

# Implementacija neuronske mreže za prepoznavanje znamenki u FPGA sustavu

---

Štajnbrikner, Matej

Master's thesis / Diplomski rad

2021

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

*Permanent link / Trajna poveznica:* <https://um.nsk.hr/um:nbn:hr:200:535168>

*Rights / Prava:* [In copyright](#)/[Zaštićeno autorskim pravom.](#)

*Download date / Datum preuzimanja:* **2025-03-29**

*Repository / Repozitorij:*

[Faculty of Electrical Engineering, Computer Science and Information Technology Osijek](#)



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

**Diplomski sveučilišni studij Računarstvo**

**IMPLEMENTACIJA NEURONSKE MREŽE ZA  
PREPOZNAVANJE ZNAMENKI U FPGA SUSTAVU**

**Diplomski rad**

**Matej Štajnbrikner**

**Osijek, 2021.**

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

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 04.09.2021.

Odboru za završne i diplomske ispite

**Imenovanje Povjerenstva za diplomski ispit**

Ime i prezime studenta:	Matej Štajnbriker
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1095R, 06.10.2019.
OIB studenta:	77136739745
Mentor:	Izv.prof.dr.sc. Tomislav Matić
Sumentor:	
Sumentor iz tvrtke:	Igor Valek
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Mario Vranješ
Član Povjerenstva 1:	Izv.prof.dr.sc. Tomislav Matić
Član Povjerenstva 2:	Izv.prof.dr.sc. Ratko Grbić
Naslov diplomskog rada:	Implementacija neuronske mreže za prepoznavanje znamenki u FPGA sustavu
Znanstvena grana rada:	<b>Elektronika (zn. polje elektrotehnika)</b>
Zadatak diplomskog rada:	U sklopu ovog diplomskog rada potrebno je izraditi model neuronske mreže za detekciju znamenki u rasponu vrijednosti od 0 do 9 iz slike veličine 28x28 piksela, koristeći pritom FPGA za digitalnu obradu prethodno učitanih slika u radnu memoriju. Nakon razvoja modela neuronske mreže u Programskom jeziku C, potrebno je izraditi digitalni dizajn u VHDL jeziku za opis fizičke arhitekture, izvršiti njegovu sintezu i optimizaciju u dostupnom FPGA razvojnom sustavu. Model, implementiran u FPGA sustavu, vrednovati provedbom mjerenja svojstava neuronske mreže koja uključuju broj ispravno i pogrešno detektiranih znamenki iz slike, brzinu treniranja i izvršavanja neuronske mreže te potrošnju električne energije prilikom izvođenja rada algoritma pojedine periferije unutar FPGA sustava. Tema
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: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	04.09.2021.
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, 10.09.2021.

**Ime i  
prezime  
studenta:**

Matej Štajnbriker

**Studij:**

Diplomski sveučilišni studij Računarstvo

**Mat. br.  
studenta,  
godina  
upisa:**

D-1095R, 06.10.2019.

**Turnitin  
podudaranje  
[%]:**

3

Ovom izjavom izjavljujem da je rad pod nazivom: **Implementacija neuronske mreže za prepoznavanje znamenki u FPGA sustavu**

izrađen pod vodstvom mentora Izv.prof.dr.sc. Tomislav Matić

i sumentora

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

<b>1. UVOD.....</b>	<b>1</b>
<b>1.1. Zadatak diplomskog rada.....</b>	<b>2</b>
<b>2. NEURONSKE MREŽE .....</b>	<b>3</b>
<b>2.1. Umjetne neuronske mreže .....</b>	<b>3</b>
<b>2.2. Konvolucijske neuronske mreže .....</b>	<b>5</b>
<b>3. PREPOZNAVANJE ZNAMENKI POMOĆU NEURONSKIH MREŽA ....</b>	<b>9</b>
<b>3.1. Pregled postojećih rješenja.....</b>	<b>9</b>
3.1.1. Konvolucijska neuronska mreža .....	10
3.1.2. Binarna neuronska mreža .....	11
3.1.3. Umjetna neuronska mreža – višeslojni perceptron.....	12
<b>4. MODEL I TRENIRANJE NEURONSKE MREŽE.....</b>	<b>16</b>
<b>4.1. Model neuronske mreže.....</b>	<b>16</b>
<b>4.2. Propagacija unaprijed .....</b>	<b>17</b>
<b>4.3. Propagacija unazad.....</b>	<b>20</b>
<b>5. IMPLEMENTACIJA SUSTAVA NA RAZVOJNU PLOČU.....</b>	<b>27</b>
<b>5.1. Korišteni alati i tehnologije .....</b>	<b>27</b>
5.1.1. ZYBO .....	27
5.1.2. FPGA.....	29
5.1.3. Vivado .....	31
<b>5.2. Ulaz i izlaz sustava .....</b>	<b>32</b>
<b>5.3. Komunikacija PS-PL .....</b>	<b>33</b>
<b>5.4. Blok za propagaciju unaprijed .....</b>	<b>34</b>
5.4.1. Sinkronizacijski blok.....	35
5.4.2. Množilo .....	36
5.4.3. Detektor zadnjeg podatka.....	37
5.4.4. Akumulator.....	37
5.4.5. Aktivator i FIFO zapisivač .....	38

<b>5.5. Blok za propagaciju unazad .....</b>	<b>39</b>
5.5.1. MUX – pogreška .....	41
5.5.2. Sinkronizacijski blok 0 i množilo 0.....	42
5.5.3. Sinkronizacijski blok 1 i množilo 1.....	43
5.5.4. Sinkronizacijski blok 2 i blok za oduzimanje .....	43
5.5.5. FIFO zapisivač .....	45
5.5.6. Sinkronizacijski blok 3 i množilo 2.....	45
5.5.7. FIFO izlazi iz neurona – mali međuspremnik.....	46
5.5.8. Sinkronizacijski blok 4 i zbrajalo .....	47
5.5.9. Demultiplekser, multiplekser i RAM memorije.....	48
5.5.10. Upravljač RAM memorijama .....	49
5.5.11. Upravljačka jedinica.....	50
<b>5.6. Pregled potpunog rješenja.....</b>	<b>52</b>
<b>5.7. Programska podrška.....</b>	<b>55</b>
5.7.1. Glavni parametri aplikacije .....	55
5.7.2. Upravljački program za trenirajući način rada sklopovlja .....	57
5.7.3. Upravljački program za klasifikacijski način rada sklopovlja .....	59
5.7.4. Glavne funkcionalnosti aplikacije .....	60
<b>6. REZULTATI MJERENJA .....</b>	<b>62</b>
<b>7. ZAKLJUČAK.....</b>	<b>72</b>
<b>LITERATURA.....</b>	<b>73</b>
<b>SAŽETAK.....</b>	<b>77</b>
<b>ABSTRACT .....</b>	<b>78</b>
<b>PRILOG.....</b>	<b>79</b>
<b>ŽIVOTOPIS.....</b>	<b>81</b>

# 1. UVOD

Razvojem računala u 21. stoljeću, neuronske mreže nalaze sve veću praktičnu primjenu. Najčešće nalaze primjenu u aproksimaciji nelinearnih funkcija, klasifikaciji uzoraka, obradi i filtriranju podataka, slijednom odlučivanju, robotici itd. Jedna od najpopularnijih primjena neuronskih mreža na kojoj se intenzivno radi napredni su sustavi za pomoć vozaču (engl. *Advance Driver – Assistance Systems* - ADAS), točnije, dio takvog sustava koji prepoznaje objekte u okolini automobila. Neke tvrtke za proizvodnju automobila već postižu rezultate na polju autonomnog upravljanja vozilom, ali nedovoljno pouzdane da bi se vozilo sigurno kretalo bez pomoći vozača [1]. Budući da je autonomno upravljani automobil sustav stvarnog vremena s kritičnim vremenskim zahtjevima, računala u takvim automobilima imaju velike zahtjeve na performanse čime dolazi do povećane potrošnje energije. Zbog prethodno navedenog sustavi s kritičnim vremenskim zahtjevima, koji implementiraju neki oblik neuronske mreže, danas koriste grafičke procesore (engl. *Graphics Processing Unit* - GPU) za treniranje i izvršavanje istreniranih neuronskih mreža. Budući da grafički procesori troše veliku količinu energije tijekom izvršavanja operacija s neuronskim mrežama, u zadnje vrijeme se počinje razmatrati korištenje programibilnih logičkih polja (engl. *Field Programmable Gate Array* - FPGA) i aplikacijsko specifičnih integriranih krugova (engl. *Application Specific Integrated Circuit* - ASIC) za tu primjenu.

FPGA se danas u industriji pojavljuje sve češće zbog male potrošnje energije, relativno male cijene i visokih performansi koje se postižu visokim stupnjem paralelizacije zadataka i razvojem sklopovlja specifične namjene. Iako je implementacija neuronskih mreža na FPGA relativno nova tema, znanstvenici dobivaju zadovoljavajuće rezultate s obzirom na potrošnju energije i dobivene performanse.

Implementacija neuronskih mreža za prepoznavanje ručno pisanih znamenki iz MNIST (engl. *Modified National Institute of Standards and Technology*) baze podataka već se duže vrijeme smatra odličnim primjerom za razumijevanje neuronskih mreža i računalnog vida. Za rješavanje ovog problema najbolje su se pokazale kombinacije konvolucijskih, rekurentnih i potpuno povezanih neuronskih mreža. Točnost klasifikacije znamenaka, korištenjem takve mreže, prelazi 99.7 %.

U okviru ovog diplomskog rada na Zynq-7000 sustavu na čipu (engl. *System on Chip* - SoC), koji je dio ZYBO razvojne ploče, implementirana je potpuno povezana duboka neuronska mreža za prepoznavanje znamenki iz MNIST baze podataka. Mreža je zatim istrenirana te su vrednovani

postotak točnosti klasifikacije znamenaka, performanse sustava te zauzeće resursa Zynq-7000 sustava na čipu. Također, dobiveni rezultati su uspoređeni s rezultatima postojećih rješenja.

Rad je strukturiran na slijedeći način. U drugom poglavlju opisane su neuronske mreže te pripadajuća podjela na najčešće korištene vrste.. Svaka od navedenih vrsti, zasebno je opisana. U trećem poglavlju dan je pregled postojećih rješenja na temu prepoznavanja znamenki pomoću neuronskih mreža. Nadalje, u četvrtom poglavlju dana su teorijska razmatranja implementirane neuronske mreže, a peto poglavlje detaljno opisuje implementaciju sustava na razvojnu ploču. U šestom poglavlju dani su rezultati mjerenja te su isti uspoređeni s rezultatima postojećih rješenja iz trećeg poglavlja. Na kraju su izvedena zaključna razmatranja o uspješnosti implementacije, rezultatima mjerenja i dane su upute o mogućim poboljšanjima predloženog algoritma.

## **1.1. Zadatak diplomskog rada**

Zadatak ovog diplomskog rada je izraditi model neuronske mreže za detekciju znamenki u rasponu vrijednosti od 0 do 9 iz slike veličine 28x28 piksela, koristeći pritom FPGA za digitalnu obradu prethodno učitanih slika u radnu memoriju. Nakon razvoja modela neuronske mreže u programskom jeziku C, potrebno je izraditi digitalni dizajn u VHDL jeziku za opis fizičke arhitekture, izvršiti njegovu sintezu i optimizaciju u Vivado softverskom paketu te implementirati sustav na ZYBO razvojnu ploču. Model, implementiran na razvojnom FPGA sustavu, treba vrednovati provedbom mjerenja svojstava neuronske mreže koja uključuju broj ispravno i pogrešno detektiranih znamenki iz slike, brzinu treniranja i izvršavanja neuronske mreže te potrošnju električne energije prilikom izvođenja rada algoritma pojedine periferije unutar razvojnog FPGA sustava.



## 2. NEURONSKE MREŽE

Neuronske mreže prvi puta su se počele spominjati krajem 19. stoljeća, a prvi moderni model neuronske mreže otkrili su Warren McCulloch i Walter Pitts 1940-ih godina [2]. Prva se praktična primjena neuronske mreže pojavila krajem 1950-ih. Tada je Frank Rosenblatt osmislio perceptron i odgovarajuće pravilo treniranja takve mreže. Nakon tog otkrića, mnogi znanstvenici mislili su da daljnje istraživanje ne može biti korisno, ali 1970-ih ponovno dolazi do velikih otkrića na polju neuronskih mreža, a 1980-ih godina otkriva se algoritam propagacije unazad (engl. *Back-propagation algorithm*) za treniranje višeslojnih perceptron mreža (engl. *Multilayer Perceptron Networks* – MLP) što se smatra jednim od ključnih otkrića. Iako su se neuronske mreže kontinuirano razvijale od svog otkrića, performanse računala su do početka 21. stoljeća činile usko grlo u praktičnom korištenju istih [3].

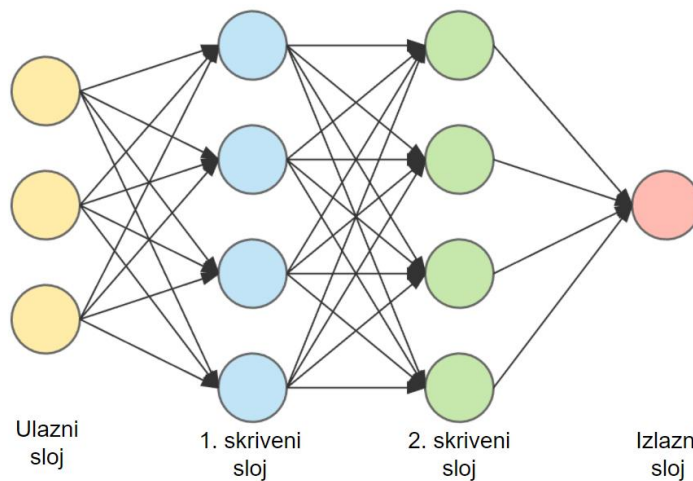
Neuronske mreže nalaze primjenu u bankarskim sustavima, automobilskoj, zrakoplovnoj i vojnoj industriji, elektronici, zabavi, financijama, telekomunikacijama, robotici itd. Široki spektar mogućnosti primjene neuronskih mreža privlači ljude da se bave istima pa se danas na većini sveučilišta barem u nekom obliku spominju neuronske mreže, a mnoge velike tvrtke poput Google-a i Microsoft-a ih primjenjuju na svakodnevnoj razini [3].

