciju vrednosti piksela u njegovoj okolini. U slučaju linearnog filtriranja, okolina jednog piksela predstavlja kvadrat neparne dimenzije d oko tog piksela, tako da svi pikseli koji su po x i y osi udaljeni za manje od $(d-1)/2$ polja od piksela koji se obrađuje, pripadaju njegovoj okolini. Dimenzija d mora biti neparan broj, kako bi piksel koji se obrađuje bio u centru kvadrata okoline.

Nakon što je određena okolina obrađivanog piksela, njegova nova vrednost se računa kao težinska suma svih piksela u njegovoj okolini po sledećoj formuli:

$$R(x,y)=\sum_{i=-a}^{a}\sum_{j=-a}^{a}K(i,j)\cdot O(x+i,y+j)$$

gde je $a=(d-1)/2$, R matrica rezultujuće slike, O matrica originalne slike, a K takozvana maska ili kernel, kvadratna matrica težina.

Linearno filtriranje se takođe može predstaviti kao prevlačenje kernela preko originalne slike, množenje vrednosti prekrivenih piksela sa njima odgovarajućim vrednostima unutar kernela i sumiranje tih proizvoda (slika 1). Iz ovog razloga se algoritmi filtriranja slike u prostornom domenu često nazivaju i funkcijama klizajućeg prozora (eng. sliding window).

Primećuje se da je rezultujuća slika manja od originalne slike, zato što za ivične piksele sa originalne slike ne postoje odgovarajući susedni pikseli. Da bi se izbeglo smanjivanje slike prilikom primene linearnog filtriranja, originalna slika se često proširi sa crnim ili belim granicama, ili sa pikselima koji nose istu vrednost kao ivični pikseli originalne slike (padding).

Linearno filtriranje je često korišćen algoritam zbog svoje jednostavnosti i velikog broja efekata koji se mogu postići primenom različitih kernela. Neki od najčešće korišćenih kernela za linearno filtriranje su Gausov kernel, kernel za izoštravanje i Sobelov kernel.

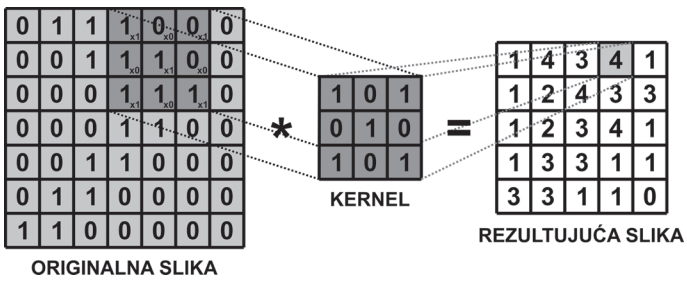
Gausov Kernel. Primenom Gausovog kernela (jednačina 1) čije vrednosti predstavljaju dvodimenzionu Gausovu raspodelu, vrednost piksela se približava vrednosti njegovih suseda i postiže se efekat blagog zamućenja slike. Pošto filtriranje Gausovim kernelom efektivno predstavlja niskopropusno filtriranje, ovaj filter se veoma često koristi za uklanjanje Gausovog šuma koji nastaje usled lošeg osvetljenja scene, visoke temperature senzora kamere, ili šuma pri prenosu podataka (Farooque i Rohankar 2013).

$$K_G=\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} \quad (1)$$

Kernel za izoštravanje. Kernel za izoštravanje (jednačina 2) povećava vrednost piksela ukoliko je veća od njegovih suseda, a u suprotnom je smanjuje. Ovime se postiže efekat visokopropusnog filtriranja slike koji pojačava i naglašava ivice.

$$K_I=\begin{bmatrix}0&-1&0\\-1&5&-1\\0&-1&0\end{bmatrix} \quad (2)$$

Sobelov kernel. Sobelov kernel za detekciju ivica (poznat kao i Sobelov operator) računa aproksimaciju gradijenta vrednosti slike (Sobel i Feldman 1968). Konkretno, primenom vertikalnog Sobelovog kernela (jednačina 3) vrednost rezultujućeg piksela na poziciji (u,v) predsta-



Slika 1. Grafički prikaz linearnog filtriranja slike

Figure 1. Visual representation of linear filtering

vljaće aproksimaciju vertikalnog gradijenta originalne slike na toj poziciji. Na ovaj način se postiže efekat izdvajanja ivica sa originalne slike.

$$K_S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (3)$$

Medijansko filtriranje

Slično linearnom filtriranju, medijansko filtriranje podrazumeva posmatranje kvadratne okoline obrađivanog piksela, ali umesto linearne kombinacije susednih piksela, medijansko filtriranje rezultujućem pikselu dodeljuje vrednost medijane svih piksela u okolini.

Glavna primena medijanskog filtriranja je otklanjanje takozvanog impulsnog (eng. salt and pepper) šuma, koji nastaje usled „mrtvih" piksela na senzoru kamere, ili usled promene vrednosti bita pri čuvanju ili prenosu podataka (Farooque i Rohankar 2013; Vasicek i Sekanina 2008). Ovaj šum se ispoljava u vidu retkih, nereprezentativnih piksela na slici, koji su jako crni ili jako beli. Impulsni šum je nepodoban za uklanjanje pomoću Gausovog filtera koji nereprezentativne piksele ne uklanja, već zamućuje i stvara efekat „pega" na slici. S druge strane, medijansko filtriranje u većini slučajeva uspešno uklanja impulsni šum i održava originalni izgled slike (Hwang i Haddad 1995).

