Detekcija lica na slici pomoću FPGA

Živković, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij Elektrotehnike

DETEKCIJA LICA NA SLICI POMOĆU FPGA

Diplomski rad

Marko Živković

Osijek, 2020

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 28.09.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Marko Živković
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1212, 19.09.2019.
OIB studenta:	68270051732
Mentor:	Izv. prof. dr. sc. Marijan Herceg
Sumentor:	Leon Šneler
Sumentor iz tvrtke:	Dario Pović
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Mario Vranješ
Član Povjerenstva 1:	Izv. prof. dr. sc. Marijan Herceg
Član Povjerenstva 2:	Izv.prof.dr.sc. Ratko Grbić
Naslov diplomskog rada:	Detekcija lica na slici pomoću FPGA
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. Polje elektrotehnika)**
Zadatak diplomskog rada:	Algoritmi za detekciju lica na slici pronalaze svoju primjenu u širokom spektru industrijske djelatnosti kao što su automobilska industrija, industrija računalnih igara, propaganda, vojna industrija i druge. U sklopu ovog diplomskog rada potrebno je izraditi algoritam za detekciju lica na slikama različitih rezolucija, koristeći pritom FPGA za digitalnu obradu prethodno učitane slike u radnu memoriju FPGA. Nakon razvoja algoritma u Matlabu, potrebno je izraditi digitalni dizajn u VHDL jeziku za opis fizičke arhitekture, a zatim izvršiti njegovu sintezu i optimizaciju u ZYNQ SoC FPGA. Nakon uspješne implementacije algoritma u FPGA, potrebno je izvršiti mjerenja svojstava implementiranog algoritma koja uključuju broj ispravno i pogrešno detektiranih lica na slici, brzinu obrade slike s obzirom na njezinu rezoluciju te potrošnju električne energije pojedine periferije FPGA prilikom izvođenja algoritma. Dobivene rezultate potrebno je usporediti s rezultatima javno dostupnih postojećih rješenja za istu namjenu.(tema rezervirana za Marko Živković), (sumentor: Dario Pović, Institut RT-RK Osijek d.o.o., Leon Šneler, FERIT)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	28.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 29.09.2020.

Ime i prezime studenta:	Marko Živković
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1212, 19.09.2019.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Detekcija lica na slici pomoću FPGA**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Marijan Herceg

i sumentora Leon Šneler

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Algoritmi za detekciju lica omogućuju digitalnim sustavima određivanje postojanja lica na slici, što otvara nove mogućnosti za primjenu digitalnih sustava. Izvođenje algoritama za detekciju lica na slici predstavlja prvi potreban korak za algoritam prepoznavanja lica na slici. Navedeni algoritmi zajedno čine cjelinu koja nas okružuje u svakodnevnom životu. Takvi algoritmi poboljšavaju kvalitetu života, omogućuju veću automatizaciju sustava, te biometrijsku autentifikaciju ljudi. Zbog navedenoga, algoritmi za detekciju i prepoznavanje lica na slici pronalaze svoju primjenu u širokom spektru industrijske djelatnosti kao što su automobilska industrija, industrija zabave, propaganda, vojna industrija i druge.

Naglim razvojem komunikacijskih tehnologija i prijenosnih uređaja, zahtjevi potrošača postaju sve veći. Najveću manu prijenosnih uređaja, predstavljaju energetski izvori navedenih uređaja (baterije). Zbog ograničene količine energije energetskog izvora i brzog razvoja tehnologije, zahtjevi su da novi prijenosni uređaji trebaju biti energetski efikasniji, ali i brži nego ikada prije. Jedan od čimbenika održavanja većih brzina prijenosnih uređaja uz što veću efikasnost se može postići tako da se rastereti procesor koji je jedan od većih potrošača energije unutar prijenosnih uređaja.

Procesor služi za više namjensku obradu podataka unutar prijenosnog uređaja. Prilikom velikog broja zahtjeva, procesor se zagrijava, te postaje manje efikasan. Opterećeni procesor se može rasteretiti tako da neke od zahtjeva direktno obrađuje integrirano programibilno polje logičkih sklopova (engl. *Field-Programmable Gate Array, FPGA*), koji zbog svoje primjene na pojedine zahtjeve zadržava jednostavnost dizajna za razliku od procesora, a uz to i veću brzinu obrade i bolju energetsku efikasnost.

Ostatak rada je strukturiran na slijedeći način. U drugom poglavlju slijedi upoznavanje sa *Viola-Jones* algoritmom [1] koji se koristi za brzu detekciju objekta na slici, te omogućuje detekciju objekta u stvarnom vremenu. U trećem poglavljem opisano je predloženo rješenje algoritma za detekciju lica na slici pomoću FPGA. U četvrtom poglavlju prikazani su rezultati testiranja predloženog rješenja te je u petom poglavlju dan kratak zaključak.

2. OPĆENITO O PROCESU DETEKCIJE LICA NA SLICI

Digitalni proces koji svojim postupkom određuje postojanje lica na zadanom slici (fotografiji ili video snimci) naziva se detekcija lica. Pojmovi detekcija lica na slici i prepoznavanje lica na slici se razlikuju. Glavno pitanje kojim se bave postupci detekcije lica na slici je „Je li na zadanoj slici prikazano lice?“, dok je glavno pitanje kojim se bave postupci prepoznavanje lica na slici „Čije je lice prikazano na zadanoj slici?“.

Iako ljudi brzo uoče lice, sam postupak digitalne detekcije lica nije toliko jednostavan. Takav postupak zahtjeva obradu velikih količina podataka što rezultira usporavanjem rada digitalnog sustava i otežavanjem izvođenja algoritma u stvarnom vremenu. Prilikom dizajniranja navedenog sustava, ovisno o primjeni, potrebno je optimizirati brzinu izvođenja i kvalitetu izvođenja digitalnog postupka detekcije lica. Kako bi se ubrzalo vrijeme izvođenja algoritma digitalne detekcije lica, poželjno je paralelizirati izvođenje postupka detekcije, što može povećati cijenu i energetsku potrošnju samog sustava.

2.1. Viola-Jones algoritam

U radu [1] predložen je algoritam pod nazivom „*Viola-Jones“* za brzu detekciju bilo kojeg objekta korištenjem ubrzanih kaskada složenih od jednostavnih značajki (engl. *Rapid Object Detection using a Boosted Cascade of Simple Features*). Navedeni algoritam je prvi puta objavljen 2001. godine, ali je još uvijek aktualan zbog brzine detekcije objekta i mogućnosti primjene algoritma za detekciju objekata na slici u stvarnom vremenu.

Paul Viola i Michael Jones su ovim radom po prvi puta uspjeli detektirati lice u vremenu od 0,07 sekundi na fotografiji razlučivosti 384x288 piksela koristeći procesor „*Intel Pentium III*“, što je omogućilo detekciju objekta na slici u stvarnom vremenu [1].

Viola-Jones algoritam se sastoji od sljedeća četiri dijela:

1. Integralna slika (engl. *integral image*) [1]
2. *Haarove* značajke (engl. *Haar-like features*) [1]
3. Algoritam za prilagodljivo ubrzanje (engl. *Adaptive Boosting, AdaBoost*) [2]
4. Kaskadna klasifikacija [1]

Integralna slika predstavlja transformirani oblik izvorne slike ili dijela (podokvira) izvorne slike. Algoritam za računanje integralne slike zbraja piksele izvorne slike prema izrazu 2.1.

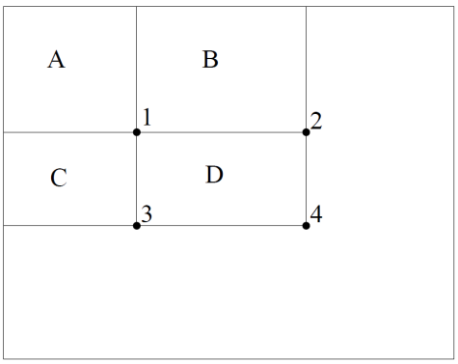
$$ii(x, y) = \sum_{x'=1,\, y'=1}^{x,y} i(x', y')\,, \tag{2.1.}$$

gdje $i(x',y')$ predstavlja vrijednost piksela na koordinatama (x',y') ulazne slike, a $ii(x,y)$ predstavlja izračunatu vrijednost piksela na koordinatama (x,y).

Integralnom slikom omogućena je obrada izvorne slike korištenjem kombinacije pravokutnih površina umjesto pojedinačnih piksela. Zbroj svih piksela unutar pravokutne površine izvorne slike može se dobiti računanjem vrijednosti četiri piksela integralne slike navedenog pravokutnog oblika po izrazu 2.2.

$$D = ii(x_1, y_1) - ii(x_2, y_2) - ii(x_3, y_3) + ii(x_4, y_4)\,, \tag{2.2.}$$

gdje je D zbroj piksela površine D, $ii(x_1, y_1)$ vrijednost piksela 1, $ii(x_2, y_2)$ vrijednost piksela 2, $ii(x_3, y_3)$ vrijednost piksela 3, a $ii(x_4, y_4)$ vrijednost piksela 4 sa slike 2.3.



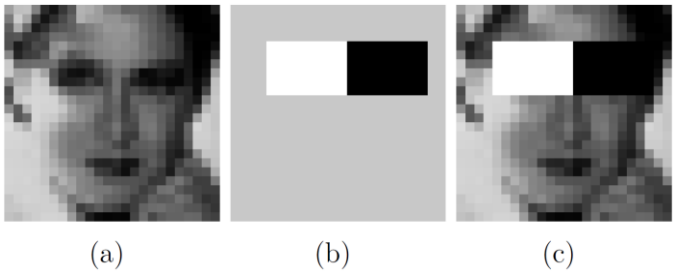
Sl. 2.3.Vizualni prikaz veze izvorne i integralne slike [1].

Na slici 2.3. prikazano je kako je pikselom 1 integralne slike obuhvaćen zbroj svih piksela izvorne slike označen površinom A, u pikselu 2 integralne slike je obuhvaćen zbroj svih piksela izvorne slike označen površinom $A+B$, u pikselu 3 integralne slike je obuhvaćen zbroj svih piksela izvorne slike označen površinom $A+C$, a u pikselu 4 integralne slike je obuhvaćen zbroj svih piksela izvorne slike označenih površina $A+B+C+D$ [1].

Nakon izračuna integralne slike iz izvorne slike, potrebno je provjeriti značajke objekta, za što se koriste *Haarove* značajke [1]. Prema [3], *Haarove* značajke predstavljaju skalarni produkt između izvorne slike i oblika *Haarove* značajke. Slijedi izraz 2.4. za računanje *Haarovih* značajki:

$$H = \sum_{x=1}^{N}\sum_{y=1}^{N} i(x,y)_{P(x,y)\ bijelo} - \sum_{x=1}^{N}\sum_{y=1}^{N} i(x,y)_{P(x,y)\ crno}\ , \quad (2.4.)$$

gdje *H* predstavlja rezultat *Haarove* značajke, *i*(*x*,*y*) predstavlja vrijednost piksela na koordinati *x*, *y*, a *P*(*x*,*y*) predstavlja oblik *Haarove* značajke na *x*, *y* koordinati izvorne slike razlučivosti *N*x*N*. *Crno* i *bijelo* predstavljaju crnu i bijelu komponentu *Haarove* značajke predočene primjerom na slici 2.5..



Sl. 2.5.Vizualni prikaz primijenjene *Haarove* značajke na slici [3].

Na slici 2.5. prikazana je primjena *Haarove* značajke na izvornoj slici. Slika *a* prikazuje izvornu sliku razlučivosti *N*x*N*, slika *b* prikazuje bijelu i crnu komponentu *Haarove* značajke na sivoj površini, te slika *c* prikazuje primijenjenu *Haarovu* značajku na izvornoj slici [3]. *Haarove* značajke mogu biti različitih veličina i oblika kako je prikazano slikom 2.6..

Sl. 2.6.Vizualni prikaz različitih *Haarovih* značajki [3].

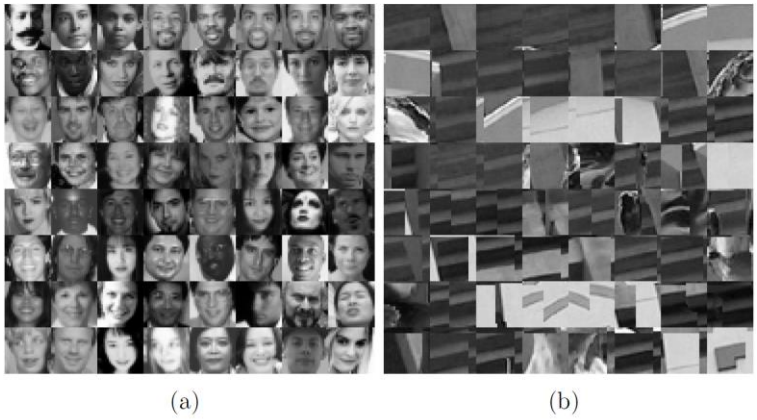
Slika 2.6. prikazuje različite oblike i veličine *Haarovih* značajki. Oblik navedenih značajki definiran je ograničenjima za svaku značajku posebno. Svakoj pojedinačnoj *Haarovoj* značajki bijela i crna komponenta trebaju biti jednake veličine, te se navedene komponente pojedinačne značajke trebaju doticati cijelim rubom. Navedenim ograničenjima postiže se manji broj kombinacija površina unutar zadanog okvira, te ubrzana obrada slike.

Haarove značajke računaju se na integralnoj slici prema [1] i izrazu 2.2., gdje se *D* računa za svaku komponentu navedene značajke. Za računanje jedne komponente *Haarove* značajke pomoću integralne slike potrebne su četiri matematičke operacije. Prilikom računanja iste

Haarove značajke pomoću zbroja pojedinih piksela potrebno je $M * N$ matematičkih operacija, gdje su *M*x*N* dimenzije *Haarove* značajke. Prema navedenom prednost računanja pomoću integralne slike se postiže računanjem površina većih od četiri piksela, kakve su mnogobrojne kod *Haarovih* značajki.

Podprozor predstavlja dio integralne slike na kojem se izvršava proces detekcije objekta. Za podprozor, veličine 24x24 piksela, u kojem se računaju *Haarove* značajke postoji preko 180.000 kombinacija navedenih značajki. U svrhu smanjenja broja ukupnih kombinacija *Haarovih* značajki koristi se *AdaBoost* algoritam čiji je zadatak istrenirati klasifikatore lokaliziranjem najbitnijih *Haarovih* značajki, te odbacivanje manje bitnih značajki [1].