Prema [4], može se razlikovati više od 20 vrsta neuronskih mreža, no u ovom slučaju detaljnije će biti objašnjene 3 najčešće korištene, a to su umjetne neuronske mreže (engl. *Artificial Neural Networks* - ANN), konvolucijske neuronske mreže (engl. *Convolution Neural Networks* - CNN) i rekurentne neuronske mreže (engl. *Recurrent Neural Networks* - RNN) [5]. U ovome poglavlju opisane su umjetne i konvolucijske neuronske mreže koje najčešće služe za rješavanje problema poput onoga koji se razmatra u ovom diplomskom radu.

### 2.1. Umjetne neuronske mreže

Umjetne neuronske mreže mogu se definirati kao skup neurona raspoređenih u više slojeva koji su međusobno povezani težinama koje je moguće trenirati za određenu primjenu. Najčešće su strukturirane tako da imaju jedan ulazni sloj (engl. *Input layer*), jedan ili više skrivenih slojeva (engl. *Hidden layers*) i jedan izlazni sloj (engl. *Output layer*). Primjer umjetne neuronske mreže s jednim ulaznim, dva skrivena i jednim izlaznim slojem, vidljiv je na slici 2.1. Strelice sa slike prikazuju težine. Ovakva neuronska mreža još se naziva i potpuno povezanom jer je svaki neuron povezan sa svim ostalim neuronima iz njemu susjednih slojeva. Ulazni sloj mreže prihvaća ulazne podatke, skriveni slojevi obrađuju ulazne podatke, a izlazni sloj proizvodi rezultat takav da je

razumljiv korisniku (npr. rezultat je 1 ako je dan sunčan, a 0 ako nije). Ovakve mreže, sposobne su rješavati probleme zadane tabličnim podacima, slikama i tekstualnim podacima [5].



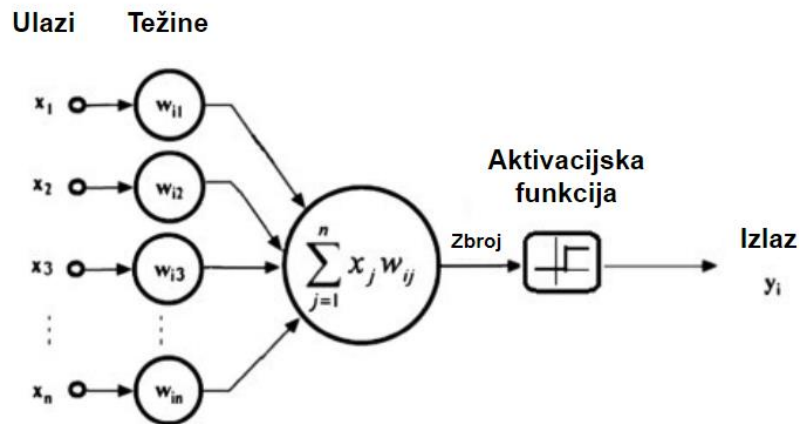
Slika 2.1 - Umjetna neuronska mreža [6]

Najveća prednost umjetnih neuronskih mreža je sposobnost učenja bilo koje nelinearne funkcije pa se zbog toga često zovu univerzalni aproksimatori funkcija (engl. *Universal Function Approximators*). Razlog tome je aktivacijska funkcija koja unosi nelinearne značajke u mrežu. Neke od najkorištenijih aktivacijskih funkcija su ReLU (engl. *Rectified Linear Unit*), Sigmoid, tanh, Softmax i Leaky ReLU. Tablica s detaljima o aktivacijskim funkcijama nalazi se u prilogu P.1 [7]. Za skrivene slojeve najčešće je korištena funkcija ReLU zbog svoje jednostavnosti i učinkovitosti, a za izlazni sloj najčešće se koristi Softmax, i to za problem klasifikacije jer na izlazu daje vjerojatnosti pojavljivanja određenih klasa čiji se objekti pojavljuju na ulazu mreže. Aktivacijska funkcija izvodi se na izlazu svakog neurona iz skrivenih slojeva ili izlaznog sloja pa se izlaz iz neurona često naziva aktivacijom [5] [7].

Neuron se općenito može definirati pomoću definicije logističke regresije. Izlaz iz neurona uvijek ovisi o ulazu i neuron se općenito koristi za kategorizaciju ulaza na izlazu. Npr. perceptron je usko vezan za binarnu logističku regresiju jer na izlazu daje 0 ili 1 [9]. Način funkcioniranja neurona prikazan je na slici 2.2. Izlaz iz neurona  $y_i$  ili aktivacija dobije se tako da se svi ulazi u neuron (aktivacije iz prethodnog sloja ili ulazi u mrežu) i pripadajuće težine pomnože, zatim se ti umnošci zbroje, a dobiveni rezultat ulazi u aktivacijsku funkciju koja daje konačan rezultat.

Prema prethodnom odjeljku može se izvesti algoritam propagacije unaprijed (engl. *Feed-forward algorithm*). Počevši od prvog skrivenog sloja u mreži, za svaki neuron izvode se operacije

kao što je prikazano na slici 2.2, a u ovom slučaju ulazi  $x_i$  sa slike su ulazi u mrežu. Nakon što su izračunate aktivacije prvog skrivenog sloja, one služe kao ulaz za idući skriveni ili izlazni sloj. Postupak se ponavlja sve dok se ne izračunaju izlazi iz mreže [10] [3].



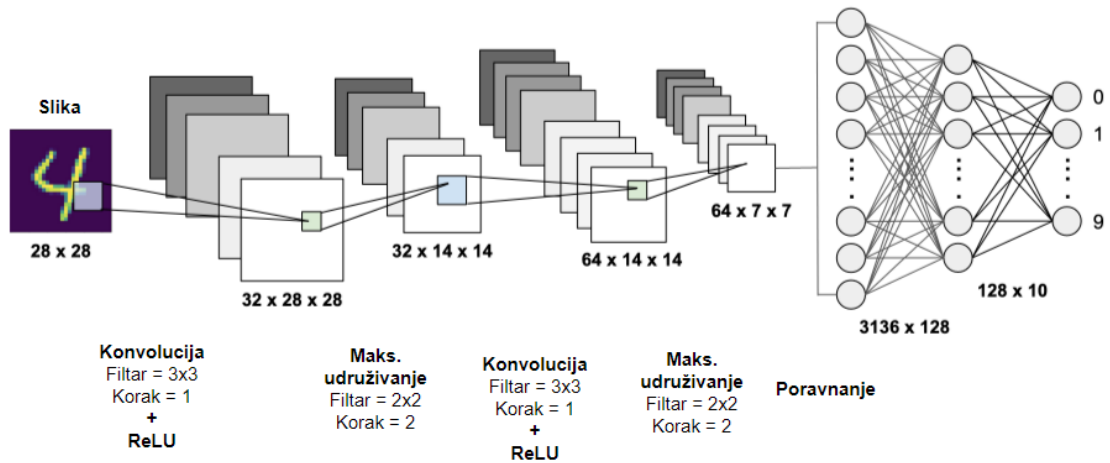
Slika 2.2 – Neuron [11]

Umjetne neuronske mreže treniraju se pomoću algoritma propagacije unazad. Princip treniranja ovakve mreže je slijedeći. Jedan uzorak iz skupa ulaznih podataka propagira se unaprijed kroz mrežu, a zatim se prema očekivanom i dobivenom izlazu iz mreže računaju pogreške. Pogreške na izlaznom sloju mreže propagiraju se unazad kroz mrežu te se prema gradijentima pogreške na određenom neuronu podešavaju težine vezane za taj neuron. Ovaj postupak ponavlja se za svaki uzorak iz skupa ulaznih podataka. Jedna epoha treniranja završena je kada su svi uzorci iz ulaznog skupa obrađeni. Nadalje, epohe treniranja izvršavaju se sve dok mreža nije dovoljno utrenirana [10] [3].

## 2.2. Konvolucijske neuronske mreže

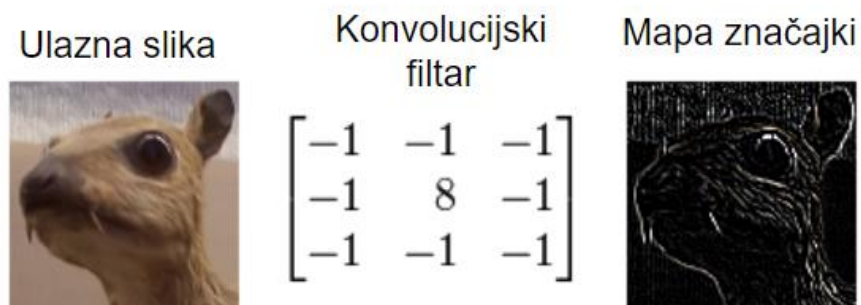
Konvolucijske neuronske mreže složene su neuronske mreže dizajnirane za analizu slike iako nalaze primjenu i u drugim područjima, kao što je procesiranje materinjeg jezika. Najčešće su strukturirane od više konvolucijskih slojeva (filtri ili kerneli) i slojeva za udruživanje (engl. *Pooling layers*) te potpuno povezanog sloja (umjetna neuronska mreža) na izlaznom dijelu neuronske mreže. Konvolucijske neuronske mreže treniraju se kao i umjetne, postupkom propagacije unazad. Također, važno je napomenuti da sloj za udruživanje uvijek dolazi nakon konvolucijskog sloja. Primjer konvolucijske neuronske mreže s dva konvolucijska sloja, dva sloja za udruživanje i tri potpuno povezana sloja na izlazu može se vidjeti na slici 2.3. Originalna slika razlučivosti 28x28 obrađuje se konvolucijskim filtrima, a zatim se rezultati konvolucije udružuju

(komprimiraju) u sloju za udruživanje na matrice razlučivosti 14x14. Zatim se ti rezultati opet obrađuju konvolucijskim filtrima, a rezultati konvolucije komprimiraju se u sloju za udruživanje i to na matrice razlučivosti 7x7 [12] [13].



Slika 2.3 - Konvolucijska neuronska mreža [14]

Konvolucija je transformacija definirana skupom težina koja se na sliku primjenjuje piksel po piksel. Skup težina još se naziva i filter. Filter se može promatrati kao kvadratna matrica dimenzije  $n$ , gdje je  $n$  često 3 ili 5, a korak (engl. *stride*) kretanja filtera po slici najčešće je 1. Jedna od zadaća konvolucije ekstrakcija je značajki sa slike kao što su ravne linije, točke, krugovi, kose linije, itd. Također, postoje konvolucijski filtri kojima se može izoštriti ili zamutiti slika ili npr. detektirati rubove sa slike. Na slici 2.4 moguće je vidjeti kakav rezultat daje konvolucija filtrom za detekciju rubova. Zbog svojstva ekstrakcije značajki (isticanje važnih značajki sa slike), konvolucijski slojevi u neuronskim mrežama značajno poboljšavaju rezultate detekcije objekata sa slika [15].



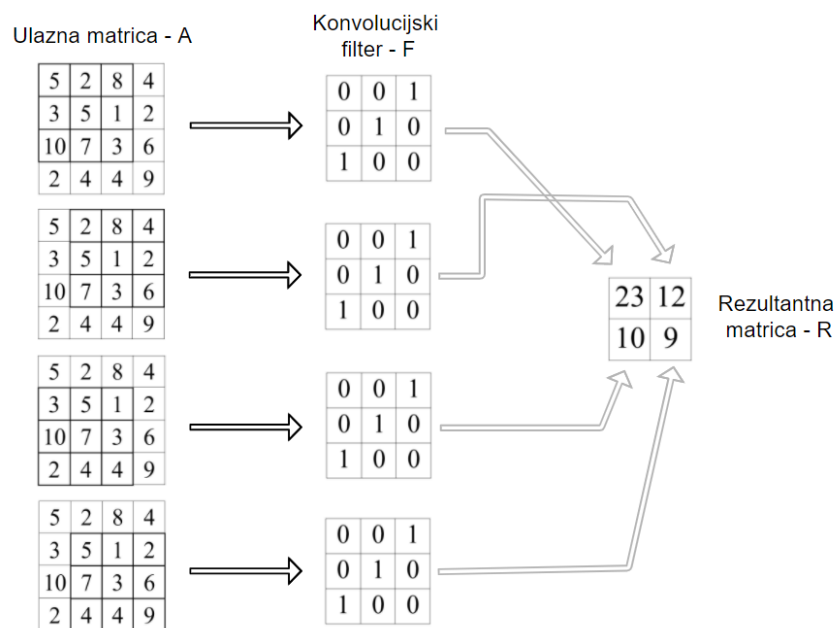
Slika 2.4 - Rezultat konvolucije filtrom za detekciju rubova [16]

Na slici 2.5 može se vidjeti primjer izvršavanja konvolucije filtrom  $F$  nad matricom  $A$ , gdje korak kretanja filtra po slici iznosi 1. Rezultat konvolucije spremljen je u matricu  $R$ :

$$R = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} \times F_{ij} \quad (2-1)$$

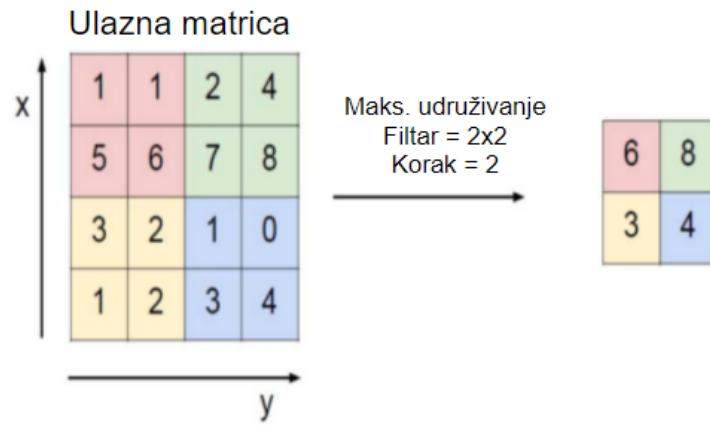
gdje je  $a_{ij}$  podmatrica matrice  $A$ ,  $n$  dimenzija matrice filtra, a  $F_{ij}$  matrica filtra.