Adaptivno medijansko filtriranje

Najmanja moguća i ujedno najčešće korišćena dimenzija okoline za medijansko filtriranje je $d = 3$. Ispostavlja se da je u većini realnih situacija ova dimenzija sasvim dovoljna za uspešno uklanjanje impulsnog šuma (Vasicek i Sekanina 2008).

Međutim, glavna mana klasičnog medijanskog filtra je osnovna pretpostavka da su nereprezentativni pikseli retki, tj. da ni u jednoj kvadratnoj okolini neće biti brojniji nego reprezentativni pikseli. Kod slika sa visokom koncentracijom impulsnog šuma (preko 10%) (slika 2a) ova pretpostavka neće uvek biti tačna (Hwang i Haddad 1995; Vasicek i Sekanina 2008), što podrazumeva da će u nekim okolinama vrednost medijane biti jednaka vrednosti impulsnog šuma i da klasičan medijanski filtar neće uspešno ukloniti taj šum (slika 2b).

Kako bi se ovakve situacije izbegle, moguće je koristiti medijansko filtriranje sa većom dimenzijom okoline *d*: od 5 pa čak do 15 piksela. Međutim, povećavanje okoline podrazumeva duže vreme izvršavanja algoritma, zato što se velika okolina *d* posmatra za sve piksele slike (a ne samo za one kojima je to potrebno), kao i lošiji kvalitet i znatno zamućenje rezultujuće slike (Vasicek i Sekanina 2008) (slika 2c).

Usled gore navedenih nedostatka klasičnog medijanskog filtriranja javio se metod adaptivnog medijanskog filtriranja (Hwang i Haddad 1995) koji donosi 2 bitna unapređenja. Najpre, adaptivno medijansko filtriranje ne menja vrednost piksela ukoliko taj piksel nije zašumljen, čime osigurava minimalan gubitak informacija i visok kvalitet rezultujuće slike. Pored toga adaptivno medijansko filtriranje najpre obrađuje prozor malih dimenzija ($d = 3$) i proširuje ga samo ukoliko je to potrebno, smanjujući tako vreme izvršavanja algoritma.

U nastavku je izložen ovaj algoritam uz kratko objašnjenje:

```
A:
Okolina = nadji_okolinu(slika, poz, dim)
min, med, max =
      izracunaj_minmedmax(Okolina)

if min < med < max
     goto B
else
     dim += 1
if dim <= dim_max
     goto A
else
     return slika[poz]
B:
if min < slika[poz] < max
     return slika[poz]
else
     return med
```

Dati pseudokod predstavlja funkciju koja kao ulaz prima sliku i poziciju piksela na slici kojeg je potrebno obraditi, a kao izlaz vraća vrednost piksela obrađenog adaptivnim medijanskim filtrom. Promenljiva dim se najpre postavi na početnu vrednost *dim min* = 3, nakon čega počinje izvršavanje bloka A. U bloku A se najpre izdvaja okolina određene dimenzije i izračunava minimum, maksimum i medijana te okoline. Zatim se

Slika 2. Prikaz slike sa visokom koncentracijom impulsnog šuma i efekta primene običnog medijanskog i adaptivnog medijanskog filtra na tu sliku: A. Slika sa impulsnim šumom koncentracije 40%; B. Slika sa impulsnim šumom koncentracije 40% filtrirana medijanskim filtrom 3 ×3; C. Slika sa impulsnim šumom koncentracije 40% filtrirana medijanskim filtrom 7 ×7; D. Slika sa impulsnim šumom koncentracije 40% filtrirana adaptivnim medijanskim filtrom.

Figure 2. Display of an image with high impulse noise concentration and the effects of applying regular median and adaptive median filters on that image: A. Image with 40% impulse noise; B. Image with 40% impulse noise filtered with median filter 3 ×3; C. Image with 40% impulse noise filtered with median filter 7 ×7; D. Image with 40% impulse noise filtered with adaptive median filter.

proverava da li je vrednost medijane između minimalne i maksimalne vrednosti u toj okolini. Ukoliko se u okolini nalazi previše nereprezentativnih piksela, ovaj uslov neće biti biti ispunjen i dimenzija okolina će biti povećavana sve dok se udeo nereprezentativnih piksela ne smanji, ili se dostigne postavljeno ograničenje dimenzije okoline *dim max*. S druge strane, ukoliko je uslov ispunjen, u bloku B se proverava da li je nijansa posmatranog piksela između minimalne i maksimalne vrednosti. Ukoliko jeste, smatra se da taj piksel ne predstavnja šum i njegova vrednost se ne menja. U suprotnom, piksel predstavlja impulsni šum i njegova vrednost se menja medijanom okoline.