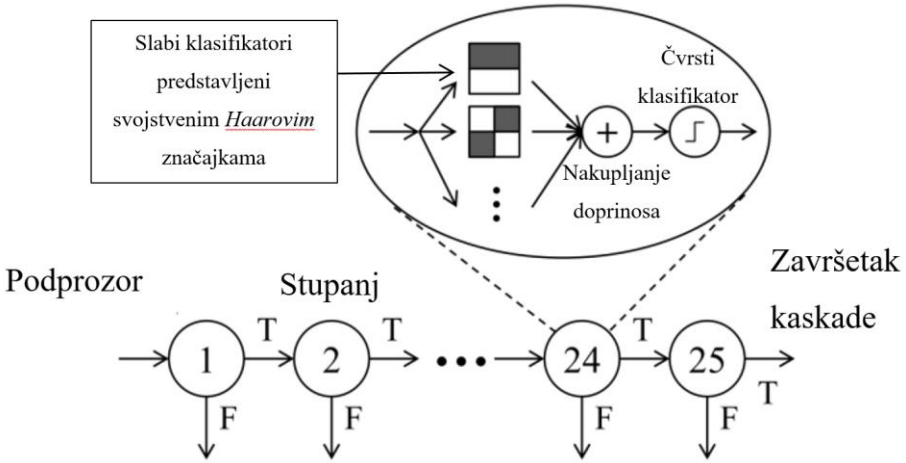
Za treniranje klasifikatora *AdaBoost* algoritmom potrebno je imati dvije baze podataka. Jedna baza podataka isključivo sa pozitivnim uzorcima (slike sa željenim objektom za detekciju), te jedna baza podataka sa negativnim uzorcima (slike bez željenog objekta za detekciju) prema primjeru sa slike 2.7..



Sl. 2.7.Vizualni prikaz pozitivnih (*a*) i negativnih (*b*) uzoraka za treniranje klasifikatora *AdaBoost* algoritmom, gdje je lice željeni objekt za detekciju [3].

Na početku algoritma inicijaliziraju se vrijednosti doprinosa svake *Haarove* značajke, na način da svaka od navedenih značajki ima jednak doprinos. Za svaku *Haarovu* značajku provjerava se koliko dobro razdvaja pozitivne i negativne uzorke. Ovisno o navedenom izvršava se korekcija doprinosa svake od navedenih značajki. Svakoj *Haarovoj* značajki s većim doprinosom, računa se slabi prag (engl. *weak threshold*) koji definira slabi klasifikator navedene značajke. Slabi prag predstavlja vrijednost uvjeta za danju klasifikaciju *Haarovih* značajki. Više slabih klasifikatora zajedno čine čvrsti klasifikator. Odbacivanjem *Haarovih* značajki koje nemaju značajan doprinos, reducira se ukupan broj navedenih značajki. Parametri dobiveni ovim algoritmom najviše utječu na točnost detekcije objekta na slici.

Dobiveni čvrsti klasifikator može razdvojiti željeni objekt od ostalih, ali ne predstavlja efikasno rješenje klasifikacije. Za navedeni čvrsti klasifikator, svako izvršavanje detekcije objekta na slici ili podokviru slike obavlja cijelu klasifikaciju, te prolazi kroz sve slabe klasifikatore bez obzira postoji li lice na slici ili ne. Kako bi se pravovremeno izbjegla obrada slike ili podokvira slike bez lica, prema [1] predložena je kaskadna klasifikacija. Navedena kaskadna klasifikacija distribuira čvrsti klasifikator po stupnjevima. Stupnjevi su izvedeni tako da svaki sljedeći stupanj sadrži više slabih klasifikatora. Početni stupanj kaskadnog klasifikatora sadrži najmanje slabih klasifikatora koji imaju najveći doprinos. Svakim sljedećim stupnjem kaskade broj slabih klasifikatora unutar stupnja se povećava.



Sl. 2.8.Vizualni prikaz kaskadnog klasifikatora [4].

Na slici 2.8. prikazana je struktura kaskadnog klasifikatora s 25 stupnjeva. Ako na stupnju dođe do istinitog stanja (engl. *True,* T), kaskada prelazi na sljedeći stupanj, a ako dođe do lažnog stanja (engl. *False,* F), kaskada javlja da je detekcija završena i da objekt neće biti detektiran.

Doprinosi slabih klasifikatora se zbrajaju do prvoga čvrstog klasifikatora, te čine akumulirani doprinos. Ukoliko na prvom stupnju akumulirani doprinos bude veći od vrijednosti prvog čvrstog praga, provjera kaskadnog klasifikatora prelazi na sljedeći stupanj. Postupak se ponavlja dok provjera kaskadnog klasifikatora ne dođe do posljednjeg stupnja koji (ako je uvjet zadovoljen) detektira objekt. Na prvom stupnju na kojem uvjet nije zadovoljen, sustav javlja da unutar trenutnog podprozora objekt neće biti detektiran. Ukoliko se unutar podprozora za detekciju ne nalazi željeni objekt, navedenim se postupkom u ranim koracima detekcije preskače cjelokupan postupak, što ubrzava performanse sustava.

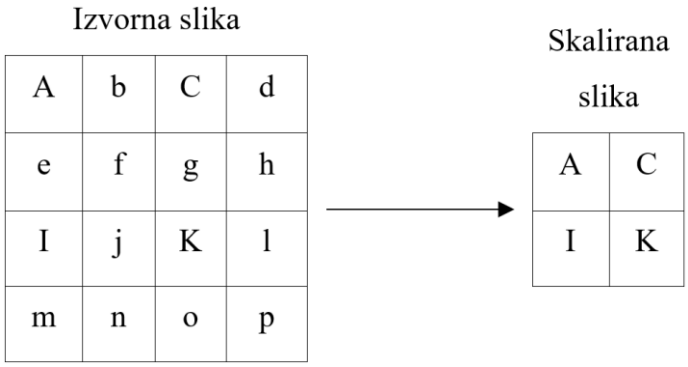
Viola-Jones algoritam može detektirati više lica na istoj slici, što omogućuje i višestruku detekciju istog lica na bliskim lokacijama slike, pa su prema [3] predstavljeni uvjeti po kojima se odbacuje redundantne detekcije lica na slici.

1. Lice je detektirano ako i samo ako je zadovoljen broj detekcija lica iste veličine na obližnjim koordinatama slike.
2. Postoji mogućnost preklapanja okvira detekcije s prethodno detektiranim okvirima. U tom slučaju postoje dva pristupa:
    a. Ako je centar manjeg okvira detekcije izvan većeg okvira detekcije, zadržavaju se oba okvira.
    b. U suprotnom, odabire se okvir s pouzdanijom detekcijom.

Zbog potrebe za obradom slika većih razlučivosti i primjene detekcije objekata u stvarnom vremenu, prema [4] je predloženo paralelno izvršavanje navedenog sustava u svrhu poboljšanja vremena detekcije objekata pri slikama veće razlučivosti. U radu je paralelizacija implementirana u dijelu digitalnog dizajna koji se smatra i najbitnijim dijelom u procesu detektiranja objekta. Zadatak navedenog dijela je na osnovu parametara dobivenih *AdaBoost* algoritmom odrediti postoji li lice unutar podprozora.

U radu se uspoređuju vremena za obradu slike jednostavnog Viola-Jones algoritma i njegove paralelne primjene na slikama razlučivosti 320x240, 640x480, i 1280x960. Detekciju lica na slici male razlučivosti 320x240 jednostavni sustav je obavljao za 0,13 sekundi, dok je paralelni sustav detekciju obavljao za 0,08 sekundi. Detekciju lica na slici srednje razlučivosti 640x480 jednostavni sustav je obavljao za 0,38 sekundi, dok je paralelni sustav detekciju obavljao za 0,23 sekundi. Detekciju lica na slici velike razlučivosti 1280x960 jednostavni sustav je obavljao za 1,43 sekundi, dok je paralelni sustav obavljao detekciju za 0,71 sekundi [4].

U radu [4] također je predstavljen algoritam za skaliranje slike koji je efikasan za digitalni dizajn. Unutar navedenog algoritma slika se skalira u ovisnosti o samo jednom pikselu unutar 4^n susjedna piksela, gdje je n nivo skaliranja slike. Skaliranje slike navedenim algoritmom postiže se manipuliranjem adresa za čitanje vrijednosti iz memorije FPGA, te ne zahtjeva dodatne matematičke operacije.



Sl. 2.9.Vizualni prikaz algoritma za skaliranje slike po pikselu.

Na slici 2.9. vizualiziran je algoritam za skaliranje slike po pikselu na kojem se vidi da su samo vrijednosti piksela izvorne slike označenih velikim slovima (*A*, *C*, *I* i *K*) preslikani u novu skaliranu sliku, dok su pikseli označeni malim slovima zanemareni prilikom skaliranja slike.

S ciljem optimizacije digitalnog dizajna detekcije objekata u radu [5] predloženo je korištenje monokromatske slike, ograničeni *AdaBoost* trening, te dva načina za implementaciju Viola-Jones algoritma na FPGA. Predloženo je korištenje monokromatske slike, jer slika u boji zauzima više bita po pikselu od monokromatske slike, te je zahtjevnija za obradu i pohranu u memoriju. Ako slika u boji koristi crveni, plavi i zeleni (engl. *Red Blue Green,* RGB) tip boja, svaka boja zauzima vrijednost od 8 bita, što u konačnici čini 24 bita informacije za samo jedan piksel. Da bi se broj bita umanjio, prema [5] RGB sliku je potrebno preobraziti u 8 bitnu monokromatsku sliku zbrajajući 29.9% crvene komponente, 58.7% zelene komponente i 11.4% plave komponente.

Kako trening kaskade *AdaBoost* algoritmom za veću preciznost sustava proizvodi veći broj parametara koje je u konačnici potrebno pohraniti u memoriju FPGA, predloženo je ograničavanje treninga *AdaBoost* algoritmom na samo jednu veličinu lica što pomaže pri redukciji izlaznih parametara. Zbog navedenog uvjeta za trening *AdaBoost* algoritmom, unutar podprozora se može detektirati samo jednu veličinu lica, te je sustav potrebno proširiti skaliranjem.

Nakon izvršenja algoritma detekcije lica cijele izvorne slike istu je potrebno ponoviti, ali na skaliranoj (umanjenoj) izvornoj slici. Takav postupak je potrebno ponavljati dok se izvorna slika više ne može skalirati, a da ne bude manja od razlučivosti podokvira integralne slike. Drugi način za proširenje sustava je da se nakon izvršenja algoritma detekcije lica cijele izvorne slike provjera ponovi, ali se skalira (uvećava) podprozor unutar kojeg se izvršava detekcija lica skupa

sa pripadnim parametrima. Takav postupak treba ponavljati dok se razlučivost podprozora skupa sa svojim parametrima može skalirati, a da ne budu veći od razlučivosti izvorne slike.

U radu, autori su usporedili jedan od navedenih optimiziranih sustava implementiranih na FPGA sa algoritmom za detekciju lica kodne biblioteke otvorenog tipa za računalni vid (engl. *Open Source Computer Vision Library, OpenCV*), te su u rezultatima dobili od 20% do 25% bolju brzinu izvođenja koda uz približnu točnost detekcije [5].

3. PREDLOŽENO RJEŠENJE ALGORITMA ZA DETEKCIJU LICA NA SLICI POMOĆU FPGA

Nakon objašnjenog *Viola-Jones* algoritma i primjera implementacije navedenog algoritma, u ovome poglavlju slijede zahtjevi koje rješenje treba ispuniti, korišteni računalni alati i oprema, te tok razvoja predloženog rješenja uz uputstva za uporabu istoga.

3.1. Zahtjevi koje predloženo rješenje treba zadovoljiti

Zahtjevi koji su postavljeni za funkcionalnost predloženog rješenja u ovom radu su sljedeći:

1. Optimizirati algoritam za detekciju lica na slici prema zahtjevima FPGA
2. Optimizirani algoritam treba podržavati različite razlučivosti ulazne slike
3. Optimizaciju algoritma realizirati u *MATLAB*-u
4. Opisati digitalni dizajn pomoću jezika za opis sklopovlja iznimno brzih ugrađenih krugova (engl. *Very High Speed Integrated Circuit Hardware Description Language,* VHDL)
5. Digitalnu obradu slike izvršiti pomoću FPGA

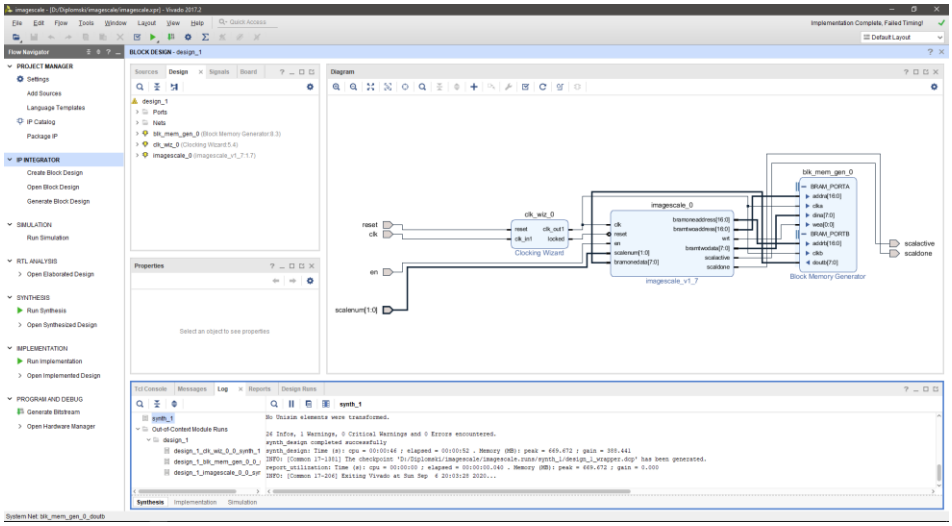
Kroz iduća potpoglavlja slijedi navod korištenih računalnih alata i opreme, te će se obraditi svaki od navedenih zahtjeva.

3.2. Korišteni alati i oprema

U ovome potpoglavlju će se navesti i pojasniti korišteni alati i oprema za funkcionalnu izvedbu algoritma za detekciju lica na slici pomoću FPGA.