Promatrajući sliku 2.5 i izraz (2-1), može se zaključiti da je rezultat 23 u matrici  $R$  dobiven primjenjivanjem formule na matricu  $a$  (gornja lijeva podmatrica matrice  $A$ ) i matricu  $F$ .



Slika 2.5 – Algoritam konvolucije u neuronskim mrežama

Zadaća slojeva za udruživanje u konvolucijskim neuronskim mrežama smanjivanje je uzoraka koji su rezultati konvolucije, čime se smanjuje složenost idućih slojeva u mreži. Pritom se važne informacije iz slike (istaknute značajke) ne gube, a mreža dobiva na brzini izvođenja. Najčešće korišten algoritam udruživanja je maksimalno udruživanje (engl. *Max pooling*), koje uzima najveći element iz promatrane podmatrice i vraća ga kao rezultat. Dimenzija podmatrice nad kojom se izvršava algoritam maksimalnog udruživanja najčešće je 2x2, a korak (engl. *stride*) kojom se podmatrica kreće po matrici najčešće je 2 [15]. Primjer izvršavanja ovog algoritma s prethodno spomenutim parametrima može se vidjeti na slici 2.6.



Slika 2.6 - Algoritam maksimalnog udruživanja [15]

### 3. PREPOZNAVANJE ZNAMENKI POMOĆU NEURONSKIH MREŽA

Problem prepoznavanja znamenki pomoću neuronskih mreža često se pojavljuje u domeni računalnog vida i dubokog učenja, a jedan od primjera je pretvorba teksta sa slike u tekstualni dokument. Postoji više baza za učenje neuronskih mreža za prepoznavanje znamenki, ali baza ručno pisanih znamenki MNIST prepoznata je od mnogih inženjera i znanstvenika kao najrelevantnija te je postala standard u području računalnog vida i dubokog učenja [17]. Iako već neko vrijeme postoje učinkovita rješenja za problem klasifikacije znamenki iz te baze, ona je još uvijek dobar primjer za razumijevanje tog područja [18]. Neki od zahtjeva koje treba ispuniti pri rješavanju ovog problema su: minimalno zauzeće resursa, maksimalna brzina izvođenja i minimalna potrošnja energije. Budući da grafički procesori, često korišteni za operacije s neuronskim mrežama, troše relativno veliku količinu energije, u zadnje vrijeme se sve više razvijaju rješenja zasnovana na FPGA tehnologiji. U ovome poglavlju dan je pregled postojećih rješenja, usko vezanih za implementaciju neuronske mreže za prepoznavanje znamenki na FPGA.

Rješavanje problema prepoznavanja znamenki u FPGA sustavu temelji se na sljedećem:

- Propagacija unaprijed:
  - Sustav na ulazu treba prihvaćati sliku iz baze te parametre mreže.
  - Poželjno je da sustav u što kraćem vremenu i sa što manje zauzetih resursa na izlazu daje vjerojatnosti pojavljivanja mogućih znamenki.
- Propagacija unazad:
  - Sustav treba uzeti u obzir pogreške izlaznog sloja, prethodne izlaze iz neurona te konfiguraciju mreže (npr. težine) koju treba korigirati s ciljem točnijeg prepoznavanja znamenaka.
  - Korigiranje parametara mreže je poželjno izvesti u što kraćem vremenu i sa što manje zauzetih resursa.

#### 3.1. Pregled postojećih rješenja

U sljedećim potpoglavljima dan je pregled radova koji se bave rješavanjem problema prepoznavanja znamenki pomoću neuronskih mreža. Pregled pojedinog rada obuhvaća opis strukture mreže, detalje implementacije mreže na FPGA sustavu i prikaz dobivenih rezultata.

### 3.1.1. Konvolucijska neuronska mreža

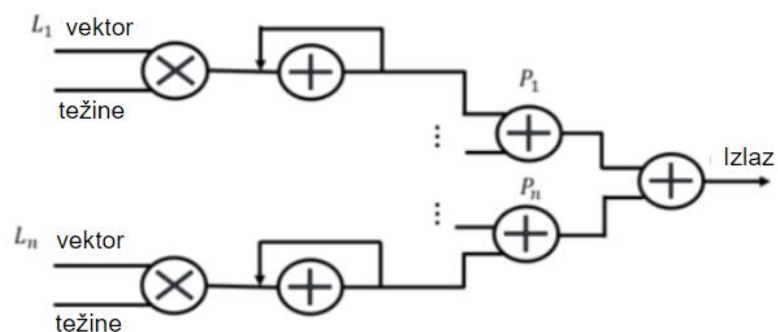
U radu [19] prikazan je sustav za prepoznavanje znamenki koji implementira konvolucijsku neuronsku mrežu. Struktura mreže prikazana je na slici 3.1. Na ulaz mreže dovodi se slika iz MNIST baze, razlučivosti 28x28. Slika se prvo obrađuje s 3 konvolucijska filtra, rezultati se udružuju, a zatim se rezultati udruživanja obrađuju s 4 konvolucijska filtra i ponovno udružuju. Na izlazu mreže nalazi se potpuno povezani sloj sa 100 ulaznih, 16 skrivenih i 10 izlaznih neurona.



Slika 3.1 - Struktura konvolucijske neuronske mreže [19]

Sustav je u potpunosti rekonfigurabilan, što znači da se sve vrste slojeva mogu postaviti prema korisničkim zahtjevima. To uključuje dimenziju konvolucijskog filtra, broj filtara, dimenziju udruživačkog filtra, korake filtara, broj neurona u potpuno povezanim slojevima itd. Tip podatka korišten u cijeloj mreži je 18-bitni broj s nepomičnim zarezom (engl. *Fixed-point*), od čega decimalni dio zauzima 13 bita. Konvolucijski sloj paraleliziran je tako da se u isto vrijeme izvodi više konvolucijskih filtriranja što rezultira istovremenim nastajanjem više rezultata konvolucije (mape značajki). Paralelizacija sloja za udruživanje, realizirana je povezivanjem jednog modula za udruživanje na svaki konvolucijski modul, gdje jedan konvolucijski modul odgovara jednom konvolucijskom filtru. Potpuno povezani sloj paraleliziran je tako da se ulazni skup težina i ulaza dijeli na više podskupova koji se zasebno sekvencijalno množe i akumuliraju, a zatim se akumulacije svih podskupova zbroje i rezultat se aktivira. Modul za paralelno izvođenje potpuno povezanog sloja vidljiv je na slici 3.2.





Slika 3.2 - Modul za paralelno izvođenje potpuno povezanog sloja [19]

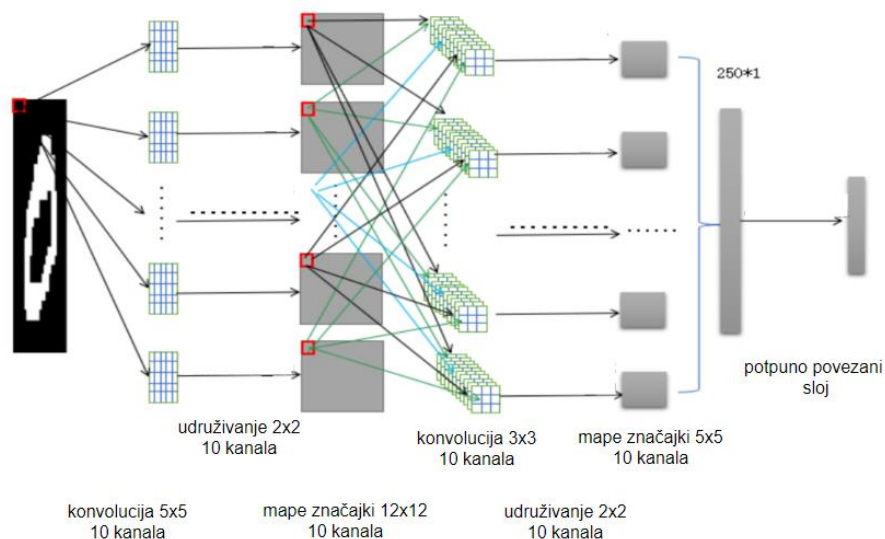
Neuronska mreža je prije implementacije na FPGA, implementirana i istrenirana na računalu. Zatim je digitalni sustav dizajniran te implementiran u Intel Cyclone 10 FPGA sustavu. Pri maksimalnoj frekvenciji rada sustava koja iznosi 150 MHz, mreža klasificira znamenku za 0,0176 ms, a postotak točno klasificiranih znamenaka iznosi 97.57 %. Zauzeće resursa Intel Cyclone 10 FPGA sustava prikazano je tablicom 3.1.

Tablica 3.1 - Zauzeće resursa Intel Cyclone10 FPGA sustava [19]

TIP BLOKA	KOLIČINA ZAUZETIH BLOKOVA
DSP	274
FF	48765
LUT	12588

### 3.1.2. Binarna neuronska mreža

U radu [20] prikazan je sustav za prepoznavanje znamenki koji implementira binarnu neuronsku mrežu (binarizirana konvolucijska mreža). Struktura mreže prikazana je na slici 3.3. Ulaz mreže je slika iz MNIST baze, razlučivosti 28x28. Slika se prvo obrađuje s 10 konvolucijskih filtara (filar dimenzije 5x5), rezultati se udružuju, a zatim se rezultati udruživanja obrađuju s 10 konvolucijskih filtara (filar dimenzije 3x3) i ponovno udružuju. Na izlazu mreže nalazi se potpuno povezani sloj sa 250 ulaznih i 10 izlaznih neurona.



Slika 3.3 - Struktura binarne (konvolucijske) neuronske mreže [20]

Konvolucijska neuronska mreža u ovom radu binarizirana je tako da su izlazi iz konvolucijskih slojeva i slojeva za udruživanje mapirani na vrijednosti -1 ili 1:

$$\text{sign}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \quad (3-1)$$

gdje je  $x$  ulaz u aktivacijsku funkciju.

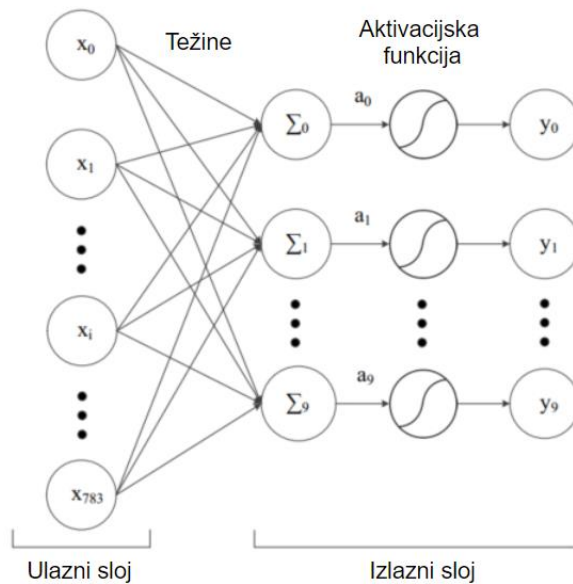
Budući da je mreža binarizirana, korištena je aktivacijska funkcija sign prikazana izrazom (3-1). Konvolucijski slojevi paralelizirani su tako da se paralelno izvodi deset filtara, a svaki od filtara ima 25 aritmetičkih jedinica koje paralelno izvode operaciju konvolucije. Potpuno povezani sloj je paraleliziran na način da se svih 10 mapa značajki na ulazu u potpuno povezani sloj u isto vrijeme koristi za generiranje izlaza iz mreže, što znači da je razina paralelizacije 10.

Neuronska mreža je prije implementacije na FPGA, implementirana i istrenirana na računalu, a zatim je digitalni sustav dizajniran i implementiran na DE2i-150 FPGA sustav. Pri maksimalnoj frekvenciji rada sustava koja iznosi 100 MHz, mreža klasificira znamenku za 18  $\mu$ s, a postotak točno klasificiranih znamenki iznosi 85%. Maksimalna snaga FPGA sustava iznosi 0.136 W.

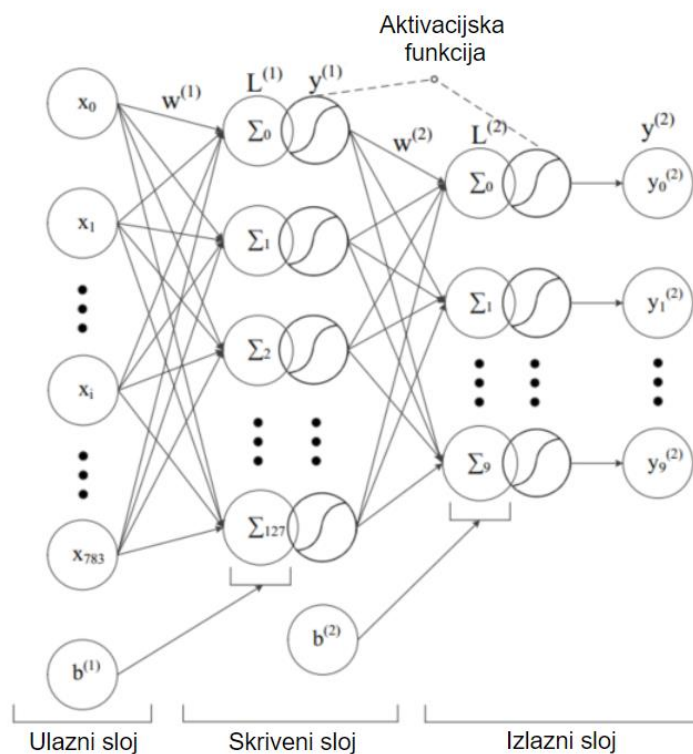
### 3.1.3. Umjetna neuronska mreža – višeslojni perceptron

U radu [21] prikazan je sustav za prepoznavanje znamenki koji implementira višeslojnu perceptron mrežu. Dane su dvije strukture mreže: dvoslojna (Slika 3.4) i troslojna (Slika 3.5). Ulaz mreže je slika iz MNIST baze, razlučivosti 28x28. Dvoslojna mreža sadrži 784 neurona na ulazu

i 10 neurona na izlazu, dok troslojna mreža sadrži 784 neurona na ulazu, 128 neurona u skrivenom sloju i 10 neurona na izlazu. Dvoslojna mreža trenirana je na FPGA sustavu, a troslojna mreža na računalu. Također, prikazani su rezultati dobiveni korištenjem aktivacijskih funkcija sigmoid i dinamički ReLU (engl. *Dynamic ReLU*).