Adaptivni medijanski filtar uspešno filtrira slike sa visokom koncentracijom impulsnog šuma, smanjuje ukupan broj operacija po filtriranju pojedinačnog piksela i deluje samo na nereprezentativne piksele, dok ostatak slike ostavlja nepromenjenim, rezultujući bržim izvršavanjem i znatno boljim kvalitetom krajnje slike (Hwang i Haddad 1995; Vasicek i Sekanina 2008) (slika 2D).

Predložena arhitektura

Algoritmi opisani u prethodnim odeljcima nisu pogodni za izvršavanje na procesorima opšte namene, jer zahtevaju stalnu komunikaciju sa velikim delom radne memorije. Za sliku dimenzija $W \times H$ i kernel dimenzije $K \times K$, prilikom linearnog i medijanskog filtriranja potrebno je ukupno $W \cdot H \cdot K^2$ pristupa različitim lokacijama u memoriji, što znači da se svi pikseli sa slike učitavaju K^2 puta, umesto samo jedanput.

Zbog njihove velike primene u otklanjanju šuma i neefikasnog izvršavanja na procesorima opšte namene, smatramo da vremenski kritčni sistemi mogu da imaju znatne koristi ubrzavanjem gore navedenih algoritama. Iz tih razloga, cilj ovog rada je dizajniranje, implementacija i testiranje specijalizovanog modula za linearno filtriranje slike, kao i dve specijalizovane arhitekture za adaptivno medijansko filtriranje i njihovo međusobno poređenje.

Glavna karakteristika predloženih arhitektura, koja im omogućava brzu i efikasnu obradu slike, jeste protočnost. Protočnost podrazumeva deljenje glavnog procesa koji se izvršava na više manjih, međusobno nezavisnih podprocesa, koji se izvršavaju jedan za drugim. Svaki od ovih podprocesa je implementiran kao zasebna komponenta koja vrši samo taj specifični podproces. Pošto su podprocesi međusobno nezavisni, sve komponente mogu da rade paralelno, vršeći svoje specifične podprocese na raziličitim delovima slike.

Kada određena komponenta završi svoj deo obrade, ona svoj rezultat prosleđuje sledećoj komponenti i prima rezultat od prethodne komponente. Na ovaj način omogućeno je maksi-

malno iskorišćenje dostupnih resursa, pošto nijedan deo arhitekture nije u praznom hodu, kao i konstantan protok podataka kroz arhitekturu koji uklanja potrebu za višestrukim čitanjem istih podataka, što smanjuje opterećenje na sporu sistemsku memoriju, i samim tim omogućava brzo izvršavanje.

Field Programmable Gate Array (FPGA)

Radi dizajniranja i testiranja specijalizovanog čipa opisanog u ovom radu, upotrebljena je posebna vrsta integrisanog kola – Field Programmable Gate Array ili FPGA, koja unutar sebe sadrži veliki broj adaptivnih logičkih blokova ili ALM-ova (eng. Adaptive Logic Module) koji ponasob mogu da vrše osnovne logičke operacije i da čuvaju nekoliko bitova podataka. Uz pomoć jezika za opis hardvera (eng. hardware description language) moguće je povezati ove blokove u strukture koje izvršavaju kompleksne funkcije. FPGA integrisana kola su posebno podobna za brzu izradu prototipova i niskotiražne aplikacije, jer se konfiguracija povezivanja potkomponenti može menjati po potrebi, u vremenu manjem od par sekundi.

Opis arhitekture modula za filtriranje slike

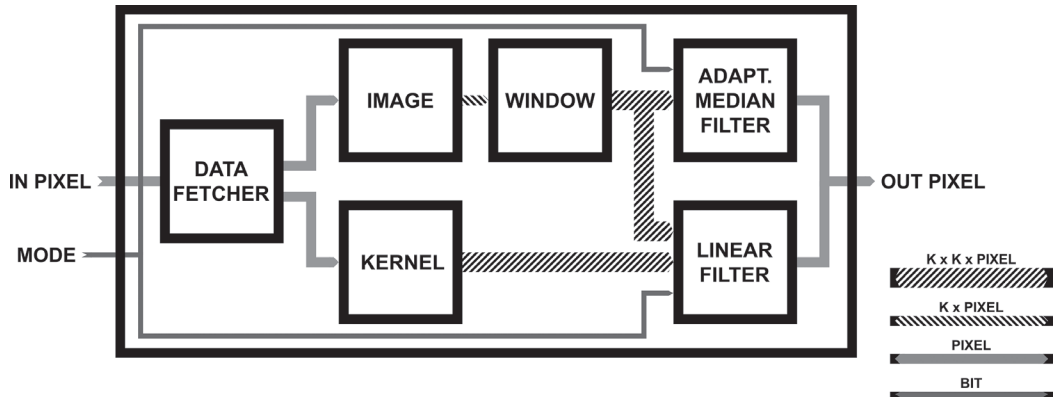
Kao što je već rečeno u prethodnom poglavlju, opisana arhitektura se sastoji od više specijalizovanih komponenti ili podmodula, koji izvršavaju neke osnovne zadatke. Na slici 3 se nalazi opšti pregled opisane arhitekture, uključujući sve komponente koje je čine i signale kojima komuniciraju. U daljem tekstu je opisan glavni interfejs modula za filtriranje slike, kao i svaka pojedinačna komponenta.