3.2.1. Vivado

Vivado Design Suite [6] je računalni programski alat proizveden od strane *Xilinx*-a za sintezu i analizu HDL dizajna, koji zamjenjuje *Xilinx ISE* s dodatnim značajkama za razvoj sustava na čipu i sintezu na visokoj razini. *Vivado* je integralno okruženje za digitalni dizajn s alatima na razini sustava integralnih krugova, izgrađenim na zajedničkom skalabilnom modelu podataka i zajedničkom okruženju za uklanjanje pogrešaka. *Vivado* omogućava sintezu i implementaciju opisanog digitalnog dizajna pomoću jezika za opisivanje fizičke arhitekture [6].



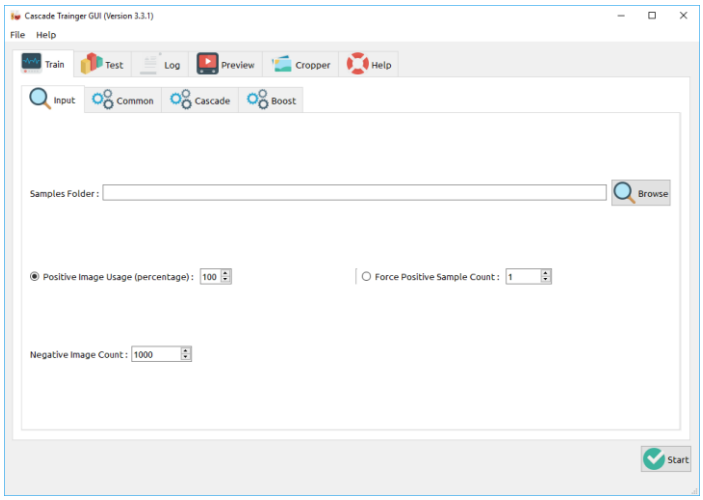
Sl. 3.1.Prikaz sučelja *Vivado*-a sa otvorenim blok dizajnom.

Na slici 3.1. je prikazano Vivado sučelje u kojem je otvoren prikaz dijagrama blok dizajna jednostavnog sustava za skaliranje slike pomoću srednje vrijednosti četiri susjedna piksela.

3.2.2. Cascade Trainer GUI

Cascade Trainer GUI je računalni programski alat koji se može koristiti za obuku, testiranje i poboljšanje modela kaskadnih klasifikatora. Koristi grafičko sučelje za postavljanje parametara i olakšavanje uporabe *OpenCV* alata za treniranje i testiranje klasifikatora [7].

Za primjenu ovoga alata potrebne su dvije baze podataka. Baza podataka isključivo sa pozitivnim uzorcima i baza podataka isključivo sa negativnim uzorcima.



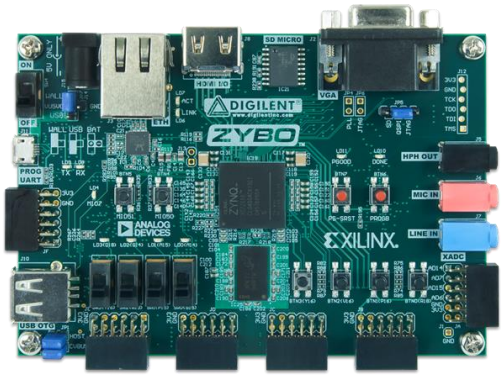
Sl. 3.2.Prikaz sučelja *Cascade Trainer GUI*-a [7].

Na slici 3.2. prikazano je sučelje *Cascade Trainer GUI*-a u početnom prozoru za treniranje kaskade.

3.2.3. ZYBO FPGA

ZYBO sustav na čipu (engl. *System on Chip*, SoC) FPGA [8] je *DIGILENT*-ova platforma razvijena oko *Zynq 7010* integralnih krugova. *Zynq 7010* temelji se na dvojezgrenom *ARM Cortex-A9* procesoru u kombinaciji sa *Xilinx 7-series* FPGA logikom. Navedeni FPGA posjeduje ugrađeni memorijski kontroler, video i audio ulazno izlazne članove, USB, Ethernet i SD podršku omogućujući dizajn sustava bez potrebe za dodatnim sklopovljem. *Zynq 7010* SoC FPGA posjeduje:

1. Dvojezgreni procesor *Cortex-A9* s 650 MHz
2. Memorijski kontroler dvostruke brzine prijenosa (engl. *Double Data Rate*, DDR) s 8 kanala za direktan pristup memoriji (engl. *Direct Memory Access*, DMA)
3. Periferni kontroleri široke propusnosti
4. Periferni kontroler niske širine pojasa
5. Re-programibilna logika ekvivalentna *Artixu-7* FPGA
    1. 4.400 logičkih kriški, svaka s četiri tablice za čitanje vrijednosti (engl. *Look-Up Tables*, LUT) sa 6 ulaza i 8 digitalnih sklopova sa dva stanja
    2. 240 kB brze memorije s nasumičnim pristupom raspodijeljene u blokove (engl. *Block Random Access Memory*, BRAM)
    3. Dva bloka za upravljanje signalom takta, svaka s fazno zaključanom petljom (engl. *Phase Locked Loop*, PLL) i upravljačem signala takta u mješovitom načinu rada
    4. 80 kriški za digitalnu obradu signala (engl. *Digital Signal Processing*, DSP) radne brzine veće od 450MHz
    5. Analogno-digitalni pretvarač [8]



Sl. 3.3.Prikaz *ZYBO* FPGA [8].

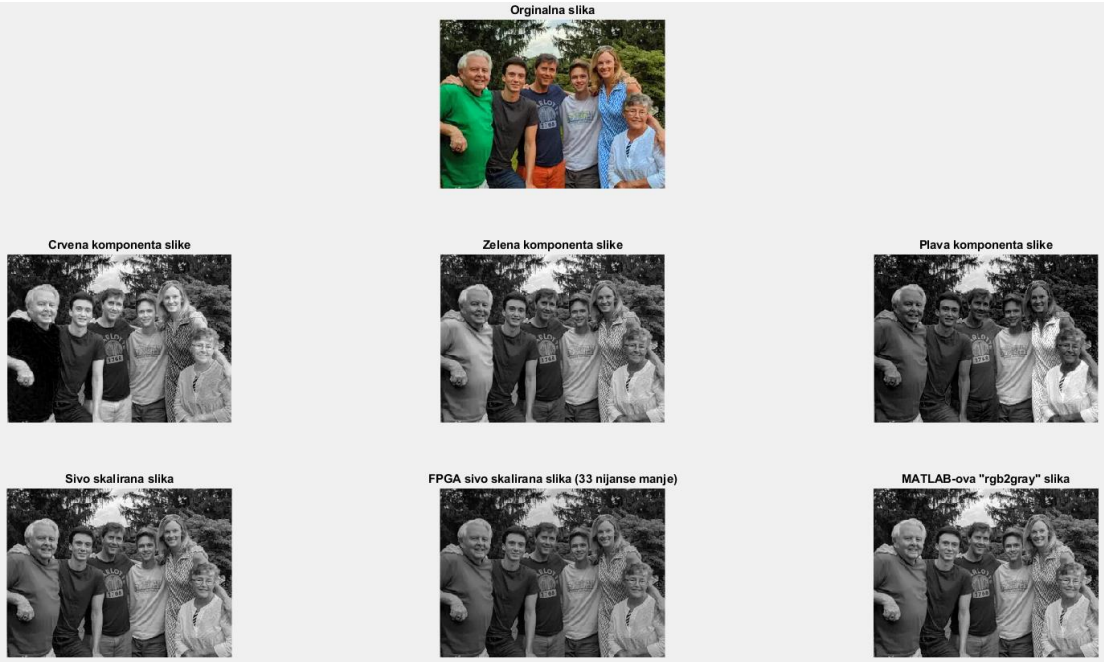
Slika 3.3. prikazuje *ZYBO* razvojnu pločicu sa *ZYNQ 7010* SoC FPG.

3.3. Opis predloženog algoritma

Prilikom implementacije *Viola-Jones* algoritma potrebno je uzeti u obzir ograničenost resursa *ZYNQ* SoC FPGA, te prema istom optimizirati algoritam za implementaciju na FPGA. Opis predloženog algoritma izveden pomoću *MATLAB*-a nalazi se u prilogu P.3.1.. S ciljem smanjenja količine bita po pikselu ulazne slike, optimiziranje ulazne slike se prema [5] izvršava konverzijom RGB slike u monokromatsku sliku. Prilikom reduciranja broja bitova, prema [5] korišten je izraz 3.4., koji iz crvene, zelene i plave komponente piksela, računa monokromatsku komponentu piksela:

$$monokromatsko(x,y) = 29.9\%crveno(x,y) + 58.7\%zeleno(x,y) + 11.4\%plavo(x,y)\,, \quad (3.4.)$$

gdje *monokromatsko*(*x*,*y*) predstavlja izračunatu monokromatsku vrijednost piksela na koordinati izvorne slike (*x*,*y*), a *crveno*, *zeleno* i *plavo* predstavlja svoju komponentu boje na istom pikselu.



Sl. 3.5. Prikaz rezultata algoritma za računanje sive slike iz RGB modela slike u boji, rađen u *MATLAB*-u.

Na slici 3.5. prikazani su rezultati algoritma za računanje monokromatske slike iz RGB modela slike u boji. Slika prikazuje postupak razdvajanja obojane slike po komponentama boje (srednji red), te rezultate zbrajanja iste tri komponente prema izrazu 3.4..

Prema *Viola-Jones* algoritmu [1] sljedeći korak je računanje integralne slike. Najveći zbroj svih monokromatskih piksela izvorne slike razlučivosti 320x240 iznosi 19.584.000, navedeni broj stane u 2^{25}, što znači da bi svaki piksel integralne izvorne slike zauzimao 25 bita (ili cijela

integralna slika $320 * 240 * 25 = 240$kB). Odabrani FPGA ima ukupnu BRAM memoriju od 240kB što je nedovoljno za cijelu integralnu izvornu sliku uz parametre digitalnog dizajna.

Zbog navedenoga, računa se integralna slika unutar podokvira izvorne slike, te se vrši detekcija lica na svakom podokviru zasebno. Izvršavanje algoritma detekcije lica unutar pojedinog podokvira omogućuje ispunjenje zahtjeva rada po kojem algoritam treba vršiti detekciju lica na raznim razlučivostima. Prema [4], zbog optimalne memorijske potrošnje, koristi se podokvir razlučivosti 39x59. Najveći zbroj svih piksela unutar zadanog podokvira iznosi 586.755, navedeni broj stane u 2^{20}, što znači da bi svaki piksel navedenog integralnog podokvira zauzimao 20 bita (ili cijeli integralni podokvir $39 * 59 * 20 = 3{,}5$kB).

Sl. 3.6. Prikaz rezultata algoritma za računanje integralne slike podokvira, rađen u *MATLAB*-u.

Slika 3.6. prikazuje 3 rezultata dobivena u *MATLAB*-u: izvornu sliku razlučivosti 320x240, ručno odabrani podokvir razlučivosti 39x59, te integralnu sliku navedenog podokvira. Dobivanje integralne slike se izvršava prema izrazu 2.1.. Zbog navedenog postupka zbrajanja na gornjim lijevim lokacijama integralne slike stoje tamniji pikseli koji su manje vrijednosti od donjih i desnih svjetlijih piksela.

Nakon dobivanja integralne slike, primjenjuje se algoritam za detekciju lica na slici. Implementirana jezgra algoritma za detekciju lica odgovara na pitanje „Nalazi li se lice na zadanoj lokaciji podprozora?“. Uloga jezgre algoritma za detekciju lica unutar podprozora je računanje *Haarovih* značajki, primjena parametara dobivenih treniranjem koristeći *AdaBoost* algoritam, te u konačnici kaskadnom klasifikacijom odrediti postojanje lica.

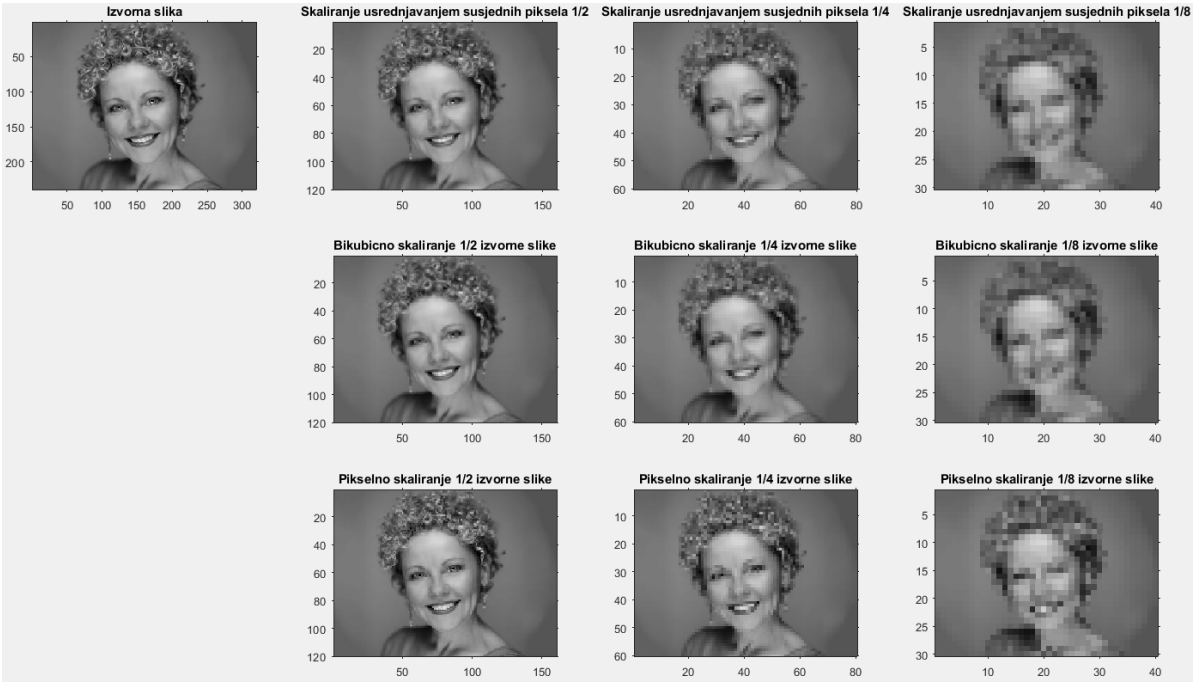
Za trening vlastite kaskade, osim algoritma za treniranje, potrebno je imati bazu slika isključivo sa željenim objektima u njima i bazu slika isključivo bez željenih objekata u njima. Što su veće baze podataka, to su rezultati precizniji, ali se time produljuje vrijeme izvršavanja

treninga. Navedeni algoritam nije nužan za detekciju lica, nego za detekciju bilo kojeg objekta za koji se parametri treniraju.