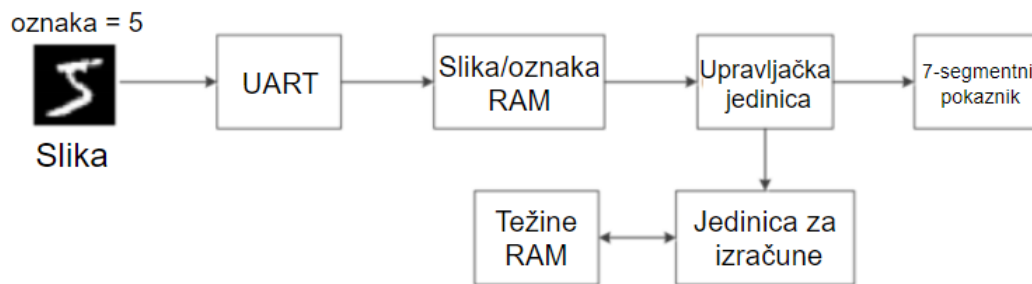


Slika 3.4 - Dvoslojna MLP neuronska mreža [21]



Slika 3.5 - Troslojna MLP neuronska mreža [21]

Slika se zajedno s očekivanim rezultatom klasifikacije (oznakom) prenosi UART (engl. *Universal Asynchronous Receiver-Transmitter*) komunikacijskim protokolom s računala na FPGA sustav i sprema u RAM (engl. *Random Access Memory*). Težine su također spremljene u RAM rezerviran za njih. Zatim se slijedno izvršavaju izračuni u neuronskoj mreži, tj. nema paralelizacije. Npr. u dvoslojnoj mreži se prvo računa izlaz prvog neurona u izlaznom sloju pa izlaz drugog neurona u izlaznom sloju itd. Treniranje mreže (propagacija unazad) na FPGA sustavu također se izvodi bez paralelizacije. Rezultat klasifikacije prikazuje se na 7-segmentnom pokazniku (engl. *7-segment display*) FPGA sustava. Budući da je u slučaju implementacije dvoslojne mreže u ovom radu omogućeno treniranje mreže na FPGA sustavu, UART-om se može postaviti način rada sustava: treniranje ili klasifikacija. Arhitektura sustava implementiranog u FPGA može se vidjeti na slici 3.6.



Slika 3.6 - Arhitektura sustava implementiranog u FPGA [21]

Digitalni sustav je dizajniran, a zatim implementiran u Intel Cyclone IVE FPGA sustavu. Tablica 3.2 prikazuje postotak točno klasificiranih znamenaka u ovisnosti o broju slojeva mreže i aktivacijskoj funkciji, a tablica 3.3 prikazuje vrijeme izvođenja klasifikacije znamenke i frekvenciju rada sustava u ovisnosti o broju slojeva mreže i aktivacijskoj funkciji. Maksimalno zauzeće logičkih elemenata FPGA sustava u slučaju troslojne mreže iznosi 34000, a za aktivacijsku funkcija D-ReLU dobiven je najbolji omjer brzine i točnosti. Postotak točno klasificiranih znamenaka na FPGA sustavu koji koristi 8-bitne podatke i računalu koje koristi 32-bitne podatke, jednak je.

Tablica 3.2 - Postotak točno klasificiranih znamenaka u ovisnosti o broju slojeva mreže i aktivacijskoj funkciji [21]

<b>AKTIVACIJSKA FUNKCIJA</b>	<b>DVOSLOJNI MLP</b>	<b>TROSLOJNI MLP</b>
sigmoid	90.3%	95.6%
middle D-ReLU	89.6%	92.7%
mean D-ReLU	89.6%	96.0%

Tablica 3.3 - Vrijeme izvođenja klasifikacije znamenke i frekvencija rada sustava u ovisnosti o postavkama mreže [21]

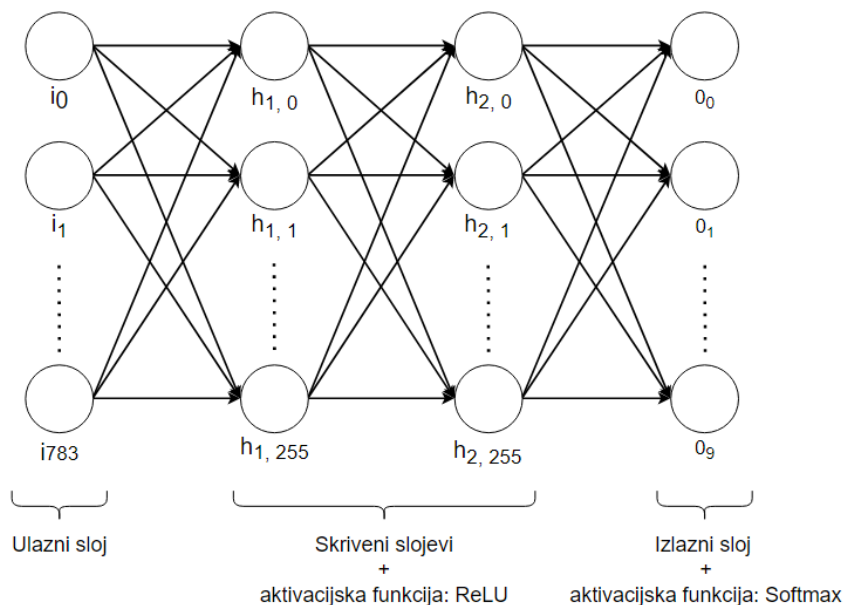
<b>AKTIVACIJSKA FUNKCIJA</b>	<b>DVOSLOJNA MREŽA</b>		<b>TROSLOJNA MREŽA</b>	
	<b>FREKVENCIJA RADA SUSTAVA</b>	<b>VRIJEME IZVOĐENJA KLASIFIKACIJE</b>	<b>FREKVENCIJA RADA SUSTAVA</b>	<b>VRIJEME IZVOĐENJA KLASIFIKACIJE</b>
sigmoid	250 MHz	0.02 s	60 MHz	0.15 s
D-ReLU	250 MHz	0.02 s	60 MHz	0.15 s

## 4. MODEL I TRENIRANJE NEURONSKE MREŽE

U ovome poglavlju opisan je model neuronske mreže implementirane u ovom radu. Također su detaljno opisani algoritmi propagacije unaprijed i unazad pomoću kojih se redom ostvaruju klasifikacija znamenki i treniranje mreže. Teorija dana u ovom poglavlju pruža osnovu za dizajniranje digitalnog sustava i implementaciju sustava na FPGA sustav koja je opisana u poglavlju 5.

### 4.1. Model neuronske mreže

U ovome radu, za rješavanje problema klasifikacije znamenaka iz MNIST baze, odabrana je umjetna neuronska mreža. Mrežu čine četiri potpuno povezana sloja: ulazni, dva skrivena i jedan izlazni sloj. Na ulazni sloj mreže dovodi se slika iz MNIST baze, razlučivosti 28x28, a iz toga slijedi da ulazni sloj sadrži 784 neurona. Svaki od skrivenih slojeva sadrži 256, a izlazni sloj 10 neurona. Model mreže, vidljiv na slici 4.1, je fleksibilan u smislu mogućnosti promjene broja neurona u slojevima i broja skrivenih slojeva.



Slika 4.1 - Model umjetne neuronske mreže

Aktivacijska funkcija korištena u skrivenim slojevima jest ReLU funkcija:

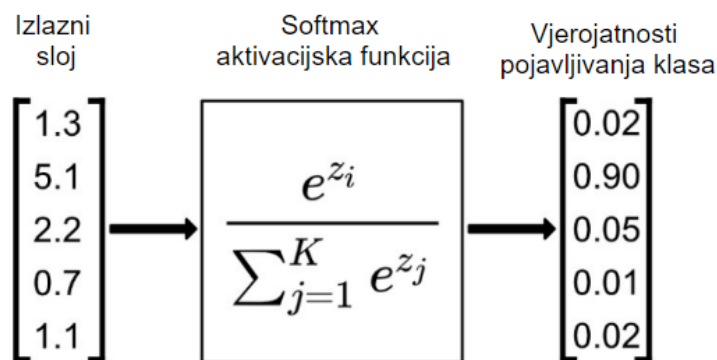
$$R(z) = \max(0, z), \quad (4-1)$$

gdje je  $z$  ulaz u aktivacijsku funkciju. Izlaz iz ReLU funkcije jednak je 0 ako je ulaz manji ili jednak 0, a inače je izlaz jednak ulazu. U ovome radu, ReLU je odabrana zbog vrlo jednostavne hardverske implementacije i jednostavnog treniranja mreže što pridonosi poboljšanju performansi i štedi resurse sustava.

Budući da je namjena sustava prepoznavanje znamenki od 0 do 9, izlazni sloj sadrži 10 neurona. Izlaz neurona u izlaznom sloju predstavlja vjerojatnost da je na slici prepoznata neka od znamenki. Npr. neuron  $O_1$  sa slike 4.1 predstavlja vjerojatnost pojavljivanja znamenke 1 na ulaznoj slici. Aktivacijska funkcija Softmax, obično korištena na izlaznom sloju za rješavanje problema klasifikacije, na izlazu daje vjerojatnost pojavljivanja određene klase. Zbog prethodno navedenog, u ovome modelu je na izlaznom sloju korištena funkcija Softmax:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}, \quad (4-2)$$

gdje je  $\sigma(\vec{z})_i$  vjerojatnost pojavljivanja  $i$ -te klase,  $\vec{z}$  ulazni vektor,  $e^{z_i}$  standardna eksponencijalna funkcija  $i$ -tog elementa ulaznog vektora,  $K$  broj klasa, a  $\sum_{j=1}^K e^{z_j}$  suma standardnih eksponencijalnih funkcija elemenata ulaznog vektora. Zbroj svih vjerojatnosti pojavljivanja klasa iznosi 1, što se može vidjeti i na primjeru izvođenja Softmax aktivacijske funkcije na slici 4.2.



Slika 4.2 - Primjer izvršavanja Softmax aktivacijske funkcije [22]

## 4.2. Propagacija unaprijed

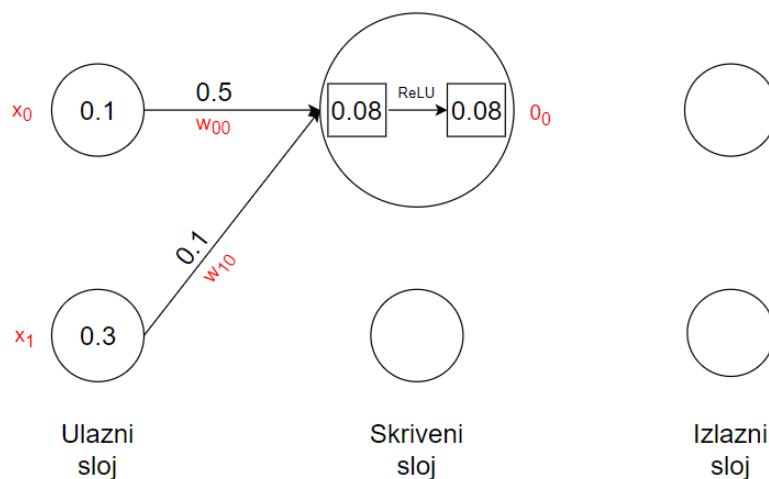
Neuronske mreže implementiraju algoritam propagacije unaprijed za izračunavanje izlaza iz mreže na temelju ulaza u istu. Ovo potpoglavlje daje teorijsku osnovu za implementaciju prethodno spomenutog algoritma u umjetnu neuronsku mrežu i primjer izvođenja algoritma na umjetnoj neuronskoj mreži malih dimenzija.

Ulazni sloj mreže služi samo za prihvaćanje ulaznih podataka, tj. u njemu se ne izvode izračuni, ali format ulaznih podataka može znatno utjecati na rezultate dobivene treniranjem. Preporučuje se da ulazni podaci budu iz skupa  $[0, 1]$  ili da prosjek ulaznih podataka bude 0, a standardna devijacija približno 1. Normalizacija i skaliranje ulaznih podataka često pomažu u izbjegavanju nepovoljnih lokalnih minimuma tijekom treniranja te ubrzavaju treniranje mreže [23].

Počevši od prvog skrivenog sloja, izlazi neurona računaju se prema sljedećem izrazu:

$$O_i = act(\sum_{j=0}^N x_j * w_{ji}), \quad (4-3)$$

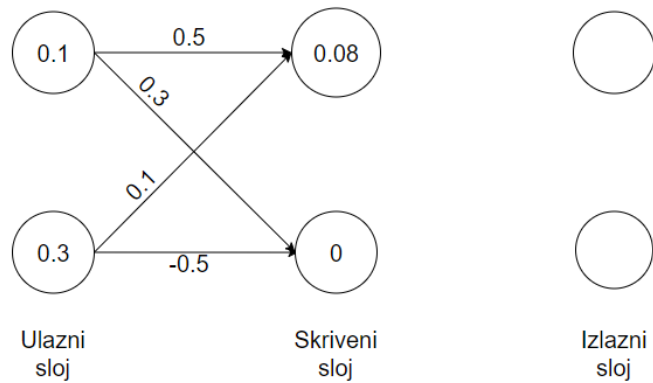
gdje je  $O_i$  izlaz iz  $i$ -tog neurona u trenutno promatranom sloju,  $N$  broj neurona u prethodnom sloju,  $x_j$  izlaz  $j$ -tog neurona iz prethodnog sloja,  $w_{ji}$  težina koja spaja  $i$ -ti neuron iz trenutno promatranog sloja s  $j$ -tim neuronom iz prethodnog sloja,  $act$  aktivacijska funkcija. Slika 4.3 prikazuje primjenu izraza (4-3) za računanje izlaza iz prvog neurona u skrivenom sloju. Pritom je korištena aktivacijska funkcija ReLU (4-1).