Predprocesiranje i interfejs

Pored osnovnih *Clock*, *Enable* i *Ready* signala, interfejs modula za filtriranje slike čine: ulazni jednobitni *Mode* signal kojim se određuje algoritam za filtriranje slike koji će modul izvršiti, kao i osmobitni *In pixel* i *Out pixel* signali kojima modul redom učitava i ispisuje celobrojne vrednosti piksela. Vrednosti piksela slike koja se filtrira moraju se unositi sekvencijalno, red po red, jedan piksel po taktu. Nakon određenog fiksiranog kašnjenja modul na svom izlazu generiše vrednosti filtriranih piksela u istom redosledu kao što su uneti. Maksimalna dimenzija okoline oba algoritma za filtriranje jednaka je K.

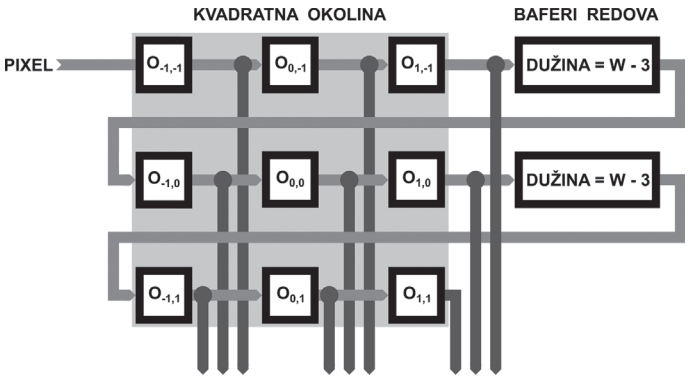
Zbog hardverskih ograničenja ulazna slika mora imati fiksiranu širinu W, dok joj visina H može biti proizvoljna. Ukoliko je potrebno obraditi sliku veće širine, ona se može razdeliti na isečke širine W, pri čemu se svaki naredni isečak preklapa sa prethodnim za $(K - 1)/2$ kolona. Pošto oba algoritma za filtriranje slike smanjuju dimenzije obrađivane slike, korisnik mora da popunjava (padding) ulaznu sliku pre unosa u modul.

Ukoliko modul izvršava linearno filtriranje potrebno je da korisnik, u prvih $K \cdot K$ taktova funkcionisanja modula, unese celobrojne osmobitne



Slika 3. Pregled arhitekture modula za filtriranje slike

Figure 3. Architecture overview of the image processing module



Slika 4. Primer *Row buffer* arhitekture za klizajući prozor dimenzije 3 ×3 (adaptirano prema: Vasicek i Sekanina 2008)

Figure 4. Example of a *Row buffer* architecture for a 3 ×3 sliding window (adapted according to: Vasicek & Sekanina 2008)

vrednosti kernela kojim će se slika filtrirati. Filtriranje kernelom dimenzije manje od K moguće je postići popunjavanjem tog kernela nulama na određenim pozicijama.

Data fetcher

Data fetcher je prosta komponenta koja, ukoliko modul izvršava linearno filtriranje, prosleđuje prvih $K \cdot K$ učitanih piksela (vrednosti kernela) u komponentu *Kernel*, a svaki naredni učitani piksel prosleđuje komponenti Image. Ukoliko modul izvršava adaptivno medijansko filtriranje ova komponenta sve učitane piksele prosleđuje komponenti Image.

Kernel

Komponenta *Kernel* sadrži pomerački registar (eng. shift register) sa $K \cdot K$ koeficijenata. Nakon što se pomerački registar napuni vrednostima prosleđenim iz komponente *Data fetcher*, signal spremnosti ove komponente postaje pozitivan. Komponenta *Linear filter* nakon toga može, po potrebi, direktno pristupati svim koeficijentima koji se nalaze unutar ove komponente.

Image i Window

Najkorišćenija hardverska implementacija funkcije klizajućeg prozora jeste takozvana *Row buffer* arhitektura (Vasick i Sekanina 2008), koja aktivno čuva K redova obrađivane slike u pomeračkom registru dimenzija $W \cdot K$. Kada se registar napuni, vrednosti unutar prvih K kolona registara predstavljaju kvadratnu okolinu piksela O u centru tog kvadrata. Prilikom svakog sledećeg učitavanja piksela u pomerački registar, prvih K kolona registara poprimaju vrednosti klizajućeg prozora pomerenog za jedan na desno.

Row buffer arhitektura je, uz male varijacije, implementirana u modulu za filtriranje slike kroz komponente *Image* i *Window*. Komponenta Image sadrži pomerački registar širine W i visine K. Od trenutka kada se prva kolona registara popuni, vrednosti unutar te kolone se prosleđuju komponenti Window.

Komponenta *Window* se sastoji od K pomeračkih registara dužine K u kojima se čuvaju poslednjih K učitanih kolona klizajućeg prozora. Od trenutka kada se ovi registri popune vrednosti unutar njih predstavljaju kvadratnu okolinu piksela koji se nalazi u centru tog kvadrata.