Sl. 3.7. Prikaz rezultata algoritma za detekciju lica napravljenog u *MATLAB*-u.

Slika 3.7. prikazuje detektirano lice u podokviru razlučivosti 39x59. Lice je detektirano algoritmom za detekciju lica izrađenom u *MATLAB*-u koristeći kaskadne parametre *OpenCV*-a. Navedeni kaskadni klasifikator se sastoji od 22 stupnja i 2135 koraka.

Kako je podprozor unutar kojeg se izvršava detekcija lica razlučivosti 24x24, prema [5] lica veća od navedene razlučivosti treba skalirati (umanjiti), što se postiže skaliranjem cijele izvorne slike. Na skaliranoj slici se izvršava detekcija lica, te je postupak potrebno ponavljati dok izvorna slika ne bude manja od podokvira integralne slike (razlučivosti 39x59).



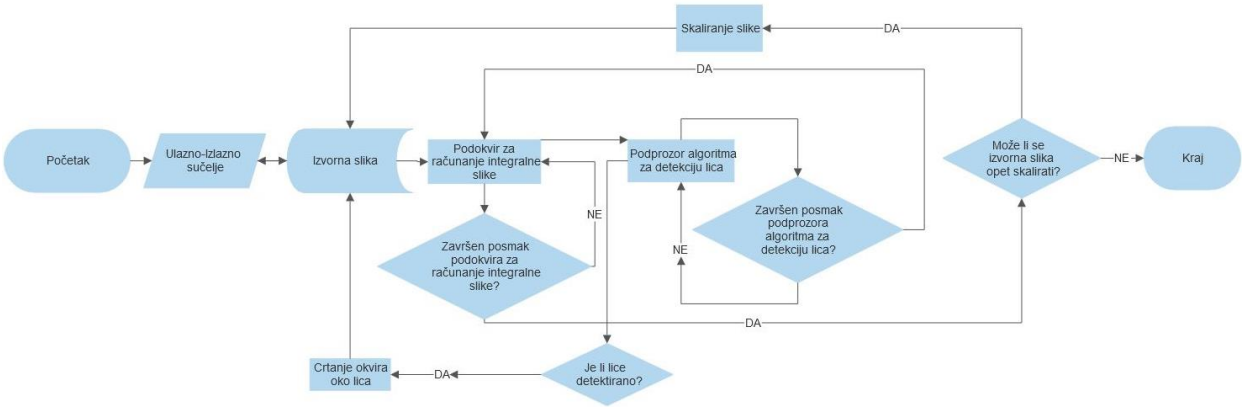
Sl. 3.8. Prikaz rezultata algoritma za skaliranje slike, rađen u *MATLAB*-u.

Na prethodnoj slici 3.8. je unutar *MATLAB*-a skalirana slika u manju koristeći tri funkcije. Prva funkcija kojom je slika skalirana je funkcija za skaliranje slike usrednjavanjem vrijednosti susjednih piksela u kojem se vrijednost svaka iduća četiri susjedna piksela zbraja i dijeli sa četiri. Druga funkcija za skaliranje je *MATLAB*-ova integralna funkcija „*imresize*“ koja koristi bi-kubičnu interpolaciju za skaliranje slike. Treća funkcija za skaliranje slike je skaliranje slike po pikselu prema [4]. Od tri navedena algoritma za skaliranje slike, odabire se najefikasniji za implementaciju u FPGA, a to je algoritam za skaliranje slike po pikselu (slika 2.9.).

Unutar ovog potpoglavlja opisan je način na koji su riješeni prvi, drugi i treći zahtjevi za rad sustava, a u nastavku rada slijedi prikaz implementacije optimiziranog algoritma u *MATLAB*-u, njegov digitalni opis u VHDL kodu za *ZYBO* FPGA, te ispunjenje preostalih zahtjeva rada.

3.4. Opis implementacije predloženog rješenja u FPGA

U ovome potpoglavlju slijedi opis FPGA implementacije predloženog rješenja.



Sl. 3.9. Prikaz dijagrama toka algoritma detekcije lica.

Na slici 3.9. prikazan je dijagram toka algoritma za detekciju lica. Na samom početku dijagrama toka nalazi se ulazno-izlazno sučelje koje je zaduženo za pohranu izvorne slike u memoriju FPGA te čitanje obrađene slike iz FPGA. Nakon što je slika pohranjena u FPGA memoriju, kreće njena obrada. Prvi korak obrade je dobivanje integralne slike unutar podokvira izvorne slike. Nakon dobivanja iste, na integralnoj slici se unutar podprozora izvršava algoritam za detekciju lica opisan u potpoglavlju 3.4.3. Ukoliko je lice detektirano, na izvornu sliku se ucrtava okvir oko detektiranog lica. Nad sučeljem za detekciju lica se vrši operacija posmaka unutar podokvira integralne slike. Ako je posmak podprozora za detekciju lica završen, obavi se jedan korak posmaka na podokviru za računanje integralne slike, te se opet obavlja detekcija unutar novog integralnog podokvira. Ukoliko su posmaci za računanje integralne slike podokvira

izvorne slike završeni i izvorna slika se može skalirati, navedena slika se skalira, te se postupak detekcije ponavlja. Završetak algoritma je kada se izvorna slika više ne može skalirati, a da ne postane manja od podokvira integralne slike.

3.4.1. Ulazno-izlazno sučelje

Ulazno-izlazno sučelje čini napredno proširivo sučelje (engl. *Advanced eXtensible Interface,* AXI4-Lite) [9] koje ostvaruje komunikaciju između procesora *Cortex-A9* i DDR memorije FPGA. Opis digitalnog dizajna ulazno-izlaznog sučelja izveden VHDL jezikom nalazi se u prilogu P.3.2.. AXI4 obitelj je sabirnica definiranih kao dio četvrte generacije standarda ARM-ove napredne sabirnice mikro-upravljačke arhitekture (engl. *ARM Advanced Microcontroler Bus Architectrue,* AMBA). AXI je prvi puta predstavljen s trećom generacijom AMBA-e.

AMBA specifikacija definira 3 AXI4 protokola:

1. AXI4 Memory-Maped (AXI4-full): adresabilan je, podržava niz podataka od maksimalno 256 jedinica podataka po transakciji
2. AXI4 Lite: Adresabilan, ne podržava niz podataka, može prenijeti samo jedan n bitovni podatak po transakciji
3. 3. AXI4 Stream: nije adresabilan , podržava neograničen tok podataka po transakciji [9].

U predloženom rješenju koristi se AXI4-Lite protokol kojeg čine četiri kanala: adresni kanal za pisanje, kanal za pisanje podataka, kanal za čitanje podataka, kanal za potvrdu. Navedeni kanali povezuju nadređeno sučelje (ZYNQ procesor) i podređeno sučelje (digitalni dizajn rada). Nadređeno sučelje određuje početak i kraj komunikacije sa podređenim sučeljem preko kanala za potvrdu bez obzira na smjer komunikacije. Podređeno sučelje šalje potvrdu nadređenom sučelju preko kanala za čitanje podataka u vidu predodređene vrijednosti nadređenog sučelja. Komunikacija se odvija u 10 koraka:

1. Podređeno sučelje čeka potvrdu nadređenog sučelja za početak
2. Nadređeno sučelje šalje potvrdu podređenom sučelju za početak
3. Nadređeno sučelje šalje podatke i adrese podređenom sučelju
4. Nadređeno sučelje šalje potvrdu podređenom sučelju za početak obrade
5. Nadređeno sučelje čeka potvrdu podređenog sučelja za završenu obradu
6. Podređeno sučelje šalje potvrdu nadređenom sučelju kada je završilo obradu podataka
7. Podređeno sučelje čeka potvrdu od nadređenog sučelja za slanje obrađenih podataka

8. Nadređeno sučelje šalje potvrdu za početak primanja obrađenih podataka
9. Nadređeno sučelje prima podatke od podređenog sučelja za poslane adrese
10. Nadređeno sučelje šalje potvrdu podređenom sučelju da su primljeni podatci i da se vrati u prvi korak

3.4.2. Postupak računanja integralne slike

Prema [1] računanje integralne slike je prvi korak ubrzanja algoritma za detekciju objekta. Integralna slika se dobije izrazom 2.1., a u radu se implementira pomoću pseudo koda 3.10.

```
1:      i, j kreću iz 1, a is kreće iz 0, gdje je is integralna slika, i
        broj aktivnog stupca podokvira slike, j broj aktivnog retka
        podokvira slike, a s podokvir izvorne slike;
2:      Za j:y, gdje je y visina podokvira slike:
3:         Za i:x, gdje je x širina podokvira slike:
4:            Ako je j=1:
5:               is(i,j)=is(i-1,j)+s(i,j);
6:               is(i-1,j)=is(i,j);
7:            A ako nije:
8:               is(i,j)=is(i-1,j)+s(i,j)+is(i,j-1);
9:               is(i-1,j)=is(i,j);
10:           Završi uvjet;
11:           Ako je i=x:
12:              i=1;
13:              j=j+1;
14:              is(i-1,j)=0;
15:           U suprotnom:
16:              i=i+1;
17:           Završi uvjet;
18:        Završi petlju;
19:     Završi petlju;
```

Sl. 3.10. Pseudo kôd algoritma za računanje integralne slike.

Na slici 3.10. je prikazan pseudo kod algoritma za računanje integralne slike iz izvorne slike, koji je korišten u radu i realiziran pomoću VHDL koda. Iz pseudo koda se može vidjeti kako se prema izrazu 2.1. vrijednosti u prvom redu izvorne slike zbrajaju sa svakom prethodnom vrijednosti integralne slike. Prilikom zbrajanja piksela u ostalim redovima, potrebno je tom načinu zbrajanja dodati još i vrijednost piksela integralne slike iz prethodnog reda i istog stupca. Algoritam se može bolje vizualizirati uz sliku 2.3.. Izračunata integralna slika se pohranjuje u BRAM memoriju. Opis digitalnog dizajna za postupak računanja integralne slike izveden VHDL kodom nalazi se u prilogu P.3.3..

3.4.3. Sučelje za detekciju lica na slici

Sučelje za detekciju lica na slici je najbitniji dio algoritma. Opis digitalnog dizajna navedenog sučelja izvedenog VHDL jezikom nalazi se u prilogu P.3.4.. Uloga istoga je uz

pomoć parametara dobivenih *AdaBoost* algoritmom pohranjenih u BRAM memoriju odrediti nalazi li se lice unutar zadanog podprozora. Da bi se to postiglo, implementirani proces detekcije lica se sastoji od 5 koraka:

1. Normalizacija kontrasta
2. Računanje *Haarovih* značajki
3. Odabir količine doprinosa *Haarovih* značajki na osnovi slabog praga
4. Akumulacija doprinosa
5. Čvrsta klasifikacija na osnovi čvrstog praga (engl. *strong threshold*)

Podprozor razlučivosti 24x24 piksela iz integralne slike može biti svjetliji ili tamniji, što nisu uvjeti po kojima se izvršava trening parametara za detekciju željenog objekta. Zbog navedenog se prema [1] predlaže normaliziranje kontrasta odabranog podprozora pomoću izraza 3.11.:

$$fn = \sqrt{(\sum_{n=1}^{N} i_n)^2 - \frac{\sum_{n=1}^{N}(i_n)^2}{N}}, \tag{3.11.}$$

Izraz 3.11. služi za izračun faktora normalizacije podprozora detekcije objekta gdje je *fn* faktor normalizacije, *N* broj piksela unutar podprozora, a i_n vrijednost intenziteta piksela unutar navedenog podprozora na izvornoj slici. Kako se radi o integralnoj slici zbroj svih piksela podprozora se računa pomoću izraza 2.2. što dovodi do izraza 3.12.

$$fn = \sqrt{\left(ii(x_1,y_1) - ii(x_2,y_2) - ii(x_3,y_3) + ii(x_4,y_4)\right)^2 - \frac{(ii(x_1,y_1)^2 - ii(x_2,y_2)^2 - ii(x_3,y_3)^2 + ii(x_4,y_4)^2)}{N}}. \tag{3.12.}$$

Iz izrazu 3.12. može se vidjeti funkcija za izračun faktora normalizacije kontrasta, te da se zbroj svih piksela unutar podprozora računa pomoću četiri vrijednosti piksela integralne slike. Izraz 3.12. zahtjeva korjenovanje, što je zahtjevna matematička funkcija za FPGA. S ciljem smanjenja prostornih resursa FPGA, za potrebe operacije vađenja drugog korijena korišteno je Vivado-vo intelektualno vlasništvo (engl. *Intellectual property*, IP) naziva digitalno računalo za rotiranje koordinata (engl. *COordinate Rotation DIgital Computer*, CORDIC). Faktor normalizacije se posebno množi sa svakom vrijednosti slabog praga, te se dobiju normalizirane vrijednost slabog praga.

Blok memorija predstavlja memoriju sa proizvoljnom duljinom vektora i veličinom memorije. Blok memorija se može postaviti u memoriju samo za čitanje (engl. *Read Only Memory*, ROM) i memoriju sa nasumičnim pristupom (engl. *Random-Access Memory*, RAM). Svaka od navedenih inicijaliziranih memorija može biti jedno-adresna ili dvo-adresna, što znači da im se može pristupati pomoću jedne ili dvije adrese jedne dimenzije. Blok memorija je

sinkrona memorija, te se upis i ispis podataka odvaja sinkrono sa signalom takta. Za pristup podatku ili memorijskom mjestu iz blok memorije, potrebno je željenoj blok memoriji predati adresu za navedeno memorijsko mjesto. Unutar digitalnog dizajna detektora lica korišteno je 20 blok memorija za pohranu 7 parametara dobivenih *AdaBoost* treniranjem. Slijedi opis pojedinih blok memorija:

1. Koordinate svake od tri komponente *Haarove* značajke (6 ROM memorija - $x_{0,1,2}$, $y_{0,1,2}$),
2. Veličina svake od komponenti *Haaroih* značajki (6 ROM memorija - $w_{0,1,2}$, $h_{0,1,2}$),
3. Količine doprinosa rezultatu *Haarovih* značajki za svaku od tri komponente značajki (3 ROM memorije - $wgh_{0,1,2}$),
4. Vrijednosti slabih pragova (1 ROM memorija),
5. Količina doprinosa ovisno o rezultatu slabih pragova (2 ROM memorije - l, r),
6. Vrijednosti čvrstih pragova (1 ROM memorija),
7. Količina slabih značajki unutar zadanog čvrstog praga (1 ROM memorija),

gdje je značajkama od 1 do 5 pristupano u blok memoriju sa rednim brojem aktivne slabe značajke, dok je značajkama 6 i 7 pristupano u blok memoriju sa rednim brojem aktivne čvrste značajke.