Slika 4.3 - Izračun izlaza prvog neurona u skrivenom sloju; aktivacijska funkcija: ReLU

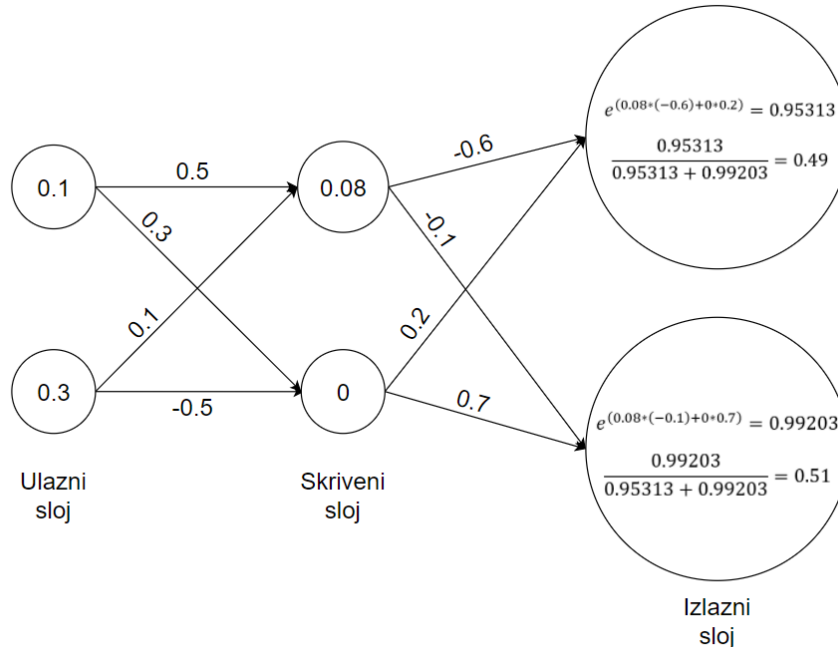
Istim pristupom izračunavaju se izlazi svih ostalih neurona u skrivenim slojevima. Rezultati su vidljivi na slici 4.4.





Slika 4.4 - Izračun izlaza svih neurona u skrivenom sloju; aktivacijska funkcija: ReLU

Izlazi neurona u izlaznom sloju također se računaju pomoću izraza (4-3), ali se u ovom slučaju koristi aktivacijska funkcija Softmax (4-2) kako bi izlaz mreže konačno dao vjerojatnosti pojavljivanja pojedinih klasa na ulazu. Slika 4.5 prikazuje primjenu izraza (4-2) i (4-3) za izračun izlaza neurona u izlaznom sloju. Na izlazu mreže može se primijetiti kako je vjerojatnost pojavljivanja prve klase 49%, a druge klase 51%, što dovodi do zaključka da se na ulazu u mrežu nalazi objekt druge klase.

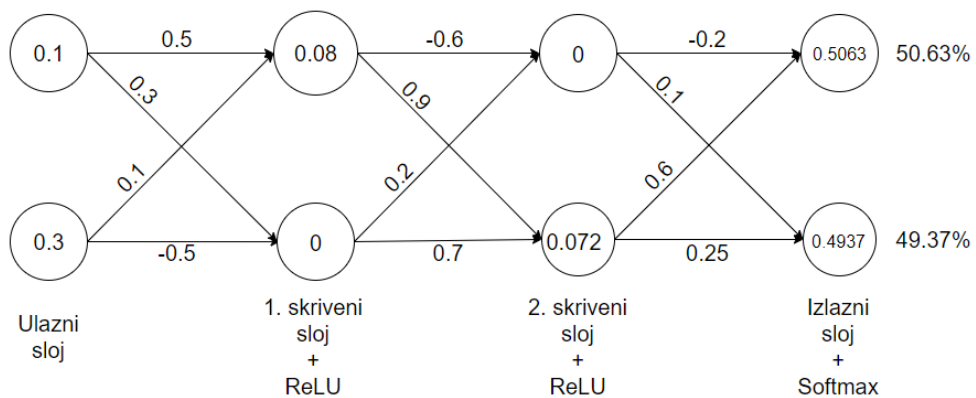


Slika 4.5 - Izračun izlaza svih neurona u izlaznom sloju; aktivacijska funkcija: Softmax

### 4.3. Propagacija unazad

Neuronske mreže implementiraju algoritam propagacije unazad s ciljem treniranja parametara mreže (npr. težine). Algoritam koristi tzv. pravilo lanca (engl. *chain rule*) za računanje gradijenata troška na svakom dijelu mreže te se na temelju njih podešavaju prethodno spomenuti parametri [24]. Ovo potpoglavlje daje teorijsku osnovu za implementaciju algoritma u umjetnu neuronsku mrežu i primjer izvođenja algoritma u umjetnoj neuronskoj mreži malih dimenzija.

Slika 4.6 prikazuje 4-slojnu umjetnu neuronsku mrežu na kojoj je prethodno urađen algoritam propagacije unaprijed te koja u ovom potpoglavlju služi kao predložak za objašnjavanje algoritma propagacije unazad. Prije ulaska u detalje algoritma, pretpostavimo da se na ulazu mreže nalazi objekt prve klase čija je vjerojatnost pojavljivanja 50.63%. Željene vjerojatnosti pojavljivanja su 100% za prvu klasu i 0% za drugu klasu.



Slika 4.6 - 4-slojna umjetna neuronska mreža s prethodno urađenim algoritmom propagacije unaprijed

Prije početka treniranja mreže, potrebno je znati pogrešku prepoznavanja. Pogreška prepoznavanja (trošak) se računa pomoću funkcije troška (engl. *Loss function*). Nadalje, trošak se koristi kod računanja gradijenata troška koji se zatim koriste za podešavanje parametara mreže. Neke od najčešće korištenih funkcija troška su: srednja kvadratna pogreška (engl. *Mean Squared Error*), binarna unakrsna entropija (engl. *Binary Cross-Entropy*), kategorička unakrsna entropija (engl. *Categorical Cross-Entropy*) i rijetka kategorička unakrsna entropija (engl. *Sparse Categorical Cross-Entropy*) [25]. Izraze ovih funkcija moguće je vidjeti u prilogu P.2. Odabir funkcije troška obično ovisi o formatu podataka na izlazu iz mreže pa je u ovome radu, zbog Softmax aktivacijske funkcije na izlaznom sloju, odabrana funkcija: kategorička unakrsna entropija.

Kategorička unakrsna entropija:

$$C(y, \hat{y}) = - \sum_{i=0}^{n-1} y_i \log(\hat{y}_i), \quad (4-4)$$

gdje je  $y_i$  očekivana vjerojatnost pojavljivanja  $i$ -te klase, a  $\hat{y}_i$  stvarna (estimirana) vjerojatnost pojavljivanja klase, obično se odabire za funkciju troška kada neuronska mreža rješava problem više-klasne klasifikacije. Izlazni sloj mreže u tom slučaju sadrži točno onoliko neurona koliko ima klasa. Također, podaci na izlazu iz mreže (vjerojatnosti pojavljivanja klasa) trebaju biti iz skupa  $[0,1]$ .

Izraz:

$$C = -(1 * \log(0.5063) + 0 * \log(0.4937)) = 0.29559 \quad (4-5)$$

prikazuje primjenu izraza (4-4) na sliku 4.6. Rezultat je trošak dobiven funkcijom kategoričke unakrsne entropije. Treniranjem se trošak smanjuje, a trošak maksimalno istrenirane mreže iznosi 0.

Pri podešavanju težina u neuronskoj mreži potrebno je znati kako pojedina težina utječe na ukupni trošak mreže. Gradijent pogreške koju čini pojedina težina računa se kao parcijalna derivacija ukupnog troška  $C$  po težini  $w$   $\frac{\partial C}{\partial w}$ .

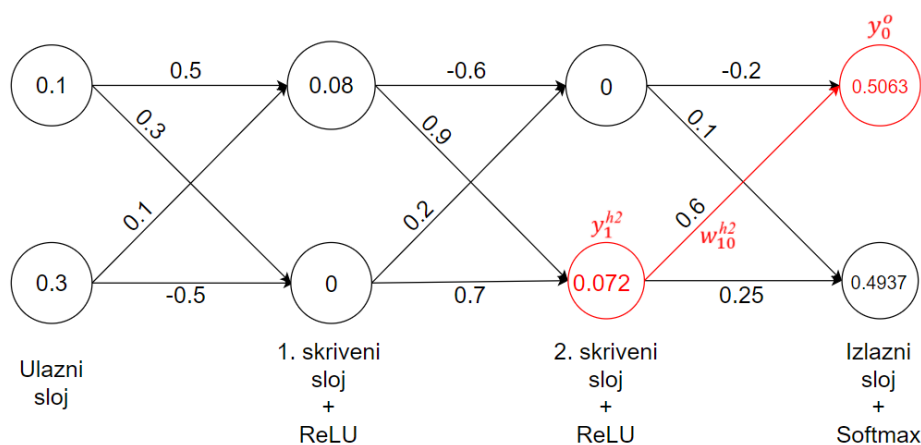
Budući da algoritam propagacije unazad prvo utječe na težine koje povezuju zadnji skriveni sloj s izlaznim slojem, kreće se od parcijalne derivacije ukupnog troška po težini iz prethodno spomenutog sloja. Izraz:

$$\frac{\partial C}{\partial w_{ni}^{h2}} = (\hat{y}_i^o - y_i^o) * y_n^{h2}, \quad (4-6)$$

gdje je  $\frac{\partial C}{\partial w_{ni}^{h2}}$  gradijent troška na težini koja povezuje  $n$ -ti neuron iz drugog skrivenog sloja sa  $i$ -tim neuronom iz izlaznog sloja,  $\hat{y}_i^o$  stvarna (estimirana) vjerojatnost pojavljivanja  $(i+1)$ -te klase (izlaz iz  $i$ -tog neurona u izlaznom sloju),  $y_i^o$  očekivana vjerojatnost pojavljivanja  $(i+1)$ -te klase, a  $y_n^{h2}$  izlaz  $n$ -tog neurona iz drugog skrivenog sloja. Ovaj izraz je dobiven pravilom lanca u kojemu su u obzir uzeti kategorička unakrsna entropija i izraz (4-3) gdje je za aktivacijsku funkciju u izlaznom sloju uzet Softmax. Izraz:

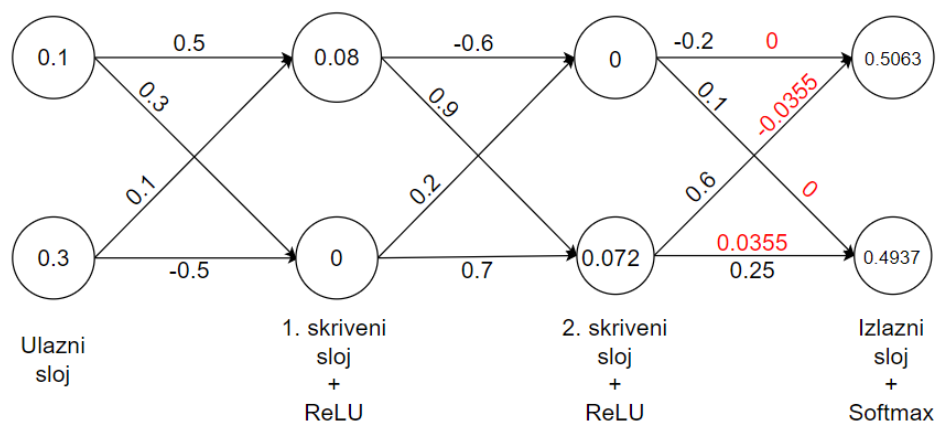
$$\frac{\partial \mathcal{C}}{\partial w_{10}^{h2}} = (0.5063 - 1) * 0.072 = -0.0355 \quad (4-7)$$

prikazuje izračun gradijenta troška težine koja povezuje crveno označene neurone sa slike 4.7.  $y_i^o$  u ovom slučaju iznosi 1 zbog pretpostavke da se na ulazu u mrežu nalazi objekt prve klase (0-ti neuron u izlaznom sloju).



Slika 4.7 - Izračun gradijenta troška težine koja povezuje drugi skriveni sloj i izlazni sloj

Izraz (4-6) je potrebno primijeniti na sve težine koje povezuju drugi skriveni sloj s izlaznim slojem mreže. Na slici 4.8, crveno su označeni izračunati gradijenti troška prethodno spomenutih težina.

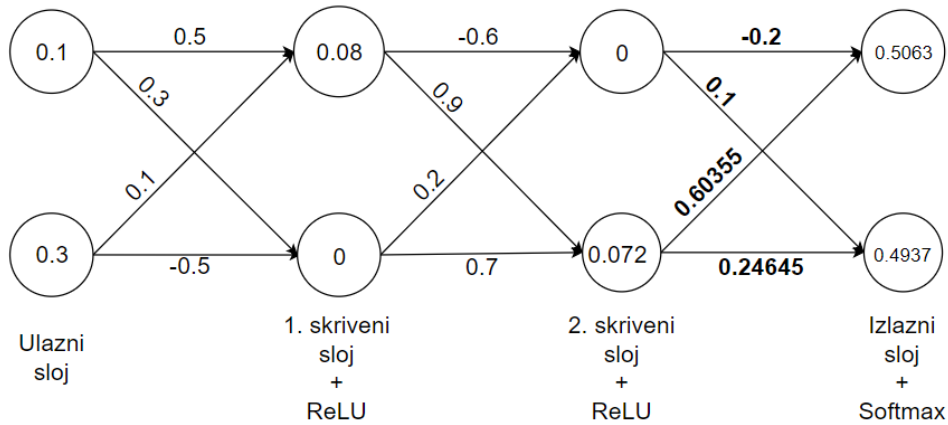


Slika 4.8 - Gradijenti troška težina koje povezuju drugi skriveni sloj i izlazni sloj

Sljedeći korak je podešavanje težina s obzirom na izračunate gradijente troška. Izraz:

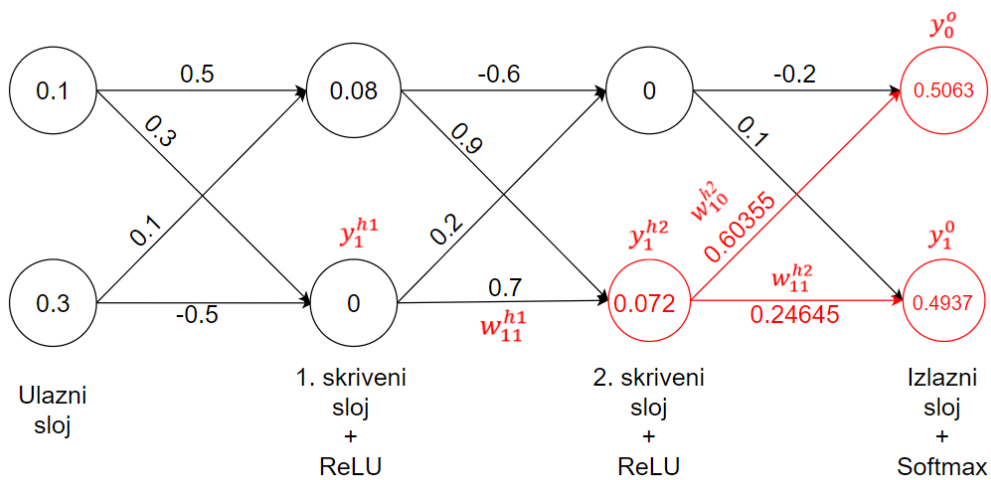
$$w_n = w_o - \left( \alpha * \frac{\partial C}{\partial w_o} \right), \quad (4-8)$$

gdje je  $w_o$  težina čija se vrijednost podešava,  $w_n$  podešena težina,  $\alpha$  koeficijent učenja (određuje brzinu treniranja mreže), a  $\frac{\partial C}{\partial w}$  gradijent troška koji uzrokuje težina, prikazuje način podešavanja težine s obzirom na gradijent troška koji ona uzrokuje. Primjenom izraza (4-8) na postojeće težine sa slike 4.8, dobivaju se novi iznosi težina prikazani na slici 4.9. Koeficijent učenja iznosi 0,1.