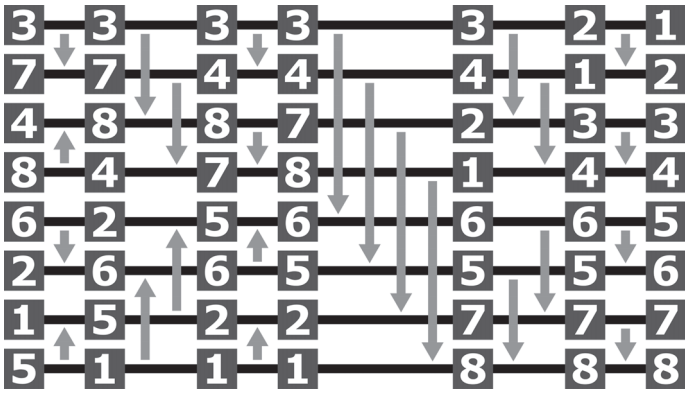
Kada se dostigne kraj reda slike, u narednih $K-1$ ciklusa signal spremnosti je postavljen na logičku nulu, jer se u tom periodu unutar pomeračkih registara istovremeno nalaze vrednosti iz trenutnog i narednog reda slike, usled prelaska u novi red. Prelazak u novi red predstavlja jedini period zastoja kada modul na svom izlazu ne generiše sledeći obrađeni piksel iako je na ulaz doveden novi piksel za obradu.

Linear filter

Komponenta *Linear filter* obavlja sve aritmetičke operacije vezane za linearno filtriranje. Komponenta množi, element po element, matricu težina kernela pribavljenu iz komponente *Kernel* sa matricom okoline trenutnog piksela, čije se vrednosti nalaze u komponenti *Window*. Izračunati proizvodi se zatim sumiraju i podele sumom vrednosti unutar kernela. Dobijena vrednost predstavlja filtriranu vrednost centra trenutno posmatrane okoline.

Adaptive median filter

Prilikom adaptivnog medijanskog filtriranja potrebno je pronaći medijanu, minimum i ma-



Slika 5. Bitonička sortirajuća mreža za niz dužine 8

Figure 5. Bitonic sorting network for an array of a length 8

ksimum određenog skupa vrednosti. Efikasno pronalaženje ovih vrednosti je netrivijalan zadatak, jer se u hardveru ne može svesti na jednostavan sled aritmetičkih operacija. U ovom radu su predstavljene dve potpuno protočne arhitekture za pronalaženje minimuma, medijane i maksimuma: bitoničko sortiranje i kumulativni histogram.

Oba algoritma su implementirana da istovremeno obezbede medijanu, minimum i maksimum za sve moguće dimenzije okoline piksela – od 3 do K, jer je nemoguće unapred znati na kojoj dimenziji će se algoritam za adaptivno medijansko filtriranje zaustaviti. Sve ove vrednosti se dalje prosleđuju prostom logičkom bloku koji primenjuje algoritam adaptivnog medijanskog filtriranja i odlučuje koja vrednost se generiše na izlazu modula.

Bitoničko sortiranje

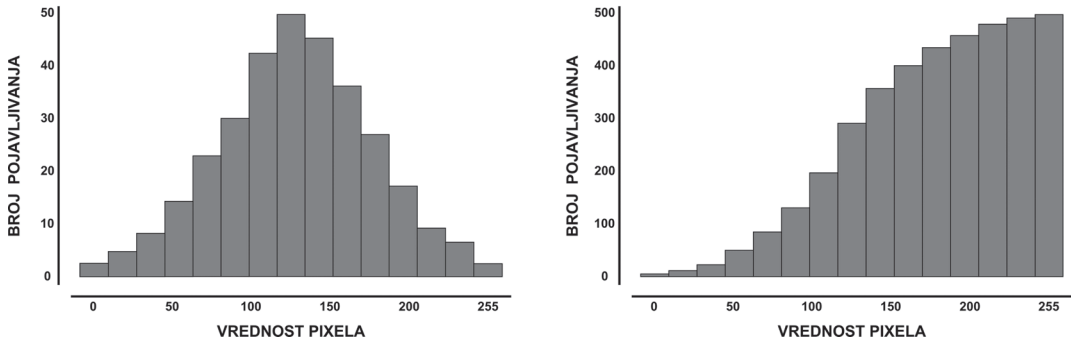
Jedan način pronalaženja medijane, minimuma i maksimuma određenog skupa vrednosti jeste da taj skup sortiramo, nakon čega medijana, minimum i maksimum predstavljaju redom srednju, prvu i poslednju vrednost u sortiranom nizu. Hardverska implementacija sortiranja postiže se takozvanim sortirajućim mrežama (eng. sorting network) koje se sastoje od fiksnog broja compare&swap (CS) modula koji porede dve vrednosti i menjaju im mesta ukoliko je to potrebno (Knuth 1997). Glavna karakteristika sortirajućih mreža jeste da je sekvenca CS operacija fiksirana, unapred određena i potpuno nezavisna od vrednosti ulaznog niza. Kao takve, sortirajuće mreže su veoma pogodne za paralelno izvršavanje, i samim tim protočnu hardversku implementaciju (Vasicek i Sekanina 2008; Knuth 1997).