Nakon izračuna faktora normalizacije, uzevši raspodijeljenu blok memoriju u obzir, svaka izračunata slaba značajka može se preciznije usporediti sa svojstvenim slabim pragom. Korištene *Haarove* značajke se računaju iz integralne slike i sastoje se od dvije ili tri pravokutne komponente. Lokacije i veličine komponenti se preuzimaju iz blok memorija 1 i 2 u vidu koordinata, gdje (x,y) predstavlja prvu koordinatu pravokutne komponente, a $(x+w,y+h)$ predstavlja zadnju koordinatu pravokutne komponente. Svaki pravokutni oblik ima svojstvenu količinu doprinosa rezultatu preuzetu iz blok memorija 3. Prema [1], [4] i izrazu 2.2., slijedi izraz 3.13..

$$sz = wgh_0 D_0 + wgh_1 D_1 + wgh_2 D_2 \,, \tag{3.13.}$$

gdje je *sz* slaba značajka, *D* zbroj piksela iz izraza 2.2., a *wgh* predstavlja količinu doprinosa rezultatu pojedinog zbroja piksela *D*. Iako se komponente *Haarovih* značajki obično oduzimaju, zbog parametra *wgh* se navedene površine zbrajaju. Parametar *wgh* može biti pozitivan ili negativan, te svojim predznakom i brojem odlučuje hoće li se vrijednosti *D* zbrajati ili oduzimati i koliki će biti doprinos pojedine vrijednosti *D*. Izračunata slaba značajka uspoređuje se s normaliziranom vrijednosti slabog praga (vrijednost slabog praga preuzeta iz blok memorije 4

pomnožena sa faktorom normalizacije). Ako je slaba značajka veća, iz blok memorije 5 se preuzima vrijednost *r*, a ako nije, preuzima se vrijednost *l*. Svaka preuzeta vrijednost iz blok memorije 5 se zbraja u akumulirani doprinos.

Kada redni broj slabe značajke dosegne vrijednost preuzetu iz blok memorije 7, vrijednost akumuliranog doprinosa se uspoređuje sa vrijednosti čvrstog praga preuzetom iz blok memorije 6. Ako vrijednost akumuliranog doprinosa nije veća kaskada staje i lice nije detektirano, a ako navedena vrijednost je veća, algoritam za detekciju lica prelazi na sljedeći stupanj kaskade, te se postupak ponavlja prema pseudo kodu 3.14.. Prema [4], dobiven je pseudo kod algoritma za detekciju lica na slici pomoću FPGA:

```
1:  i, j kreću iz 1, gdje je i trenutni korak slabe značajke,
    a j trenutni korak čvrste značajke;
2:  akumulacija_doprinosa i zbroj_dosadašnjh_slabih_značajki kreću iz 0;
3:  normalizirana_vrijednost=normaliziraj vrijednost podprozora;
4:  Za i:x, gdje je x najveći redni broj slabe značajke:
5:     dohvati parametre iz BRAM-a za adrese i, j;
6:     slaba_značajka=izračunaj slabu značajku uzevši u obzir težine;
7:    slabi_prag=
      normalizirana_vrijednost*količina_slabog_praga(i);
8:        Ako je slaba_značajka > slabi_prag:
9:           akumulacija_doprinosa =
             akumulacija_doprinosa + istinita_količina_doprinosa(i);
10:       Ako nije:
11:          akumulacija_doprinosa =
             akumulacija_doprinosa + lažna_količina_doprinosa(i)
12:       Završi uvjet;
13:    Ako je i =
       broj_slabih_značajki_pod(j)+zbroj_dosadašnjh_slabih_značajki:
14:       zbroj_dosadašnjh_slabih_značajki =
          zbroj_dosadašnjh_slabih_značajki + 1;
15:       Ako je akumulacija_doprinosa > čvrsti_prag(j):
16:         Ako i >= x:
17:             OBJEKT JE DETEKTIRAN;
18:             Završi;
19:          Ako nije:
20:             j = j + 1;
21:             akumulacija_doprinosa = 0;
22:          Završi uvjet;
23:       Ako nije:
24:          OBJEKT NIJE DETEKTIRAN;
25:          završi;
26:       Završi uvjet;
27:    i = i + 1;
28: Završi petlju;
```

Sl. 3.14. Pseudo kôd algoritma za detekciju lica unutar podprozora [4].

Na slici 3.14. prikazan je pseudo kod za detekciju lica unutar podprozora. Zadatak algoritma za detekciju lica unutar podprozora je detektirati objekt na osnovu pohranjenih parametara, te *i* i *j* koordinata.

3.4.4. Crtanje okvira oko detektiranog objekta

Crtanje okvira oko detektiranog objekta ima funkciju označavanja pronađenog objekta na izvornoj slici za kasniju vizualizaciju rješenja algoritma. Opis digitalnog dizajna za postupak crtanja okvira oko detektiranog objekta izveden VHDL jezikom nalazi se u prilogu P.3.5.. Navedeni algoritam treba imati informacije o razlučivosti izvorne slike, razlučivosti podokvira integralne slike, razlučivosti podprozora, početnu adresu (koordinatu) podokvira integralne slike, početnu adresu (koordinatu) podprozora detektora lica i nivo skaliranja slike za detektirani objekt. Pomoću navedenih parametara, algoritam digitalnog dizajna za crtanje okvira oko detektiranog lica može izračunati adresu za početak crtanja okvira oko objekta, te crtati okvir rezolucije podprozora pomnožen sa 2^n, gdje je n nivo skaliranja slike.

$$s_d = s_{ii} + integer\left(\frac{s_p}{M_{ii}}\right) * M_i * 2^n + (s_p) mod (M_{ii}) * 2^n \,, \tag{3.15.}$$

U izrazu 3.15. s_d predstavlja početnu adresu jedne dimenzije za crtanje okvira oko detektiranog objekta, s_{ii} predstavlja početnu adresu jedne dimenzije integralne slike, s_p predstavlja početnu adresu jedne dimenzije podprozora, M_i predstavlja komponentu razlučivosti M_ixN_i kod izvorne slike, M_{ii} predstavlja komponentu razlučivosti M_{ii}xN_{ii} kod integralne slike, a n označava nivo aktivnog skaliranja. Pomoću navedenog izraza se dobije početna adresa za crtanje okvira oko detektiranog objekta. Proporcionalno sa n uz indeks 2^n raste svaka rezolucijska komponenta navedenog okvira. Za razlučivost podprozora M_pxN_p crta se okvir oko detektiranog objekta veličine $(2^n * M_p)$x$(2^n * N_p)$ lociran desno i ispod početne adrese za crtanje okvira.

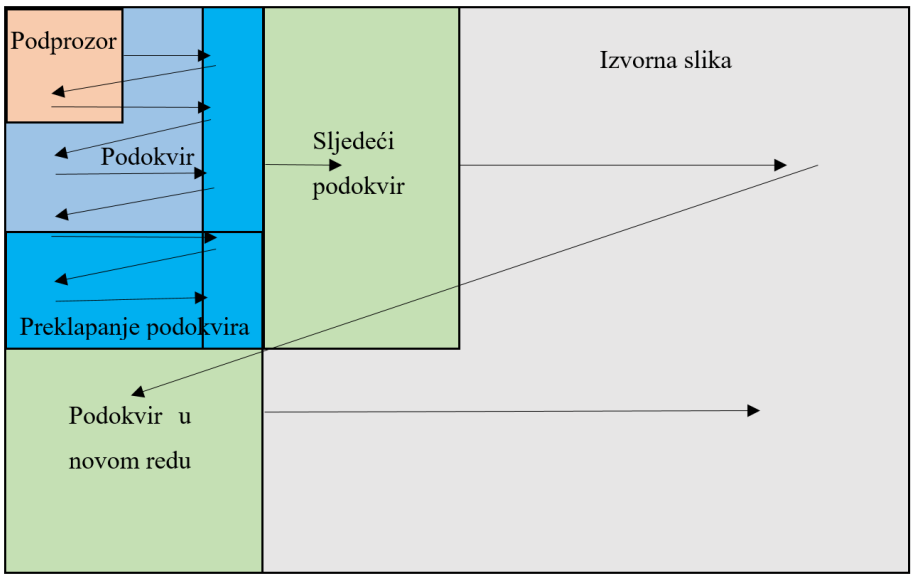
3.4.5. Skaliranje slike

Zbog efikasnog izvođenja digitalnog dizajna na FPGA i ubrzanja algoritma za detekciju lica na slici, koristi se algoritam za skaliranje slike po pikselu koji je vizualiziran na slici 2.9.. Prednost navedenog algoritma prema ostalim algoritmima za skaliranje slike je da sučelje za računanje integralne slike može direktno čitati skalirane podatke iz BRAM-a jednostavnom manipulacijom adresa. Primjena skaliranja izvorne slike jednostavnom manipulacijom adresa izvedena VHDL kodom nalazi se u prilozima P.3.3., P.3.5. i P.3.6.. Zbog navedenog pristupa dodatno vrijeme za obavljanje matematičkih operacija (koje bi skaliranje slike usrednjavanjem vrijednosti piksela i bi-kubično skaliranje trebalo) nije potrebno. Ako obavljanje matematičkih operacija navedenih algoritama za skaliranje slike ne postoji, dodatno sklopovlje unutar FPGA nije potrebno.

BRAM-i iz kojih se vrijednost piksela iščitava zahtijevaju barem dva signala takta za prijenos informacije. Kako algoritam za skaliranje slike usrednjavanjem susjednih piksela treba četiri vrijednosti iz BRAM memorije, navedenom algoritmu je potrebno osam signala takta za izračunavanje jednog piksela. Osim navedenih, nedostatak istog algoritma je da se skalirana slika negdje treba pohraniti, a to zahtjeva dodatnu BRAM memoriju za razliku od jednostavnog skaliranja po pikselu koje iz BRAM memorije sa pohranjenom izvornom slikom direktno računa skaliranu integralnu sliku.

3.4.6. Upravljačka jedinica implementiranog dizajna

Upravljačka jedinica implementiranog dizajna je dio digitalnog dizajna algoritma detekcije lica na FPGA koji je zadužen za upravljanje podokvirom za računanje integralne slike i podprozorom za detekciju lica, te za povećanje rednog broja skaliranja koji direktno utječe na manipulaciju adresama za skaliranje slike. Opis digitalnog dizajna upravljačke jedinice izvedenog VHDL jezikom nalazi se u prilogu P.3.6.. Upravljanje podokvirom za računanje integralne slike i podprozorom za detekciju lica se izvršava funkcijom posmaka.



Sl. 3.16. Vizualni prikaz algoritma upravljačke jedinice.

Prema slici 3.16. podprozor unutar integralne slike (podokvira) vrši operaciju posmaka za jednu po jednu adresu dok podprozor ne dotakne desni rub podokvira. Nad podprozorom za detekciju lica moguće je vršiti funkciju posmaka za više adresnih mjesta po koraku, ali se navedenim postupkom gubi na preciznosti algoritma dok se povećava brzina sustava. Ako je desni rub podokvira dotaknut, podprozor se postavlja u novi red, a procedura se ponavlja dok podprozor ne dotakne donji i desni rub podokvira istovremeno.

Nakon što je završen postupak algoritma detekcije lica unutar podprozora na zadnjoj iteraciji posmaka, podprozor se postavlja u početnu adresu, te se izvršava operacija posmaka na podokviru integralne slike. Podokvir integralne slike se operacijom posmaka pomjera za *I-P+1* adresnih mjesta, gdje je *I* širina integralne slike, a *P* širina podprozora, a potom se pokreće operacija posmaka podprozora unutar novog podokvira. Ukoliko bi podokvir u idućem adresnom posmaku prešao preko desnog ruba izvorne slike, potrebno ga je postaviti da dotiče rubnu liniju izvorne slike.

U slučaju kada podokvir dotakne desni rub izvorne slike, izvršava se posmak koji podokvir postavlja u inicijalnu adresu novog reda izvorne slike. Novi red izvorne slike se računa izrazom *J-R+1*, gdje je *J* visina integralne slike, a *R* visina podprozora. Ukoliko bi podokvir prešao preko donjeg ruba izvorne slike, potrebno ga je postaviti da dotiče rubnu liniju slike. Kada podprozor i podokvir dotiču donji desni rub izvorne slike, nivo skaliranja slike se povećava za jedan, a adrese se postavljaju na početak. Zbog skalirane slike operacija posmaka na podokviru se izvršava sa 2^n puta većim adresnim korakom, gdje je *n* nivo skalirane slike.

```
1:  PP predstavlja početnu adresu podprozora, PO početnu adresu podokvira;
2:  n predstavlja nivo skaliranja slike koji kreće iz 0,
    gdje je T najveći nivo skaliranja slike;
3:  i, j kreću iz 1, gdje su i, j koordinate početne točke podokvira
    na izvornoj slici razlučivosti IxJ;
4:  x, y kreću iz 1, gdje su x, y koordinate početne točke
    podprozora razlučivosti MxN unutar podokvira razlučivosti XxY;
5:  Za n:T
6:     Za j:(J-Y+1),
7:        Za i:(I-X+1),
8:           Integriraj podokvir na PO(i,j);
9:           Za y:(Y-N+1),
10:             Za x:(X-M+1),
11:                Detektiraj objekt na PP(x,y);
12:                x=x+1;
13:             Završi petlju;
14:             y=y+1;
15:          Završi petlju;
16:          Ako je (i+(X-M+1)*2^n) > (I-(X+1)*2^n):
17:             i=I-(X+1)*2^n;
18:          Ako nije:
19:             i=i+(X-M+1)*2^n;
20:          Završi uvjet;
21:       Završi petlju;
22:       Ako je (j+(Y-N)*2^n+1) > (J-Y*2^n+1):
23:          j=J-Y*2^n+1;
24:       Ako nije:
25:          j=j+(Y-N)*2^n+1;
26:       Završi uvjet;
27:    Završi petlju;
28:    n=n+1;
29: Završi petlju;
```

Sl. 3.17. Pseudo kôd algoritma upravljačke jedinice.