Slika 4.9 - Podešene težine koje povezuju drugi skriveni sloj i izlazni sloj

Podešavanje težina koje povezuju prvi i drugi skriveni sloj nešto je složeniji postupak. Izraz za gradijent troška prethodno spomenute težine također se dobiva pravilom lanca, a lanac obuhvaća sve puteve od date težine do svih izlaza iz mreže. Na slici 4.10, crveno su označeni svi putevi od težine  $w_{11}^{h1}$  do svih izlaza iz mreže.



Slika 4.10 - Putevi od težine  $w_{11}^{h1}$  do izlaza iz mreže

Pravilom lanca moguće je dobiti izraz za gradijent troška težine  $w_{11}^{h1}$ :

$$\frac{\partial C}{\partial w_{11}^{h1}} = \begin{cases} [(\hat{y}_0^o - y_0^o) * w_{10}^{h2} + (\hat{y}_1^o - y_1^o) * w_{11}^{h2}] * y_1^{h1}, & y_1^{h2} > 0 \\ 0, & y_1^{h2} \leq 0 \end{cases} \quad (4-9)$$

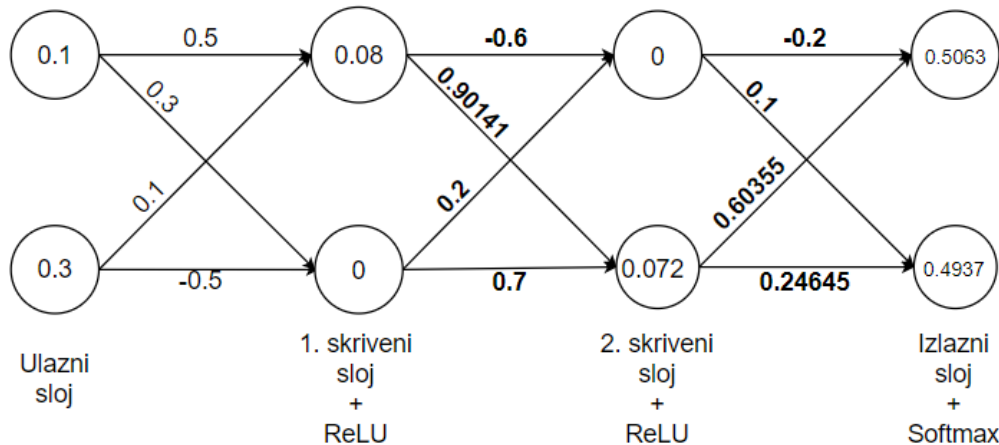
Povećanjem broja neurona u slojevima, ovaj izraz bi postao nepregledan pa se uvodi zamjena:

$$grad_{ij}^{h2} = (\hat{y}_j^o - y_j^o) * w_{ij}^{h2}. \quad (4-10)$$

Nadalje, uvođenjem zamjene (4-10) u izraz (4-9) i poopćavanjem izraza (4-9), dobije se opći izraz za izračunavanje gradijenta troška bilo koje težine koja spaja prvi skriveni s drugim skrivenim slojem mreže:

$$\frac{\partial C}{\partial w_{ij}^{h1}} = \begin{cases} \left( \sum_{k=0}^{n-1} grad_{jk}^{h2} \right) * y_i^{h1}, & y_j^{h2} > 0 \\ 0, & y_j^{h2} \leq 0 \end{cases}, \quad (4-11)$$

Primjenjivanjem izraza (4-11) i (4-8) na sliku 4.9, izračunavaju se nove vrijednosti prethodno spomenutih težina. Rezultati su prikazani na slici 4.11.

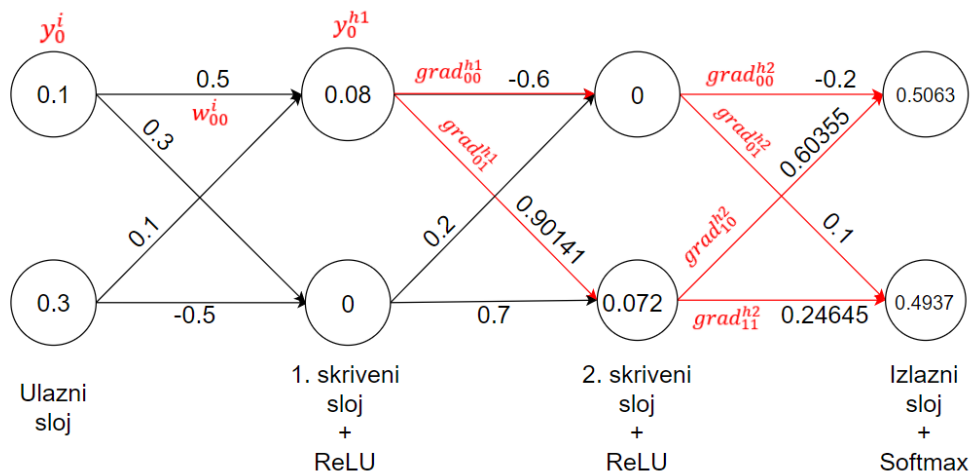


Slika 4.11 - Podešene težine koje spajaju prvi i drugi skriveni sloj

Izraz za gradijent troška težine koja povezuje ulazni i prvi skriveni sloj također se dobiva pravilom lanca, no zbog složenosti izraza odmah se uvodi zamjena po uzoru na (4-10) i (4-11):

$$grad_{ij}^{h1} = \begin{cases} \left( \sum_{k=0}^{n-1} grad_{jk}^{h2} \right) * w_{ij}^{h1}, & y_j^{h2} > 0 \\ 0, & y_j^{h2} \leq 0 \end{cases}, \quad (4-12)$$

gdje je  $n$  broj neurona u izlaznom sloju. Nadalje, pri izračunu gradijena troška prethodno spomenutih težina, u obzir se uzimaju prethodno izračunati gradijenti dani izrazima (4-10) i (4-12). Na slici 4.12 crveno su označeni parametri koje treba uzeti u obzir za izračun gradijenta troška težine  $w_{00}^i$ .

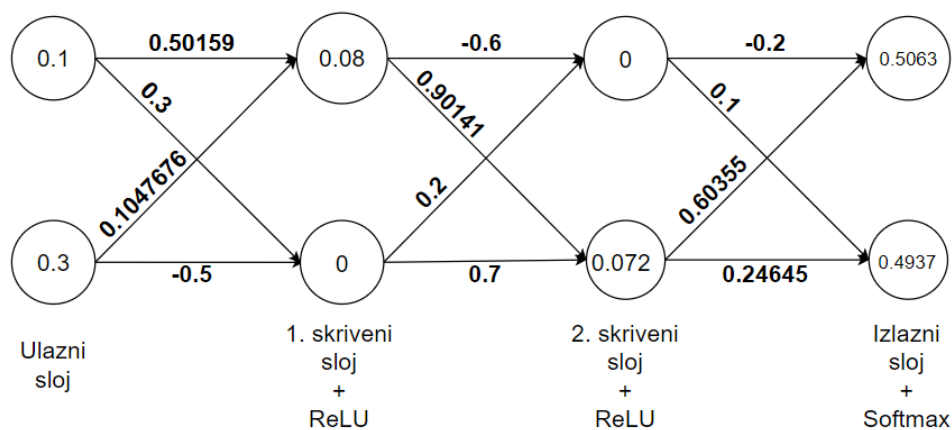


Slika 4.12 - Parametri za računanje gradijenta troška težine  $w_{00}^i$

Pomoću primjera na slici 4.12, analogno izrazu (4-11) može se izvesti opći izraz za izračun gradijenta troška težine koja spaja ulazni sloj i prvi skriveni sloj:

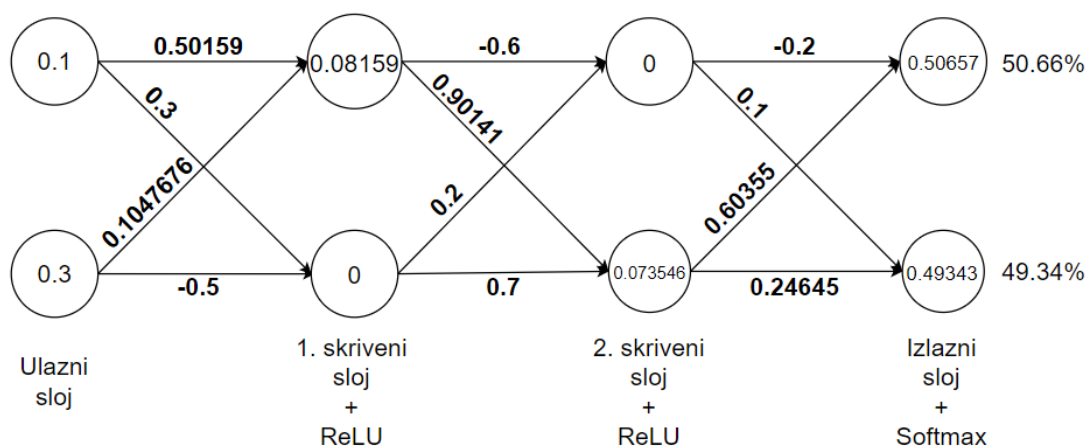
$$\frac{\partial C}{\partial w_{ij}^i} = \begin{cases} \left( \sum_{k=0}^{n-1} grad_{jk}^{h1} \right) * y_i, & y_j^{h1} > 0 \\ 0, & y_j^{h1} \leq 0 \end{cases}, \quad (4-13)$$

gdje je  $n$  broj neurona u drugom skrivenom sloju. Nadalje, korištenjem izraza (4-13) i (4-8), mogu se izračunati nove vrijednosti težina koje su prikazane na slici 4.13.



Slika 4.13 - Izračunate težine koje spajaju ulazni i prvi skriveni sloj

Na mreži s podešenim težinama je ponovno primijenjen algoritam propagacije unaprijed. Rezultati su prikazani na slici 4.14. Može se zaključiti da je rezultat nešto prihvatljiviji jer se vjerojatnost pojavljivanja prve klase povećava te bi se iterativnim treniranjem mreže dobivali sve bolji rezultati. Također se može eksperimentirati s koeficijentom učenja kako bi se ubrzalo treniranje, ali treba paziti na to da prevelika vrijednost koeficijenta može uzrokovati povećanje umjesto smanjenja ukupnog troška treniranja, a premala vrijednost koeficijenta može uzrokovati veoma sporo treniranje [26].



Slika 4.14 - Rezultati propagacije unaprijed nakon treniranja mreže



## 5. IMPLEMENTACIJA SUSTAVA NA RAZVOJNU PLOČU

U ovome poglavlju dani su svi detalji implementacije sustava za prepoznavanje znamenki na ZYBO razvojnu ploču. Također su opisani alati i tehnologije potrebni za sintezu i analizu digitalnih sustava te ZYBO razvojna ploča. Opis sustava je dan tako da su prvo opisani njegovi podsustavi, a zatim je sustav prikazan u cjelini. Na kraju poglavlja, nakon opisa sklopovlja, opisana je softverska podrška u vidu aplikacije kreirane za Zynq-7000 procesorski sustav.

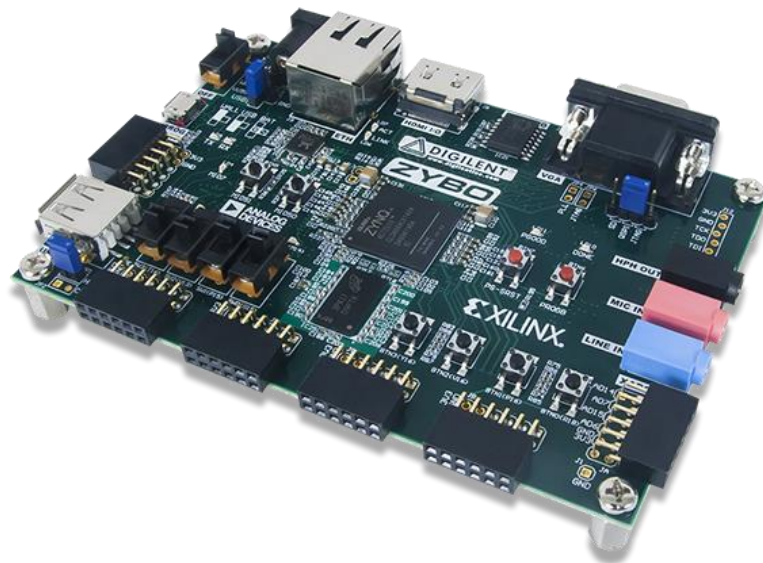
### 5.1. Korišteni alati i tehnologije

Sljedeća potpoglavljja opisuju alate i tehnologije korištene pri implementaciji sustava na razvojnu ploču, a to su: ZYBO razvojna ploča, FPGA i Vivado softverski paket za sintezu i analizu digitalnih sustava.