Bitoničko soritranje je često korišćen algoritam za sortiranje pomoću sortirajućih mreža. Ovaj algoritam se zasniva na manipulaciji ulaznog niza u bitonički niz – niz koji samo jednom menja monotonost, i koji je zbog toga lak za sortiranje. Kako bi uneti niz od n elemenata postao bitonički, algoritam rekurzivno poziva samog sebe i sortira prvu polovinu niza u rastućem poretku, a drugu polovinu niza u opadajućem poretku. Nastali bitonički niz se zatim sortira funkcijom bitoničko spajanje koja primenjuje CS operaciju na svaki i-ti element sa $i + m$-tim elementom, gde je početno $m = n/2$, a zatim polovi m i ponovno vrši poređenje i premeštanje, sve dok m ne dostigne vrednost 1. Na slici 5 predstavljen je primer bitoničke soritrajuće mreže koja sortira niz od 8 vrednosti. Strelice na slici predstavljaju CS operacije koje na vrh strelice postavljaju veću od dve upoređene vrednosti.

Sortirajuća mreža, primenjena u komponenti za adaptivno medijansko filtriranje, u svakom ciklusu prima jedan niz dužine 2^k, a nakon prvobitnih $k \cdot (k + 1)/2$ ciklusa popunjavanja mreže, na izlazu daje sortirani niz u svakom ciklusu, čime je postignut konstantan protok podataka.

Kumulativni histogram

Drugi pristup za pronalaženje medijane, minimuma i maksimuma svodi se na pronalaženje određenih stubaca na kumulativnom histogramu vrednosti unutar klizajućeg prozora. Za razliku od običnog histograma, visina stubaca kod kumulativnog histograma predstavlja broj piksela čija je vrednost manja ili jednaka indeksu tog stupca (slika 6).



Slika 6. Običan histogram (levo) i njegov odgovarajući kumulativni histogram (desno)

Figure 6. Regular histogram (left) and its corresponding cumulative histogram (right)

Prvi stubac čija je vrednost veća ili jednaka broju 1, sredini broja piksela ($((K^2 - 1)/2 + 1)$ i ukupnom broju piksela (K^2), redom označavaju minimum, medijanu i maksimum vrednosti unutar klizajućeg prozora.

Implementacija kumulativnog histograma je izdeljena u više celina: prioritetni dekoder, sumator histograma i prioritetni enkoder.

Prioritetni dekoder. Prioritetni dekoder za svaki od K^2 piksela klizajućeg prozora dekodira vrednost tog piksela u $L = 2^8$ pojedinačnih jednobitnih signala, pri čemu svi signali sa indeksom manjim ili jednakim vrednosti tog piksela bivaju postavljeni na logičku jedinicu, a u suprotnom na logičku nulu. Dakle, ukoliko je vrednost piksela jednaka P, od L jednobitnih signala, $P + 1$ će biti pozitivni, dok će ostali biti negativni.

Sumator histograma. Sumator histograma prima $K^2 \cdot L$ jednobitnih signala iz prioritetnog dekodera i sabira sve signale sa istim indeksima. Rezultat ovog sabiranja je L osmobitnih signala koji predstavljaju stupce kumulativnog histograma klizajućeg prozora.

Prioritetni enkoder. Nakon toga, prioritetni enkoder, po ranije navedenom kriterijumu, pronalazi koji stupci histograma odgovaraju medijani, minimumu i maksimumu i njihove indekse postavlja kao vrednosti izlaza komponente kumulativni histogram.

Rezultati i diskusija

Svaka pojedinačna komponenta je sintetisana i analizirana pomoću Quartus Prime razvojnog okruženja. Potom je svaka komponenta simulirana i testirana pomoću ModelSim okruženja za simulaciju. Nakon toga su sve komponente povezane i testiran je rad celokupnog modula. Za potrebe verifikovanja ispravnog funkcionisanja opisanog modula implementirani su referentni algoritmi u programskom jeziku Python. Rezultati pokazuju da se izlaz simuliranog modula podudara sa izlazom referentnih algoritama.

Svaka komponenta je dizajnirana da radi sa isečkom slike širine $W = 256$ i maksimalnom dimenzijom kernela $K = 15$. Kerneli dimenzija K većih od 9 se ređe viđaju u praksi, zbog relativno dužeg vremena izvršavanja na procesoru i činjenice da se slični efekti mogu postići primenom više kernela manjih dimenzija. Uprkos tome naša arhitektura je dizajnirana da radi sa kernelima do dimenzije 15, radi testiranja granica celog dizajna i poređenja arhitektura za adaptivno medijansko filtriranje na širem spektru maksimalnih dimenzija prozora (od 5 do 15).

Specifikacije svih komponenti, osim komponente za adaptivno medijansko filtriranje, dati su u tabeli 1.

U ovoj tabeli zauzeće adaptivnih logičkih blokova i zauzeće registara odnose se na hardversko zauzeće komponenti. Kao što je već pomenuto u uvodu, adaptivni logički blokovi su osnovni gradivni elementi FPGA čipa. Uređaji 5. generacije (Aria V, Cyclone V, Stratix V) u proseku sadrže 30 000 ALM-ova koji se sastoje od 4 registara, 2 ćelije za kombinacionu logiku i dva sabirača.