Iz pseudo koda 3.17. može se vidjeti prethodno objašnjeni algoritam upravljačke jedinice koji upravlja s početnim adresama i rednim brojem skaliranja slike. Kako se prilikom skaliranja slike koristi manipulacija adresama, na pseudo kodu algoritma upravljačke jedinice se na više mjesta može vidjeti množenje sa 2^n gdje *n* predstavlja nivo skaliranja slike kojem je zadatak imaginarno proširiti podokvir preko izvorne slike uzimajući svaku 2^n-tu vrijednost piksela.

3.5. Konačni dizajn algoritma za detekciju lica na slici pomoću FPGA

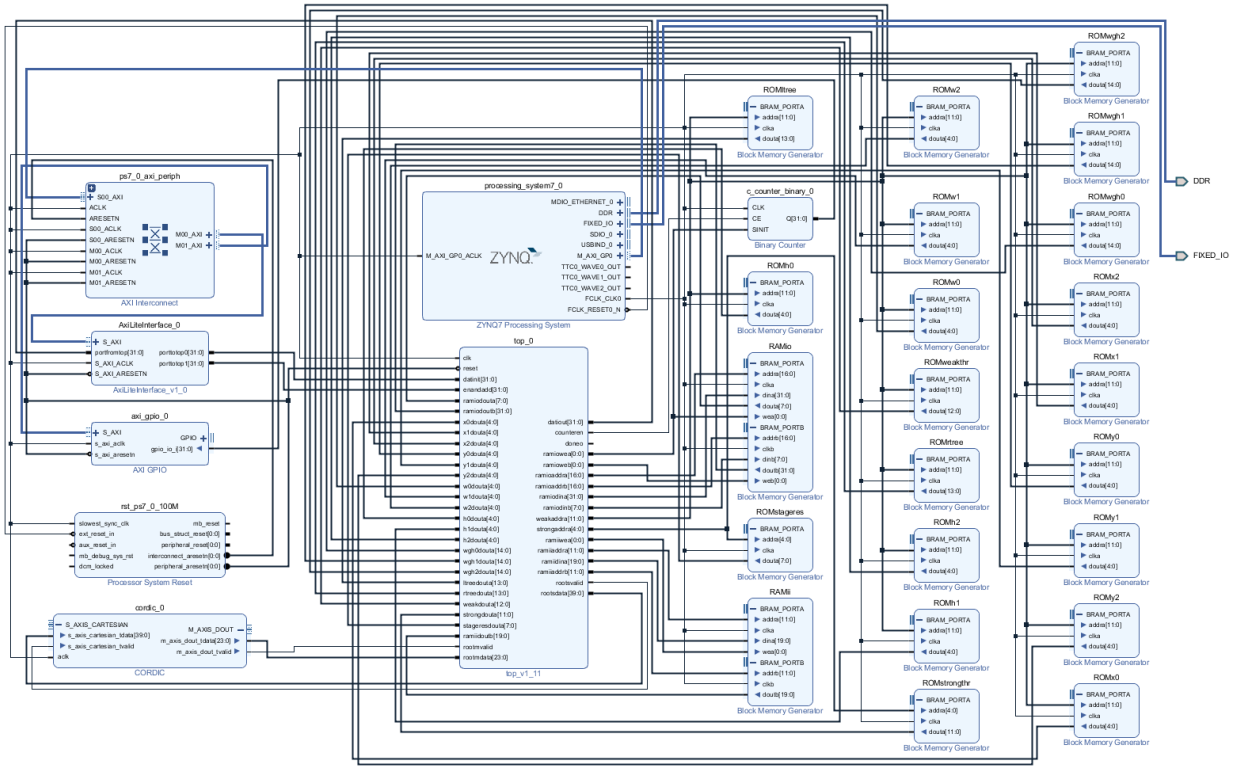
Na osnovu prije opisanih jedinica, implementiran je konačan dizajn algoritma za detekciju lica pomoću FPGA, koji je opisna u nastavku ovoga potpoglavlja. Cjeloviti *Vivado* projekt izveden u istoimenom računalnom programu nalazi se u prilogu P.3.7.. Digitalni dizajn implementiranog algoritma detekcije lica na slici pomoću FPGA dizajniran je kao općeniti algoritam za detekciju objekta.

Da bi se implementirao sustav po želji, potrebno je u BRAM memoriju pohraniti željene parametre za detekciju objekta, te izmijeniti navedene generičke varijable unutar „*gTop.vhd*“ datoteke:

- *imageaddresswidth* – Određuje količinu bita potrebnu za adrese izvorne slike
- *iimageaddresswidth* – Određuje količinu bita potrebnu za adrese integralne slike
- *bramonex* – Određuje širinu izvorne slike
- *bramoney* – Određuje visinu izvorne slike
- *integralx* – Određuje širinu integralnog podokvira
- *integraly* – Određuje visinu integralnog podokvira
- *subx* – Određuje širinu podprozora
- *suby* – Određuje visinu podprozora
- *largersubbitnum* – Određuje količinu bita potrebnu za veći od parametara visine ili širine podprozora
- *weak_feature_limit* – Određuje maksimalan broj slabih pragova
- *iidatabitleng* – Određuje količinu bita potrebnu za spremanje informacije piksela u integralnu sliku
- *regnum* – Opisuje širinu ulaznih i izlaznih registara sustava
- *maxscalenum* – Određuje maksimalni nivo skaliranja slike.

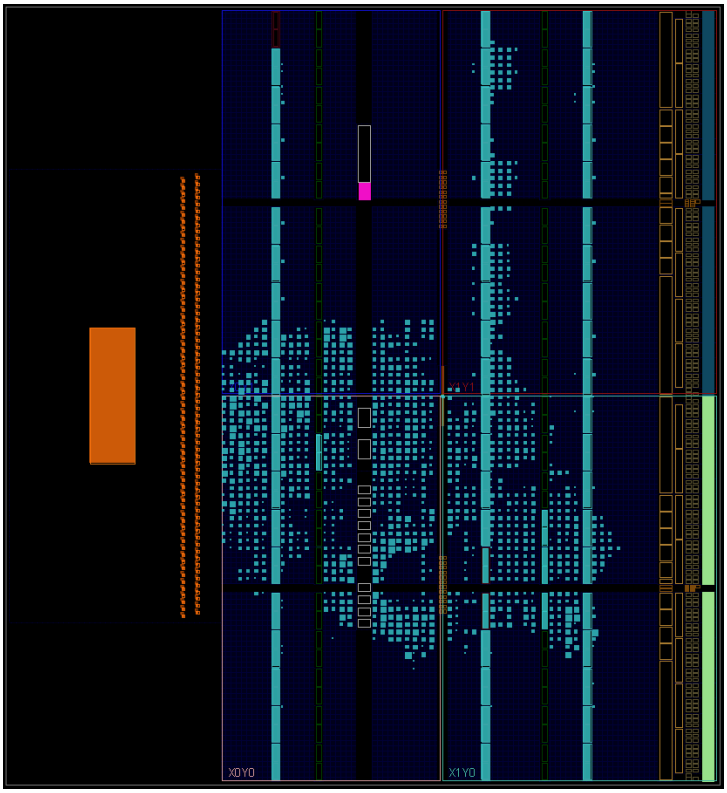
Za predloženo rješenje postavljeni su parametri sustava za detekciju lica koji može obraditi izvornu sliku razlučivosti 320x240 piksela (koju je moguće skalirati samo 2 puta a da ne postane

manja od razlučivosti podokvira), podokvirom razlučivosti 39x59 piksela (za čije podatke treba barem 20 bita po pikselu) i podprozorom razlučivosti 24x24 piksela (za koje treba barem 5 bita). Sustav koristi 2913 slabih pragova raspoređenih u 25 stupnjeva po prijedlogu literature [4], a ulazni i izlazni registri koriste po 32 bita.



Sl. 3.18. Vizualni prikaz blok dizajna algoritma za detekciju lica na slici pomoću FPGA.

Na slici 3.18. prikazan je blok dizajn algoritma za detekciju lica pomoću FPGA sa potrebnim jedinicama za implementaciju dizajna na *ZYNQ* SoC FPGA. Blok dizajn sa slike prikazuje blok „*top_0*“ unutar kojega se nalazi cijeli algoritam za detekciju lica pomoću FPGA opisan VHDL kodom, što je zahtjev rada. Uz njega su još prisutne potrebne BRAM memorije za pohranu parametara za detekciju lica, te za pohranu izvorne slike i integralne slike. Osim BRAM-a, mogu se vidjeti korišteni IP-evi: „cordic_0“ koji je zadužen za korjenovanje, „*c_counter_binary*“ koji je zadužen za brojanje potrebnog vremena izvršavanja implementiranog algoritma detekcije lica pomoću FPGA, AXI sučelja za komunikaciju između procesora i FPGA, te ZYNQ procesor.



Sl. 3.19. Vizualni prikaz implementiranog dizajna algoritma za detekciju lica na slici pomoću FPGA.

Na implementiranom dizajnu algoritma za detekciju lica na slici pomoću FPGA, prema slici 3.19. može se vidjeti raspored digitalnog dizajna unutar *ZYNQ* SoC FPGA. Svjetlo plavom bojom i svjetlo narančastom bojom su označeni upotrebljavani dijelovi *ZYNQ* SoC FPGA.

Kako je vidljivo na slici 3.19., *ZYNQ* SoC FPGA je poprilično upotrjebljen, a posebno su upotrijebljeni BRAM-i. Iz dobivenog izvještaja o implementiranom dizajnu na 55.56 MHz, može se vidjeti sveukupno zauzeće *ZYNQ* SoC FPGA navedenim dizajnom (prikazano tablicom 3.20.), te potrošnja energije pojedinog dijela i cjelovitog navedenog digitalnog dizajna algoritma za detekciju lica (prikazano tablicom 3.21.).

Tablica 3.20. Prikaz upotrijebljenosti *ZYNQ* SoC FPGA predloženim rješenjem.

Resurs	Dostupno	Upotrjebljeno	Upotrjebljeno [%]
LUT	17600	3462	19.67
LUTRAM	6000	96	1.60
FF	35200	2271	6.45
BRAM	60	59	98.33
DSP	80	8	10.00
BUFG	32	1	3.13

Tablica 3.20. prikazuje dostupne i upotrijebljene količine pojedinog resursa FPGA za digitalni dizajn predloženog rješenja, gdje su digitalni sklopovi sa dva stanja (engl. *Flip-Flop,* FF), slučajno pristupne memorije tablica za čitanje vrijednosti (engl. *Look-Up Tables Random Access Memory,* LUTRAM), a globalni međuspremnici (engl. *BUFfers Global,* BUFG). Moguće je poboljšanje iskoristivosti resursa paralelnim izvođenjem dijela dizajna, ili smanjiti količinu potrebnih resursa smanjenjem razlučivosti izvorne i integralne slike, te optimizacijom digitalnog dizajna.

Tablica 3.21. Prikaz rasporeda snage i termičkog zagrijavanja *ZYNQ* SoC FPGA predloženog rješenja.

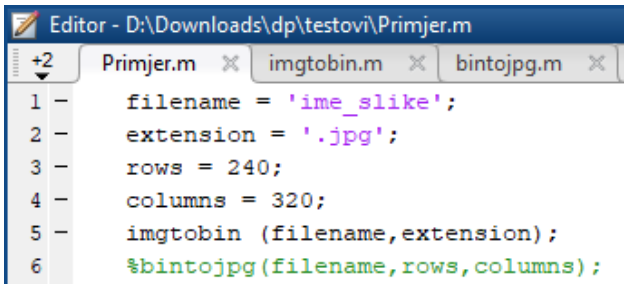
Raspored snage	Snaga [W]	
Signali takta	0.006	
Prijenosni signali	0.011	
Logički sklopovi	0.008	
BRAM-i	0.010	
DSP-ovi	0.003	
Procesor	1.555	
Cjeloviti dizajn	1.594	
Nezavisna potrošnja	0.133	
Ukupna potrošnja	1.728	Termičko zagrijavanje = 44.9 ºC

Tablica 3.21. prikazuje raspored snage unutar FPGA, pri implementaciji digitalnog dizajna algoritma za detekciju lica na slici pomoću FPGA. Na tablici je prikazan i iznos termičkog zagrijavanja FPGA za predloženo rješenje. Moguće su optimizacije potrošnje digitalnog dizajna: smanjenjem broja množenja i dijeljenja unutar digitalnog dizajna, smanjenjem frekvencije signala takta (koja iznosi 55.56MHz prilikom navedene implementacije) i uključivanjem digitalnih sklopova samo onda kada ih sustav treba koristiti.

3.6. Pokretanje algoritma za detekciju lica na slici pomoću FPGA

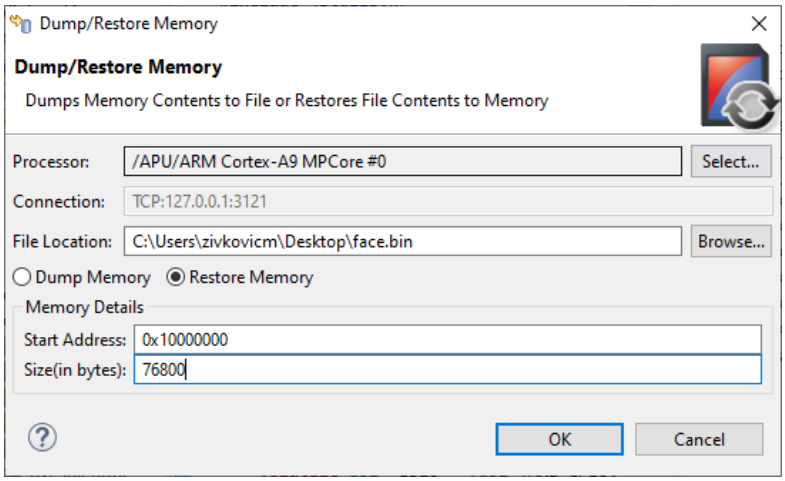
Unutar ovoga potpoglavlja opisan je način korištenja algoritma za detekciju lica na slici pomoću FPGA. Potrebno je koristiti računalne programske alate *MATLAB* i *Vivado razvojno okruženje (engl, Software Development Kit,* SDK*)*. Osim računalnih alata potrebno je imati i sliku formata zajedničke skupine fotografa (engl. *Joint Photographic Group,* JPG) ili sliku formata prijenosne mrežne grafike (engl. *Portable Network Graphics,* PNG) željene razlučivosti na kojoj se želi provesti detekcija lica.