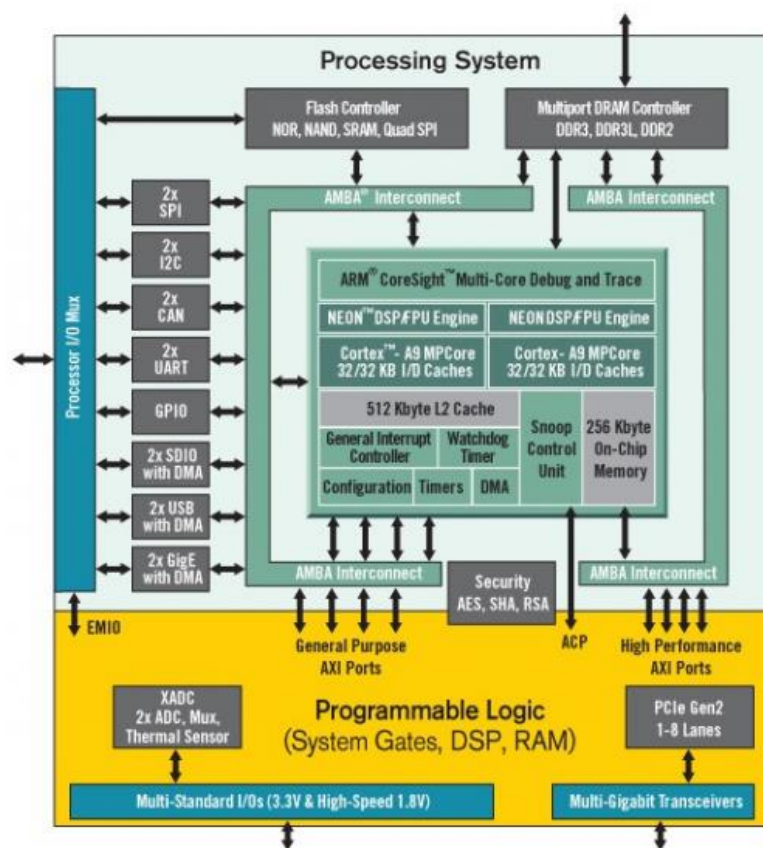
#### 5.1.1. ZYBO

ZYBO je ploča koju je proizveo Digilent za razvoj digitalnih krugova i softvera za ugradbene sustave, a temelji se na Xilinx-ovom Z-7010 sustavu na čipu te je vidljiva na slici 5.1. Z-7010 je najmanji član Xilinx-ove Zynq-7000 porodice te integrira ARM Cortex-A9 procesor i FPGA logiku serije Xilinx-7, a arhitektura Z-7010 vidljiva je na slici 5.2. Neke od glavnih značajki koje ploča sadrži su:

- DDR3 (engl. *Double Data Rate Random Access Memory*) memorija – 512 MB,
- SSM2603 – audio kodek s pripadajućim ulaznim i izlaznim priključnicama,
- jedan HDMI priključak – ulaz/izlaz,
- USB s dvije uloge: „uređaj“ i „domaćin“,
- Ethernet s pripadajućim RJ-45 priključkom,
- utor za SD karticu,
- VGA,
- 4 sklopke i 4 tipkala – ulazi spojeni na FPGA,
- 4 LED diode – izlazi spojeni na FPGA,
- 6 Pmod priključaka – ulazi/izlazi spojeni na FPGA,
- 2 tipkala – ulazi spojeni na procesor [27].



Slika 5.1 - ZYBO razvojna ploča [27]



Slika 5.2 - Arhitektura Z-7010 sustava na čipu [27]

Prethodno navedene značajke omogućuju da se cijeli softverski i sklopovski sustav implementira na ploču bez dodatnog sklopovlja. Sinteza i analiza digitalnih sklopova na ZYBO

ploči je omogućena kroz Vivado softverski paket, a kreiranje aplikacija za ARM Cortex-A9 procesor je omogućeno kroz Xilinx SDK razvojno okruženje. Također, važno je naglasiti da FPGA unutar Z-7010 sustava na čipu sadrži 4400 logičkih ćelija, gdje svaku od njih čini četiri LUT-a (engl. *Lookup Table*) sa šest ulaza te osam registara. FPGA sadrži i 240 KB brze blokovske RAM memorije, dva podsustava za upravljanje signalom takta od kojih svaki sadrži PLL (engl. *Phase-Locked Loop*) i MMCM (engl. *Mixed-Mode Clock Manager*) te 80 DSP (engl. *Digital Signal Processing*) blokova. Detaljniji prikaz resursa FPGA sustava može se vidjeti u tablici 5.1. Ploča se za potrebe programiranja i otklanjanja grešaka spaja na računalo preko USB micro-B priključka, a korišteni protokol je JTAG (engl. *Joint Test Action Group*) [27].

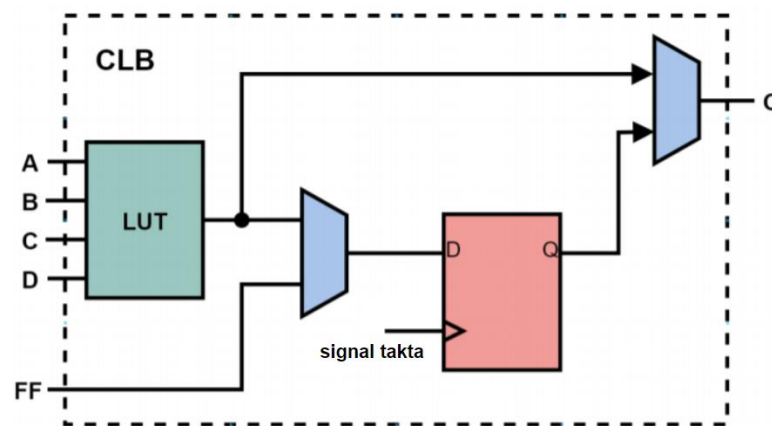
Tablica 5.1 - Dostupni resursi FPGA [27]

TIP BLOKA	DOSTUPNA KOLIČINA BLOKOVA
DSP	80
FF	35200
LUT	17600
BRAM	240 KB

### 5.1.2. FPGA

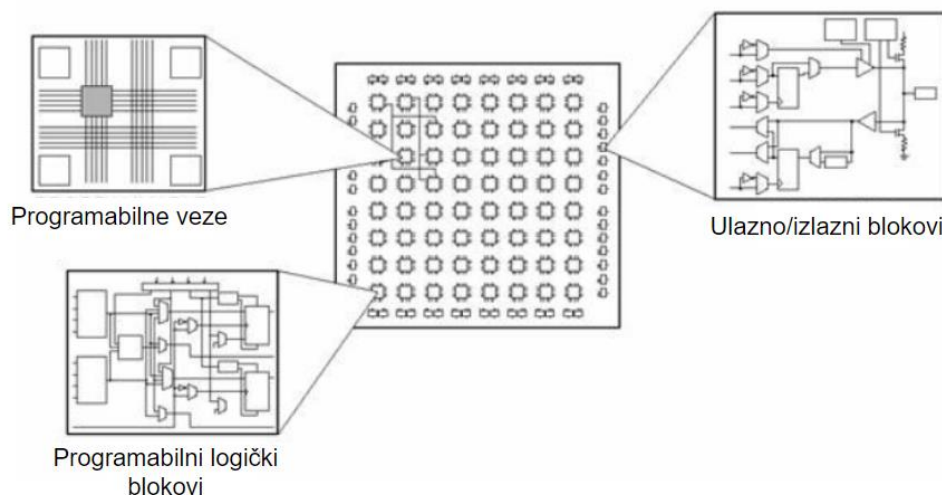
1960-ih i 1970-ih godina proizvodili su se integrirani krugovi koji su sadržavali logička vrata temeljena na TTL (engl. *Transistor-Transistor Logic*) tehnologiji. Takvi integrirani krugovi su sadržavali i do 100000 tranzistora ranih 1980-ih pa su prema tome sadržavali na tisuće logičkih vrata. Takvi su se integrirani krugovi najčešće koristili za izvođenje raznih logičkih funkcija kao što su multipleksiranje, dekodiranje, uspoređivanje, zbrajanje i slično. Zatim su se pojavili programabilni logički sklopovi (engl. *Programmable Logic Devices - PLDs*) koji su integrirali polja logičkih sklopova. Takvi sklopovi mogli su se programirati pomoću računalno potpomognutih alata (engl. *computer-aided tools - CAD*). Spajanjem više PLD-ova u jednu cjelinu nastali su kompleksni programabilni logički sklopovi (engl. *Complex Programmable Logic Devices - CPLDs*). Konačno, sredinom 1980-ih godina predstavljena je potpuno različita arhitektura, tj. FPGA, koja koristi tzv. „lookup“ tablice umjesto I i ILI logičkih vrata za implementaciju kombinacijske logike. Takvi sklopovi se sastoje od konfigurabilnih logičkih blokova (engl. *Configurable Logic Blocks - CLB*) okruženih ulazno/izlaznim blokovima [28].

CLB-ovi najčešće sadrže: „lookup“ tablice, malu količinu RAM memorije, registre, digitalni upravljač signalom takta (engl. *Digital Clock Manager - DCM*) itd. Primjer jednostavnog CLB-a prikazan je na slici 5.3. Svrha „lookup“ tablica je implementacija kombinacijske logike, registara – spremanje male količine podataka, RAM memorije – spremanje velike količine podataka i DCM-a – eliminiranje vremenskih kašnjenja kod distribucije signala takta [28].



Slika 5.3 - Blokovski dijagram jednostavnog CLB-a [29]

Matrična struktura CLB-ova u FPGA sustavu, vidljiva na slici 5.4, omogućuje alatima za sintezu i analizu da smjeste određene module dizajniranog sustava tako da se postigne što manja vremenska propagacija signala, što za cilj ima povećanje frekvencije rada sustava.



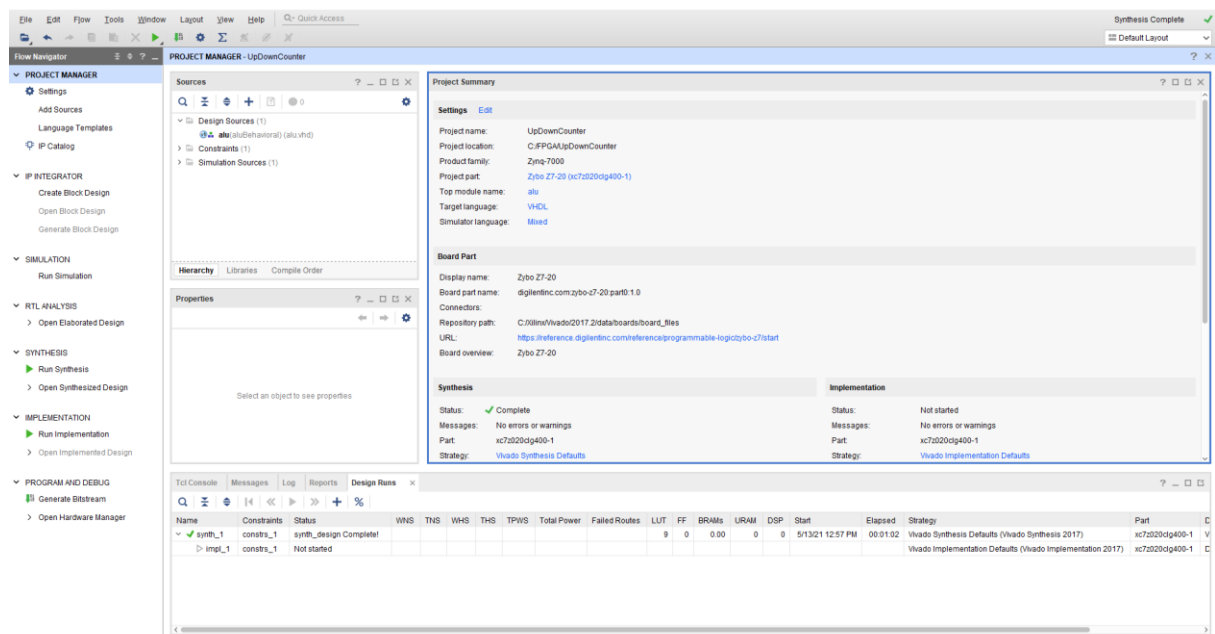
Slika 5.4 - Struktura FPGA [30]

FPGA danas nalazi široku primjenu u sustavima za obradu slike i zvuka, automobilskoj, zrakoplovnoj, vojnoj industriji itd. Može se reći da je glavna primjena FPGA izrada prototipa za

ASIC i procesore jer ga je moguće reprogramirati novim verzijama sustava sve dok verzija nije konačna. FPGA se često može pronaći i u gotovim proizvodima koji se ne proizvode u velikoj količini pa proizvodnja ASIC-a najčešće nije isplativa [31].

### 5.1.3. Vivado

Vivado je softverski paket kojega je 2012. godine proizveo Xilinx za sintezu i analizu digitalnih sustava. Zamijenio je Xilinx ISE te su dodane značajke za razvoj na sustavima na čipu i sintezu na visokoj razini (engl. *High-level synthesis*). Kao i prethodnik ISE, Vivado uključuje logički simulator ISIM. Slika 5.5 prikazuje glavni prozor Vivado softverskog paketa. Najvažniji dio paketa je izbornik s lijeve strane „Flow Navigator“ pomoću kojega se lako može doći do svih glavnih komponenata paketa [32].