Komponenta *Data fetcher*, zbog proste funkcije koju vrši, očekivano zauzima jako malo resursa. Komponente *Kernel* i *Window* koje obe čuvaju i prosleđuju $K \cdot K$ osmobitnih vrednosti,

Tabela 1. Zauzeće hardverskih resursa svih komponenti, osim komponente za adaptivno medijansko filtriranje

Komponenta	Zauzeće ALM-ova	Zauzeće registara
Image	7 177	28 707
Window	906	1 809
Linear filter	1 076	17
Kernel	910	1 809
Data fetcher	11	26
Sve zajedno	14 414	34 011

očekivano zauzimaju isti broj registara i sličan broj ALM-ova. Od njih malo veće zauzeće ALM-ova ima komponenta *Linear filter*, koja ne čuva nikakve vrednosti, ali vrši veliki broj operacija množenja i sabiranja, kao i jednu operaciju deljenja, koja je hardverski jako zahtevna. Komponenta *Image,* koja po dizajnu čuva $K \cdot W$ osmobitnih vrednosti, očekivano zauzima najveći broj registara (otprilike W/K puta više nego *Window*), i samim tim značajno više ALM-ova. Očekuje se da će konačno zauzeće resursa celokupne arhitekture za linearno filtriranje biti veće od sume svih pojedinačnih komponenti, zbog utrošenja resursa na povezivanje i komunikaciju između komponenti, što je i potvrđeno nakon sintetisanja cele arhitekture.

U tabelama 2 i 3 date su specifikacije komponenti *Bitonik sort* i *Kumulativni histogram* za različite veličine K. Na osnovu podataka iz tabela može se zaključiti da komponenta *Kumulativni histogram* zauzima znatno više računarskih resursa od komponente *Bitonik sort*. Za K od 5 do

Tabela 2. Zauzeće hardverskih resursa komponente *Bitonik sort* za različite maksimalne veličine prozora pri adaptivnom medijanskom filtriranju

Maksimalna veličina prozora K	Zauzeće ALM-ova	Zauzeće registara
15	125 194	159 378
13	81 316	103 802
11	43 631	55 546
9	25 144	32 210
7	9 944	12 916
5	3 217	4 108

Tabela 3. Zauzeće hardverskih resursa komponente *Kumulativni histogram* za različite maksimalne veličine prozora pri adaptivnom medijanskom filtriranju

Maksimalna veličina prozora K	Zauzeće ALM-ova	Zauzeće registara
15	225 752	184 858
13	152 324	125 400
11	96 622	80 230
9	56 279	47 556
7	28 845	25 082
5	11 870	11 024

7, koji imaju najveću praktičnu primenu (Vasicek i Sekanina 2008), primećuje se da komponenta *Kumulativni histogram* zauzima približno duplo više registara i tri puta više ALM-ova od komponente *Bitonik sort*.

Pored zauzeća hardverskih resursa, testirana je i maksimalna podržana učestanost *Fmax* signala takta, pri kojoj ne dolazi do neočekivanog ili nedefinisanog ponašanja celokupne arhitekture. Ovu vrednost procenjuje razvojno okruženje Quartus Prime tako što simulira rad arhitekture pri različitim frekvencijama na FPGA čipu koji radi na naponu od 1100 mV, i koji je temperature 85 stepeni celzijusa. Pošto FPGA čipovi, kao i većina digitalnih električnih kola, radi brže i bolje pri visokim radnim naponima i nižim temperaturama, granice rada naše arhitekture testirane su u najgorim mogućim uslovima: nizak radni napon i visoka temperatura. Očekuje se da *Fmax* zavisi od broja operacija koje arhitektura vrši, kao i količine podataka koju svaka komponenta prima i šalje po taktu, pošto je potrebno da svi ulazni i izlazni signali budu sinhronizovani. *Fmax* je procenjen za tri varijacije predložene arhitekture: arhitektura koja vrši samo linearno filtriranje, arhitektura koja vrši samo adaptivno medijansko filtriranje bitoničkim sortiranjem i arhitektura koja vrši samo adaptivno medijansko filtriranje kumulativnim histogramom (tabela 4).

Tabela 4. Maksimalna podržana učestanost za predložene arhitekture

Arhitektura	Učestanost *Fmax* [MHz]
Linearno filtriranje	11.64
Adaptivno medijansko filtriranje bitoničkim sortiranjem	145.79
Adaptivno medijansko filtriranje kumulativnim histogramom	122.87

Primećuje se da je maksimalna podržana učestanost arhitekture za linearno filtriranje značajno niža od arhitektura za adaptivno medijansko filtriranje. Pošto se sve tri arhitekture velikim delom preklapaju, pretpostavlja se da glavni faktor koji utiče na varijacije u *Fmax* vrednostima nije celokupan dizajn arhitekture, niti neefikasan protok podataka kroz istu. Iz toga sledi da glavni doprinos jako niskoj *Fmax* vrednosti arhitekture za linearno filtriranje ima neefikasna implementacija komponente *Linear filter*, koja u jednom taktu izvršava veliki broj prostih, sekvencijalnih i međusobno nezavisnih operacija, što se kosi sa originalnom idejom protočnosti arhitekture. Očekuje se da se *Fmax* arhitekture za linearno filtriranje može značajno povećati podelom komponente *Linear filter* na više prostijih podkomponenti, što je plan za budući rad.