Prvi korak je pokrenuti *MATLAB*, te otvoriti kod iz priloga P.3.8. pod nazivom „*DiplPrimjer.m*“ u kojem se nalaze dvije funkcije. Funkcija za transformaciju slike u format toka okteta za računalnu programsku podršku (engl. *Application/Octet-Stream*, BIN), te funkcija za transformaciju BIN formata u JPG format. Nakon podešavanja koda prema slici 3.22. potrebno je unijeti ime slike na kojoj se želi detektirati lice, produžetak za format te slike, te širinu i visinu izvorne slike. Nakon pritiska tipke „*f5*“ generira se datoteka istog naziva kao i izvorna slika, ali produžetka formata „*.bin*“, te se time završava prvi korak uporabe.



Sl. 3.22. Prikaz prvog koraka za uporabu.

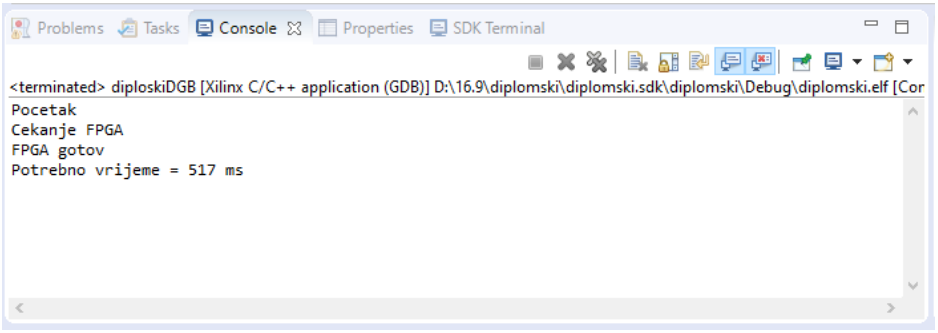
U drugom koraku potrebno je pokrenuti *Vivado SDK*, kliknuti na „*Xilinx Tools*“ u gornjem izborniku, te pokrenuti „*Dump/Restore Data File*“. Pokrenuti izbornik treba postaviti prema slici 3.23..



Sl. 3.23. Prikaz drugog koraka za uporabu.

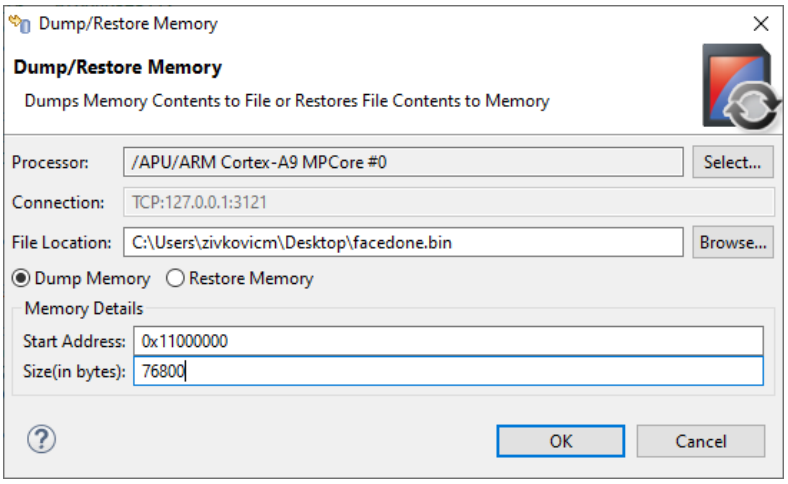
Nakon postavljanja izbornika prema slici 3.23., potrebno je locirati vlastito izvezenu BIN datoteku, te postaviti broj bajtova potrebnih za cijelu izvornu sliku. Potvrđivanjem postavki kojima se odabrana BIN datoteka pohranjuje u DDR memoriju odabranog procesora *ZYNQ* SoC FPGA na adresi $0x10000000_{(16)}$ i pokretanjem algoritma za detekciju lica na slici, u *Vivado SDK*

dobije se ispis sličan ispisu sa slike 3.24.. Na prikazanom ispisu su vidljiva stanja obrade slike, te vrijeme trajanja implementiranog algoritma detekcije lica na slici pomoću FPGA.



Sl. 3.24. Primjer prikaza ispisa u Vivado SDK nakon pokretanja detekcije lica na slici pomoću FPGA.

Treći korak za uporabu je izvesti detektiranu sliku iz DDR memorije FPGA na adresi $0x11000000_{(16)}$ u memoriju računala, što se postiže ponovnim pokretanjem „*Dump/Restore Data File*“, te postavljanjem prema slici 3.25. uz vlastiti odabir imena za izvezenu datoteku, te broj bajtova potreban za cijelu sliku.



Sl. 3.25. Prikaz trećeg koraka za uporabu.

U zadnjem koraku treba novu dobivenu datoteku BIN formata iz DDR memorije FPGA transformirati u JPG format koristeći *MATLAB* kod iz priloga „*DiplPrimjer.m*“. Kod je potrebno postaviti prema slici 3.26., te unijeti ime slike nove datoteke izvezene iz DDR memorije BIN formata. Nakon pritiska tipke „*f5*“ generira se datoteka istog naziva kao i izvezena datoteka BIN formata, ali produžetka „*.jpg*“.

```
Editor - D:\Downloads\dp\testovi\Primjer.m
+2  Primjer.m | imgtobin.m | bintojpg.m
1 -   filename = 'ime_slike';
2 -   extension = '.jpg';
3 -   rows = 240;
4 -   columns = 320;
5     %imgtobin (filename,extension);
6 -   bintojpg(filename,rows,columns);
```

Sl. 3.26. Prikaz zadnjeg koraka za uporabu.

Nakon zadnjeg koraka moguće je vidjeti rezultate detektirane slike te ju usporediti sa slikom prije detekcije, što se može vidjeti i na slici 3.27..

Sl. 3.27. Prikaz primjera uspješne detekcije lica na slici pomoću FPGA.

Na slici 3.27. prikazan je primjer usporedbe slike prije algoritma za detekciju lica i nakon algoritma za detekciju lica na slici pomoću FPGA. Vidljive promjene na slici nakon algoritma za detekciju lica su crni okviri oko uspješno detektiranog lica, što znači da je lice detektirano više puta za redom na drugim, ali bliskim adresama za detekciju lica. U nastavku rada proći će se kroz analizu testiranja algoritma za detekciju lica pomoću FPGA, gdje će se diskutirati o rezultatima testiranja, te će se predstavljeni sustav usporediti sa sličnim sustavom.

4. TESTIRANJE RJEŠENJA ALGORITMA ZA DETEKCIJU LICA NA SLICI POMOĆU FPGA

Unutar ovog poglavlja predstaviti će se način testiranja i rezultati testiranja detekcije na slici pomoću FPGA, te će se analizirati dobiveni rezultati uz usporedbu sa sličnim rješenjem.

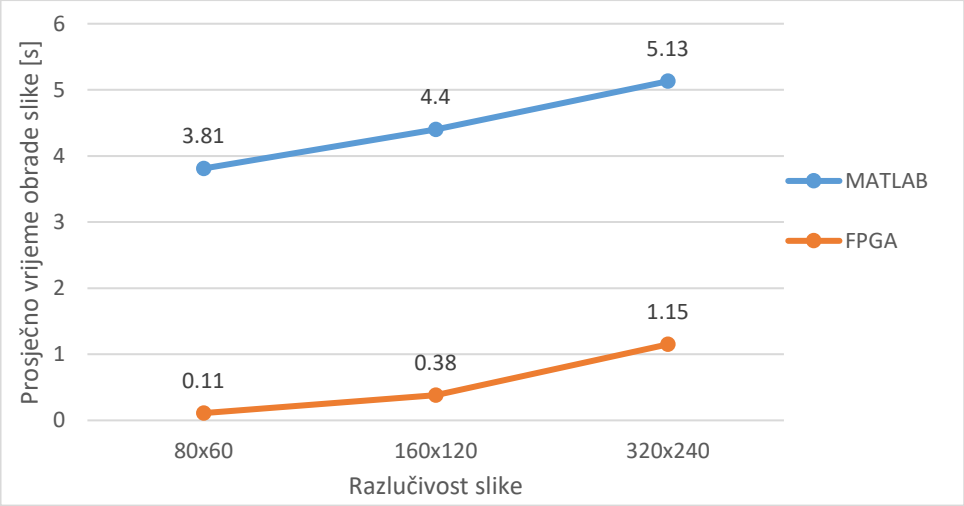
4.1. Testiranje rješenja algoritma za detekciju lica na slici pomoću FPGA

Testiranje rješenja za detekciju lica na slici pomoću FPGA provedeno je na *ZYNQ* SoC FPGA pomoću *MATLAB*-a i *Vivado SDK* po uputama za uporabu iz podpoglavlja 3.6. Digitalni dizajn algoritma za detekciju lica implementiran je sa signalom takta od 55.56MHz. Kroz sustav je testirano ukupno 90 slika iz priloga P.4.1.. Slike su razlučivosti 320x240, 160x120 i 80x60 od kojih je 50 slika sa licima (pozitivni uzorci) i 40 slika bez lica (negativni uzorci). Kod slika bez lica se promatraju dobre detekcije (engl. *true negtives*) i loše detekcije (engl. *false positives*), te se donosi binarna odluka o detekciji. Kod slika sa licima se promatra broj lica koji je dobro detektiran (engl. *true positives*) i loše detektiran (engl. *false negatives*). U tablici 4.1. su prikazani rezultati detekcije za navedenih 90 slika detektiranih pomoću FPGA uz parametre za detekciju lica na slici prema [4], te detektiranih sa *MATLAB*-om uz *OpenCV* parametre za detekciju lica na slici.

Tablica 4.1. Usporedni prikaz rezultata detekcije lica na testnim slikama između FPGA i MATLAB detekcije.

/	N	Broj lica na slikama	FPGA					MATLAB				
			Dobre detekcije	Loše detekcije	Prosječno vrijeme [s]			Dobre detekcije	Loše detekcije	Prosječno vrijeme [s]		
					320x240	160x120	80x60			320x240	160x120	80x60
Negativni uzorci	40	0	33	7	0,96	0,31	0,09	37	3	4,12	3,86	3,35
	[%]		83	17				92	8			
Pozitivni uzorci	50	166	125	101	1,31	0,44	0,12	148	73	5,93	4,83	4,17
	[%]		75	61				89	44			
Ukupno [%]			79	41	1,15	0,38	0,11	90	28	5,13	4,40	3,81

Iz tablice 4.1. se vidi da FPGA detektira 75% lica, a *MATLAB* uz parametre *OpenCV*-a 89% lica, što je 84% detekcija lica pomoću FPGA u usporedbi sa *MATLAB*-om. Iz iste se vidi da FPGA uspješno detektira 79% detekcija, a *MATLAB* 90%, što je 88% uspjeha detekcije lica pomoću FPGA u usporedbi sa *MATLAB*-om. Navedenom tablicom je prikazana i vremenska prednost obrade slike pomoću FPGA. FPGA izvršava obradu slike 4,46 puta brže od *MATLAB*-a za najveću testiranu sliku, 11,58 puta brže od *MATLAB*-a za srednju testiranu sliku, te 34,64 puta brže od *MATLAB*-a za najmanju testiranu sliku.



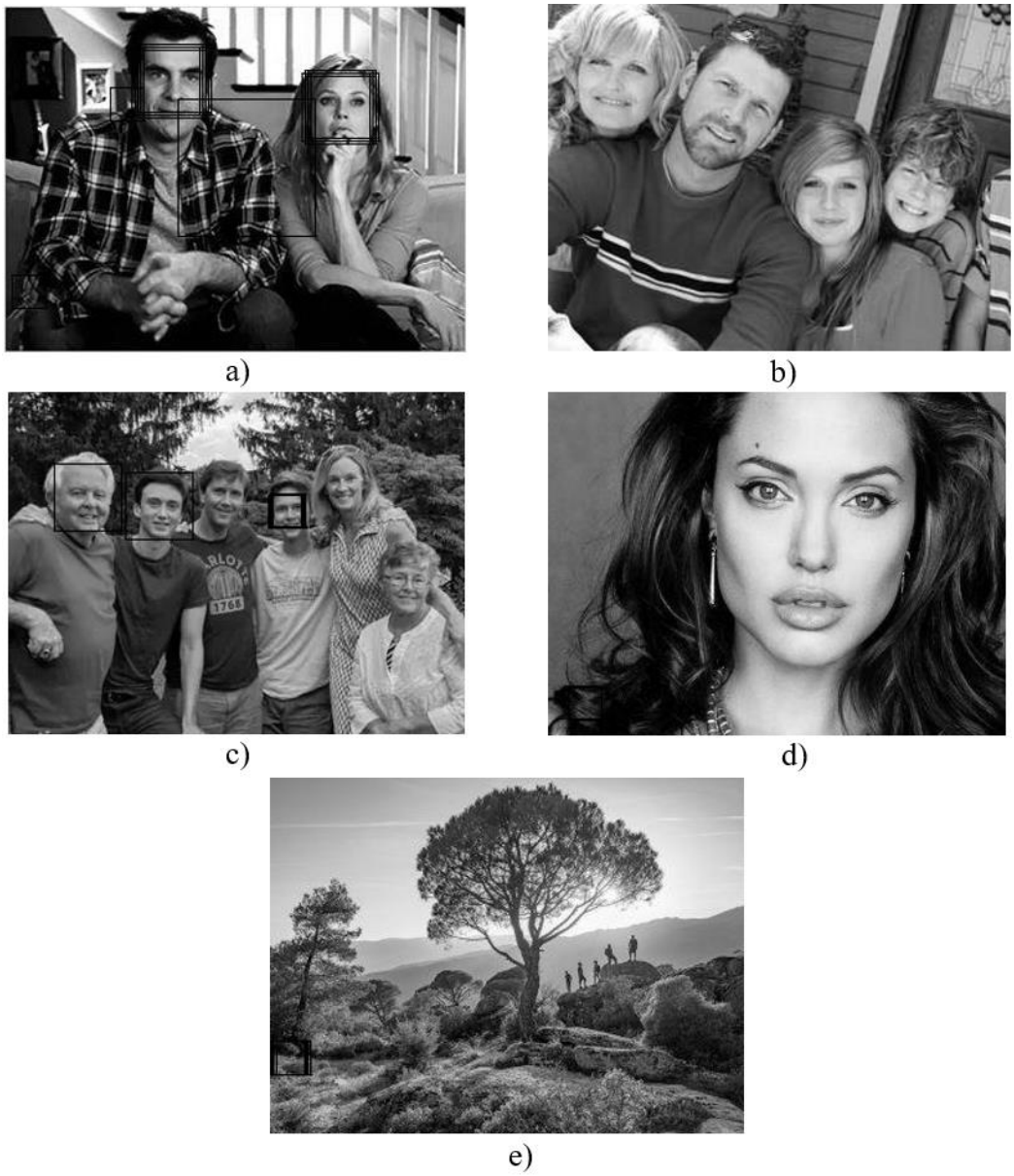
Sl. 4.2. Grafički prikaz prosječnog vremena obrade slike u ovisnosti o razlučivosti slike.