Između arhitektura za adaptivno medijansko filtriranje, bitoničko sortiranje dozvoljava veću vrednost *Fmax* od kumulativnog histograma. Ovaj rezultat je očekivan, s obzirom da komponenta za bitoničko sortiranje u svakom taktu barata sa manjim brojem vrednosti ($K \cdot K$ u poređenju sa $K \cdot K \cdot 2^8$), i izvršava relativno prostije operacije (compare&swap u poređenju sa sabiranjem).

Zaključak

U ovom radu prikazana je nova protočna, hardverska implementacija modula za linearno i adaptivno medijansko filtriranje slike. Prikazana arhitektura je implementirana na FPGA čipu, nakon čega je potvrđena njena funkcionalnost. Arhitektura je sposobna da filtira slike proizvoljnih dimenzija brzinom 145.79 miliona piksela u sekundi, što je ekvivaletno 70 slika po sekundi pri punoj HD (eng. Full HD) rezoluciji. Predložene su i testirane dve različite arhitekture adaptivnog medijanskog filtriranja, bitoničko sortiranje i kumulativni histogram, od kojih se arhitektura sa bitoničkim sortiranjem pokazala kao bolja po zauzeću registara, aritmetičko-logičkih jedinica, kao i po brzini izvršavanja.

Pronađena je i predstavljena mana u predloženom dizajnu komponente za linearno filtriranje, koja podrazumeva sekvencijalno izvršavanje više operacija u jednom taktu i koja značajno ograničava maksimalnu podržanu učestanost arhitekture. Budućim redizajniranjem komponente za linearno filtriranje očekuje se bitno povećanje brzine izvršavanja modula.

Literatura

Farooque M. A., Rohankar J. S. 2013. Survey on various noises and techniques for denoising the color

image. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, **2** (11): 217.

Sobel I., Feldman G. 1968. A 3 ×3 isotropic gradient operator for image processing. A talk at the Stanford Artificial Project. U *Pattern Classification and Scene Analysis* (ur. R. Duda i P. Hart). Wiley, str. 271–272.

Hwang H., Haddad R. A. 1995. Adaptive median filters: new algorithms and results. *IEEE Transactions on image processing*, **4** (4): 499.

Vasicek Z., Sekanina L. 2008. Novel hardware implementation of adaptive median filters. U *2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (XI), April, Proceedings*. IEEE, str. 1-6.

Knuth D. E. 1997. *The art of computer programming* (Vol. 3). Pearson Education

---

Mihailo Grbić and Filip Parag

Pipelined FPGA Architecture for Image Processing with Linear and Adaptive Median Filter

This paper describes a pipelined architecture for digital image processing with a linear and adaptive median filter on FPGA (Field-programmable gate array) chips. These algorithms achieve remarkable results in noise removal, specifically Guassian noise for linear filtering and impulse noise for adaptive median filtering. As such, they are often used by all systems which incorporate a camera, right after an image is obtained from the camera sensor. While these and many other similar image processing algorithms can be implemented on a regular general-purpose processor, such implementations are relatively time expensive, due to the fact that they require multiple readings of the same data from memory. This problem is most evident in time-critical or real-time video processing systems, which often have to choose between lowering the image/video resolution, reducing the framerate, or avoiding these algorithms entirely.

In order to avoid this compromise, more and more systems are starting to use specialized hardware modules that quickly and efficiently execute necessary denoising image processing algorithms, with minimal load on system memory. This paper describes one such module for linear and adaptive median filtering.

The pipelined nature of the proposed architecture entails the division of a master process, that is executed into several smaller, mutually independent subprocesses, which are executed one after the other. Each of these subprocesses is implemented as a separate component that performs only that specific subprocess. Because the subprocesses are independent of each other, all components can operate in parallel, performing their specific subprocesses on different parts of the image. When a component completes its task, it passes its result to the next component and receives the result from the previous component. This enables maximum utilization of available resources, since no part of the architecture is idle, as well as a constant flow of data through the architecture, which eliminates the need for multiple readings of the same data, reducing the load on slow system memory and thus allowing much faster execution.

Adaptive median filtering requires finding the minimum, median and maximum of an array of values. As this task is relatively hard to execute in hardware, two different methods of finding the minimum, median and maximum values, were proposed and compared. The *Bitonic sort* method finds these values by sorting the array and then taking the first, middle, and last value respectively. The *Cumulative histogram* method constructs a cumulative histogram out of this array of values, and then finds the first bars which contain one, half, and all values from the array respectively. Out of the two proposed methods, bitonic sorting proved superior, taking up 50% fewer registers and logic blocks and achieving a 20% higher maximum allowed frequency.

The whole architecture was synthesized and behaviorally simulated to confirm its functionality. The proposed module shows significant promise for real-world use in time-critical image processing systems.