Slika 4.2. grafički prikazuje ovisnost prosječnog vremena obrade slike detekcijom lica za FPGA i *MATLAB* u ovisnosti o razlučivosti slike. Može se vidjeti kako vrijeme trajanja obrade slike raste sa povećanjem razlučivosti izvorne slike, te da *MATLAB* treba znatno više vremena od FPGA za obradu slike.

Prilikom testiranja predloženog rješenja algoritma za detekciju lica na slici pomoću FPGA, tablicom 4.1. prikazan je i prosjek od 41% loših detekcija zbog ograničenja predloženog rješenja. Najveća testirana izvorna slika je razlučivosti 320x240, a integralna slika je razlučivosti 39x59. Kako se izvorna slika može skalirati dok ne bude manja od integralne slike, prema potpoglavlju 3.4.5, navedena slika se može skalirati 2 puta. Navedeno rješenje za ovu sliku može detektirati 3 veličine lica koje određuje razlučivost podokvira, a to su razlučivosti: 24x24, 48x48 i 96x96. Lica koja ne stanu unutar navedenih okvira, neće biti detektirana.

Drugo ograničenje algoritma je određeno parametrima za detekciju lica treniranima na slikama u idealnim uvjetima. Na slikama odabranima za testiranje uvjeti ne moraju biti idealni, te dolazi do neželjenog ponašanja algoritma za detekciju lica.

Treće ograničenje algoritma je raspoloživa BRAM memorija na FPGA u koju je potrebno pohraniti parametre za detekciju lica. Kako bi se svojstvo algoritma za detekciju lica optimiziralo, u sustav za treniranje parametara potrebno je dodati više pozitivnih i negativnih uzoraka, što može povećati broj parametara za detekciju. Kada bi na raspolaganju bilo više BRAM memorije unutar navedenog FPGA, algoritmom bi se mogla vršiti detekcija slike u boji, te bi i komponente boje pridonijele boljem krajnjem rezultatu.



Sl. 4.3. Prikaz loše detektiranih testnih uzoraka slika nakon detekcije lica pomoću FPGA.

Na slici 4.3.a su vidljiva dva uspješno detektirana lica i dva lažno detektirana lica zbog povećanog broja detalja na slici, te nedovoljno negativnih uzoraka prilikom treninga klasifikatora. U sredini slike je prikazan i najveći podprozor za detekciju predloženog rješenja, te se svako lice veće od njega ne može detektirati. Na slici 4.3.b nije prikazana ni jedna detekcija na slici, jer tri lica na slici nisu uspravna, a preostalo lice nije obuhvaćeno okvirom za detekciju. Na slici 4.3.c prikazana je slika sa 3 uspješno detektirana lica i 3 lica koja nisu detektirana. Dva lica nisu detektirana, jer im lica nisu obuhvaćena dimenzijom podprozora ni nakon skaliranja slike, a jedno lice nije detektirano zbog naočala. Slika 4.3.d prikazuje lice na slici koje je preveliko za detekciju algoritmom FPGA, te iz navedenog razloga nije detektirano. Na slici 4.3.e

je prikazana slika bez lica na kojoj je pogrešno detektirano lice više puta na istom mjestu zbog malog broja uzorka negativnih slika prilikom treniranja parametara za detekciju. Na istoj slici unutar okvira za detekciju, može se uočiti sličnost uokvirenog oblika sa licem.

Kako bi se algoritam unaprijedio i smanjio količinu pogrešnih detekcija, potrebno je izvršiti trening boljih parametara za detekciju lica, sa više pozitivnih i negativnih uzoraka slika. Moguće je izmijeniti digitalni dizajn da se prilikom skaliranja slike, slika skalira sa sporije rastućim indeksom od zadanog 2^n (gdje je *n* nivo skaliranja slike). Navedeni pristup bi usporilo izvođenje algoritma za detekciju lica, ali bi povećao broj veličina lica koje je moguće detektirati. Za optimizaciju brzine algoritma za detekciju lica, moguće je uzimati svaki i-ti podprozor na provjeru što bi za razliku od predloženog rješenja smanjilo točnost algoritma, ali bi ubrzalo algoritam. Prilikom dovršetka računanja integralne slike, slobodne su dvije adresne i podatkovne veze na BRAM-u integralne slike, pa je moguće već postojećem podprozoru za detekciju dodati paralelni podprozor, te ubrzati algoritam bez da se utječe na točnost, ali sa povećanjem korištene površine i potrošnje energije *ZYNQ* SoC FPGA.

5. ZAKLJUČAK

Za ostvarivanje zadatka diplomskog rada bilo je potrebno u *MATLAB*-u razviti optimizirani algoritam za detekciju lica na slikama različitih razlučivosti. Navedeni algoritam opisati VHDL kodom, te opisanim digitalnim dizajnom u *ZYNQ* SoC FPGA obaviti detekciju lica na slici.

Predloženo rješenje prati Viola-Jones algoritam, po kojem je razvijan optimizirani algoritam unutar *MATLAB*-a. Predloženo rješenje je optimizirano u dva dijela digitalnog dizajna algoritma za detekcije lica na slici pomoću FPGA. Prvi dio optimizacije je izveden prilikom skaliranja slike, gdje se umjesto procesa skaliranja slike izvela manipulacija adresama za izravno čitanje skalirane slike iz izvorne slike, a drugi dio optimizacije algoritma je izveden kod crtanja okvira oko lica koje se izvršava paralelno uz detekciju podprozora.

Kako je cijeli digitalni dizajn opisan VHDL kodom izveden koristeći generičke varijable, lako ga je optimizirati za rad na drugim razlučivostima slika (pod uvjetom da slika željene razlučivosti stane u BRAM memoriju) i za detekciju bilo kojeg drugog objekta. Ovakva izvedba rješenja se može primijeniti za rad u stvarnom vremenu u kojima je potrebna pravovremena detekcija lica (ili detekcija bilo kojeg drugog objekta) uz poštivanje ograničenja izvedenog sustava.

Osim bolje treniranih parametara za detekciju objekta, izvedeno rješenje je moguće optimizirati po svrsi primjene, gdje svrha primjene određuje važnost preciznosti algoritma detekcije i brzine sustava. Izvedeno rješenje je moguće optimizirati paralelizacijom podsustava detekcije lica ili paralelnim izvođenjem više jednakih sustava sa ciljem detekcije lica na slici veće rezolucije, unutar jednakog vremenskog intervala.

Iz rezultata testiranja predloženog rješenja, može se zaključiti kako je algoritam za detekciju lica na slici pomoću FPGA znatno brži od algoritma za detekciju lica na slici pomoću *MATLAB*-a, dok je preciznost detekcije lica nešto manja. Osim navedene, prednost predloženog rješenja je u energetskoj efikasnosti FPGA.

LITERATURA

[1] P. Viola i M. Jones, "Rapid object detection using a boosted cascade of simple features," u *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 1:511*, 2001..

[2] Y. Freund i R. E. Schapire, "A decision - theoretic generalization of on-line learning and an application to boosting," u *Computational Learning Theory: Eurocolt '95*, Barcelona, 1995..

[3] Y.-Q. Wang, "An Analysis of the Viola-Jones Face Detection Algorithm," u *Image Processing On Line*, Cachan, Francuska, lipanj 2014..

[4] P. Irgens, C. Bader, T. Le i D. Saxena, "Parallelization of the Viola-Jones Face Detection Algorithm on an FPGA," Dept. of Electrical and Computer Engineering, Marquette University, Milwaukee, WI, USA, svibanj 2016.

[5] L. Acasandrei i A. Barriga, "FPGA implementation of an embedded face detection system based on LEON3," University of Seville, Seville, Spain, 2011.

[6] Xilinx, "Vivado," Xilinx, [Mrežno]. Available: https://www.xilinx.com/products/design-tools/vivado.html. [Pokušaj pristupa 7 rujan 2020].

[7] A. T. Ahmadi, "Amin-Ahmadi," Amin-Ahmadi, [Mrežno]. Available: http://amin-ahmadi.com/cascade-trainer-gui/. [Pokušaj pristupa 7 rujan 2020].

[8] DIGILENT, "ZYBO FPGA Board Reference Manual," 11 travanj 2016. [Mrežno]. Available: https://reference.digilentinc.com/_media/zybo:zybo_rm.pdf. [Pokušaj pristupa 7 rujan 2020].

[9] ARM, "AMBA AXI and ACE Protocol Specification," 2011. [Mrežno]. Available: http://www.gstitt.ece.ufl.edu/courses/fall15/eel4720_5721/labs/refs/AXI4_specification.pdf. [Pokušaj pristupa 7 rujan 2020].

[10] MathWorks, "MATLAB," MathWorks, [Mrežno]. Available: https://www.mathworks.com/products/matlab.html. [Pokušaj pristupa 7 rujan 2020].

SAŽETAK

Naslov: Detekcija lica na slici pomoću FPGA

Algoritmi za detekciju lica na slici pronalaze primjenu u širokom spektru industrijske djelatnosti kao što su automobilska industrija, industrija računalnih igara, vojna industrija i druge. U sklopu ovog diplomskog rada bilo je potrebno razviti optimizirani algoritam za detekciju lica na slici različitih razlučivosti, opisati digitalni dizajn VHDL jezikom, te pomoću FPGA obaviti detekciju lica na slici. Nakon obrade već postojećih rješenja, predloženo je rješenje algoritma za detekciju lica na slici pomoću FPGA, počevši od samog razvoja i optimizacije algoritma u *MATLAB*-u do krajnjeg digitalnog dizajna opisanog VHDL kodom i implementiranog na *ZYNQ* SoC FPGA. Na kraju rada dani su rezultati testiranja iz kojih je vidljivo de je postignuta detekcija lica s točnošću od 79% na slici razlučivosti 320x240 pomoću FPGA signala takta od 55.56MHz za prosječno vrijeme detekcije od 1.15 s. Predloženo rješenje je moguće prilagoditi za druge razlučivosti i za detekciju drugih objekata.

Ključne riječi: *detekcija, FPGA, lica, objekta, VHDL, Viola-Jones, ZYNQ*

ABSTRACT

Title: Face detection on the image using FPGA

Algorithms for face detection on the image find their application in a wide range of industrial activities such as the automotive industry, the computer game industry, the military industry and others. In this thesis, development an optimized algorithm for face detection on images of different resolutions, description of digital design in VHDL language, and face detection on images using FPGA are performed. After analysis of the existing solutions for the face detection algorithm, solution in this work was proposed. Elaboration of the proposed solutions starts from the development and optimization of the algorithm in MATLAB to the final digital design described in VHDL code and implemented on ZYNQ SoC FPGA. At the end of the paper, test results are given which show that face detection is achieved with an accuracy of 79% in the 320x240 resolution image using FPGA signal clock of 55.56MHz for an average detection time of 1.15 s. The proposed solution can be adjusted for other resolutions and for the detection of other objects.

Keywords: *detection, face, FPGA, object, VHDL, Viola-Jones, ZYNQ*

ŽIVOTOPIS

Marko Živković rođen je u Vinkovcima 8.2.1995. Od rođenja živi u selu Ivankovo koje se nalazi u okolici grada Vinkovci. Pohađao je osnovnu školu „August Cesarec“ u Ivankovu, te ju je završio sa odličnim uspjehom. Obrazovanje nastavlja sa srednjom školom „Tehnička Škola Ruđera Boškovića Vinkovci“ u Vinkovcima upisujući smjer elektrotehnike. Srednju školu završava radom „Izrada efekt pedale za gitaru“ koji je obranio sa odličnim uspjehom. Zbog stalnog interesa, želje za unaprjeđenjem svojega znanja i poznavanja elektrotehnike, 2014. godine upisuje preddiplomski studij elektrotehnike na Elektrotehničkom Fakultetu u Osijeku koji je promijenio naziv u „Fakultet Elektrotehnike, Računarstva i Informacijskih Tehnologija Osijek“ u Osijeku, te ga završava 2018. godine sa titulom sveučilišnog prvostupnika (baccalaureus) inženjera elektrotehnike i informacijske tehnologije. Iste godine nastavlja školovanje upisom diplomskog studija elektrotehnike, a 2019. godine predstavlja znanstveno istraživački rad „Ispitivanja Djelotvornosti Zaštitnog Oklapanja – Simulacija i Mjerenje“ na 12. Danima Inženjera Elektrotehnike u Vodicama.

_____

Potpis autora

PRILOZI

P.3.1. – Prilog se nalazi na priloženom multimedijskom disku.

P.3.2. – Prilog se nalazi na priloženom multimedijskom disku.

P.3.3. – Prilog se nalazi na priloženom multimedijskom disku.

P.3.4. – Prilog se nalazi na priloženom multimedijskom disku.

P.3.5. – Prilog se nalazi na priloženom multimedijskom disku.

P.3.6. – Prilog se nalazi na priloženom multimedijskom disku.

P.3.7. – Prilog se nalazi na priloženom multimedijskom disku.

P.3.8. – Prilog se nalazi na priloženom multimedijskom disku.

P.4.1. – Prilog se nalazi na priloženom multimedijskom disku.