Slika 5.5 - Glavni prozor Vivado softverskog paketa

Glavne komponente Vivado-a su:

- „Vivado High-Level Synthesis“ – prevoditelj koji omogućuje opisivanje hardvera u C, C++ i SystemC programskim jezicima,
- „Vivado Simulator“ – simulator koji podržava više-jezično simuliranje, TCL skripte i kriptirane IP jezgre (engl. *Intellectual Property core*),
- „Vivado IP Integrator“ – komponenta koja omogućava jednostavnu integraciju i konfiguriranje IP jezgri te kreiranje blokovskih dijagrama sustava,

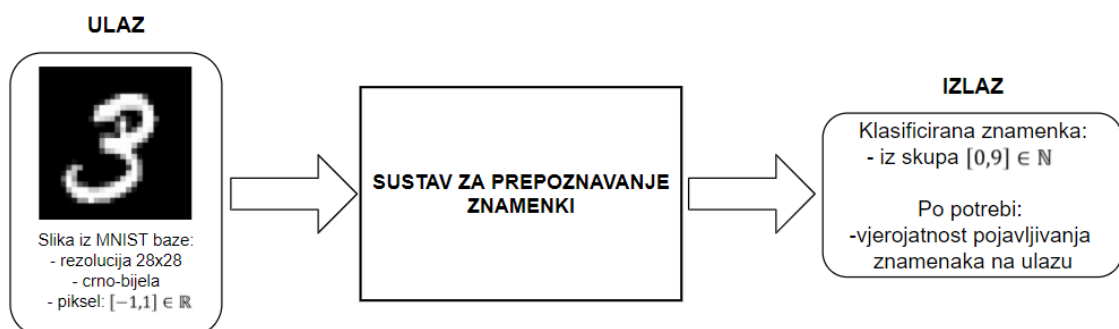
- „Vivado TCL Store“ – sustav skriptiranja koji služi za razvoj dodataka za Vivado, a može služiti i za modifikaciju mogućnosti koje on pruža [32].

Vivado pruža podršku jedino za Xilinx-ove FPGA sustave serije 7 i novije sustave kao što su UltraScale i UltraScale+. Za razvoj digitalnih krugova na starijim sustavima, potrebno je koristiti ISE softverski paket [32].

## 5.2. Ulaz i izlaz sustava

Ulaz i izlaz sustava za prepoznavanje znamenki redom predstavljaju slika s koje treba prepoznati i klasificirati znamenku te rezultat klasifikacije, tj. znamenka koja se najvjerojatnije nalazi na slici. U ovome slučaju, ulazna slika je rezolucije 28x28 te je crno-bijela. Pikseli slika iz MNIST baze zadani su vrijednostima iz skupa  $[0,255] \in \mathbb{N}$ , gdje 0 predstavlja bijelu boju, a 255 crnu boju. Iz prethodno navedenog može se zaključiti da vrijednosti iz skupa  $[1,254] \in \mathbb{N}$  predstavljaju nijanse sive boje. Zbog velikog broja operacija množenja i zbrajanja u neuronskoj mreži koje mogu rezultirati vrlo velikim brojevima, skup ulaznih vrijednosti je mapiran na  $[-1,1] \in \mathbb{R}$ . Tip podatka, kojima je predstavljen piksel, jest 32-bitni s pomičnim zarezom.

Znamenke koje sadrži dekadski sustav su iz skupa  $[0,9] \in \mathbb{N}$  pa iz toga slijedi da se na izlazu sustava može pojaviti samo jedna od tih znamenki. Budući da rezultat klasifikacije nikad nije u potpunosti točan, tj. sustav za rezultat daje onu znamenku čija je vjerojatnost pojavljivanja najveća, po potrebi se uz rezultat klasifikacije može prikazati i vjerojatnost pojavljivanja znamenaka na slici. Slika 5.6 prikazuje blokovski dijagram koji ističe ulaz i izlaz sustava za prepoznavanje znamenki.

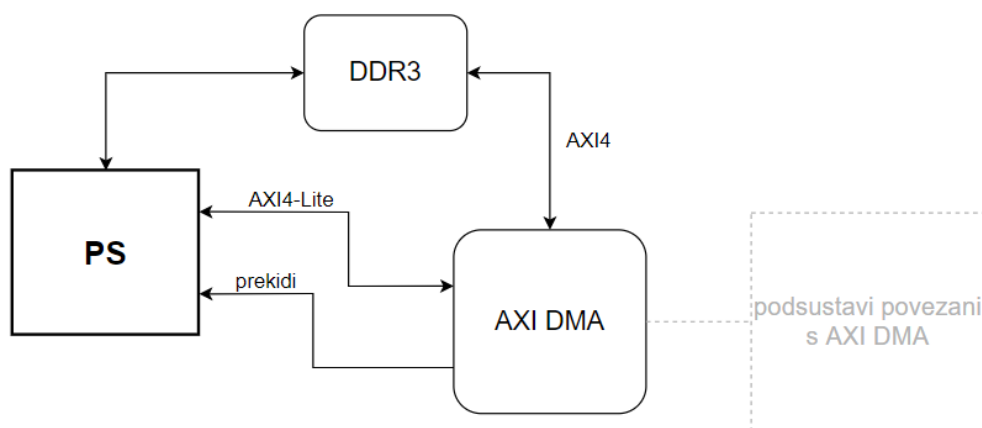


Slika 5.6 - Blokovski dijagram ulaza i izlaza sustava

### 5.3. Komunikacija PS-PL

Komunikacija između PS-a (engl. *Processing System*) i PL-a (engl. *Programmable Logic*) ostvarena je upisivanjem i čitanjem podataka iz registara u PL-u. Sučelje PS-a prema PL-u, u ovome slučaju, je ostvareno kroz AXI4-Lite protokol. AXI4-Lite je jednostavnija izvedba AXI4 (engl. *Advanced Extensible Interface 4*) protokola koja omogućuje komunikaciju između procesora i nekog modula u PL-u koji implementira AXI4-Lite sučelje. Općenito se koristi kada se procesorom želi zapisati ili pročitati podatak iz nekog registra u PL-u. Također, AXI4-Lite sučelje sadrži i adresne signale koji omogućuju pristup bilo kojem registru u PL-u (registri su adresirani), korištenjem samo jednog AXI4-sučelja na PS-u. Glavna razlika između AXI4 i AXI4-Lite protokola je u tome što AXI4-Lite ne podržava prijenos niza podataka u jednoj transakciji, već samo jedan 32-bitni ili 64-bitni podatak po transakciji [33].

U ovome radu, upravljanje AXI DMA (engl. *Direct Memory Access*) IP jezgrom ostvareno je AXI4-Lite protokolom. Protokolom se omogućuje komunikacija s internim registrima IP jezgre kao što su: upravljački registri, statusni registri, registri za pohranu izvorne i odredišne adrese itd. Prethodno navedeno omogućuje procesoru inicijalizaciju DMA transfera uz sve potrebne parametre [34]. Slika 5.7 prikazuje blokovsku shemu povezivanja PS-a i AXI DMA IP jezgre. Sa slike je moguće zaključiti da AXI DMA ima vezu sa PS-om jedino preko AXI4-Lite sučelja i signala prekida, a preko AXI4 čita ili piše podatke u DDR3 memoriju, neovisno o PS-u.

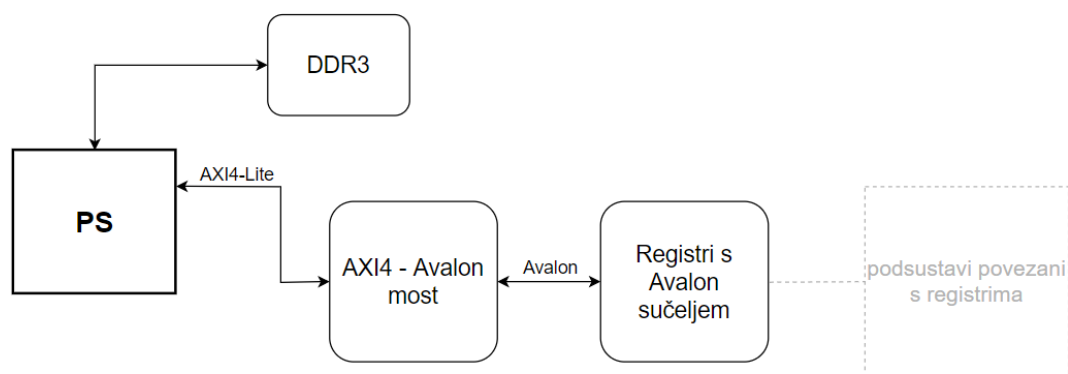


Slika 5.7 - Blokovska shema povezivanja AXI DMA IP jezgre i PS-a

Signali prekida, vidljivi na slici 5.7, također omogućuju komunikaciju između PS-a i PL-a, no ne postoji uvijek jednak broj signala prekida. Npr. ako je AXI DMA postavljen tako da je uključen samo kanal za čitanje, postoji samo signal prekida koji obavještava procesor da je čitanje podataka

završeno. Ako su uključeni i kanal za pisanje i kanal za čitanje, postoji signal prekida za oba kanala.

Osim internih registara AXI DMA IP jezgre, u sustavu za prepoznavanje znamenki postoje registri pomoću kojih se upravlja cijelim sustavom s naglaskom na blok za propagaciju unaprijed i blok za propagaciju unazad. Pristup tim registrima omogućen je Avalon protokolom [35]. Avalon ima jednaku zadaću kao i AXI4-Lite protokol, ali je implementacija Avalon sučelja nešto jednostavnija i zauzima manje resursa FPGA sustava. Slika 5.8 prikazuje blokovsku shemu povezivanja PS-a i registara s Avalon sučeljem.



Slika 5.8 - Blokowska shema povezivanja PS-a i registara s Avalon sučeljem

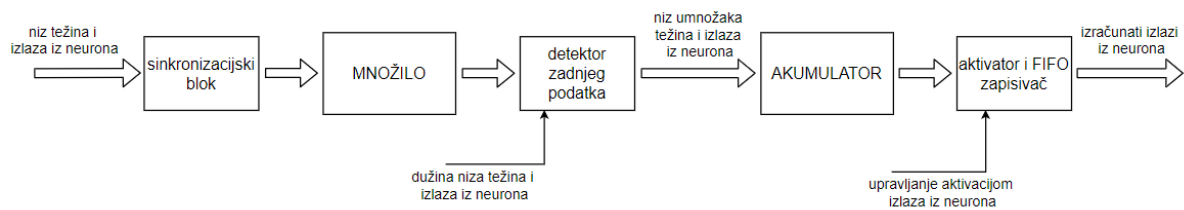
Budući da PS prema PL-u pruža AXI4 sučelje, a ne Avalon, u sustav je ugrađena IP jezgra vidljiva na slici 5.8 pod nazivom „AXI4 – Avalon most“ koja izvršava prilagodbu između ta dva protokola. Ta IP jezgra je preuzeta iz Xilinx-ovog IP repozitorija [36].

#### 5.4. Blok za propagaciju unaprijed

Blok za propagaciju unaprijed implementira funkcionalnosti za izvršavanje algoritma propagacije unaprijed. Njegova osnovna zadaća je iz dvije FIFO memorije, koje spremaju težine i izlaze iz neurona, čitati podatke te na temelju njih računati izlaze iz neurona u sljedećem sloju. Slika 5.9 prikazuje blokovsku shemu bloka za propagaciju unaprijed. Komunikacija između modula je ostvarena AXI4-Stream protokolom [37]. *Sinkronizacijski blok* prosljeđuje podatke prema *množilu* ako su podaci na izlazima obje FIFO memorije validni, tj. niti jedna memorija nije prazna. Zatim se umnožak težine i izlaza iz neurona dovodi na ulaz *detektora zadnjeg podatka* koji prema zadanom vektoru *dužina niza težina i izlaza iz neurona* obavještava *akumulator* da dolazi do prijenosa zadnjeg podatka u nizu. *Akumulator* akumulira umnoške izračunate u *množilu* zaključno sa zadnjim podatkom u nizu. Validni rezultat akumulacije se u modulu *aktivator i FIFO*



*zapisivač* aktivira (ReLU) ako je aktivan signal *upravljanje aktivacijom izlaza iz neurona* te sprema u FIFO i zatim preko DMA u DDR3 memoriju.



Slika 5.9 - Blokovska shema bloka za propagaciju unaprijed

### 5.4.1. Sinkronizacijski blok

*Sinkronizacijski blok* s pripadajućim ulazima i izlazima je prikazan na slici 5.10.

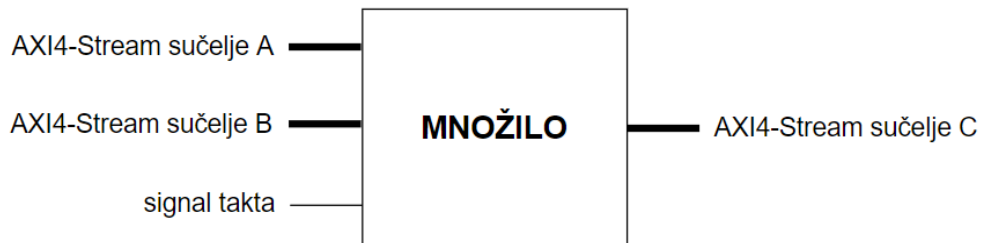


Slika 5.10 – Sinkronizacijski blok s pripadajućim ulazima i izlazima

Zadaća ovog modula je prosljeđivanje podataka sa *AXI4-Stream sučelja A* na *AXI4-Stream sučelje C* i sa *AXI4-Stream sučelja B* na *AXI4-Stream sučelje D*. Težine iz DDR3 memorije se preko DMA upisuju u FIFO iz kojega *sinkronizacijski blok* čita podatke. Isto vrijedi i za izlaze iz neurona. Uvjet prosljeđivanja podataka s ulaza na izlaz *sinkronizacijskog bloka* je da su oba podatka na sučeljima *A* i *B* validni, iz čega slijedi da niti jedan FIFO nije prazan. Također, važno je istaknuti da je upravo zbog postavki *množila* i *akumulatora* sa slike 5.9, koje omogućuju ne-blokirajući način rada AXI4-Stream protokola, kreiran *sinkronizacijski blok*. Ne-blokirajući način rada AXI4-Stream protokola onemogućuje dvostrano rukovanje (engl. *Two-way handshaking*). Iz prethodno navedenog slijedi da ne-blokirajući način rada AXI4-Stream protokola implementira jednostrano rukovanje (engl. *One-way handshaking*), tj. modul pošiljatelj javlja primatelju da je podatak validan i bez povratne informacije o spremnosti primanja podatka, šalje podatak primatelju.

### 5.4.2. Množilo

Množilo s pripadajućim ulazima i izlazima je prikazano na slici 5.11. IP jezgra množila je preuzeta iz Xilinx-ovog IP repozitorija [38].



Slika 5.11 – Množilo s pripadajućim ulazima i izlazima

Zadaća ovog modula je vratiti rezultat množenja dva 32-bitna operanda predstavljena aritmetikom pomičnog zareza. Parametri modula (IP jezgre „Floating point“) su prikazani tablicom 5.2.

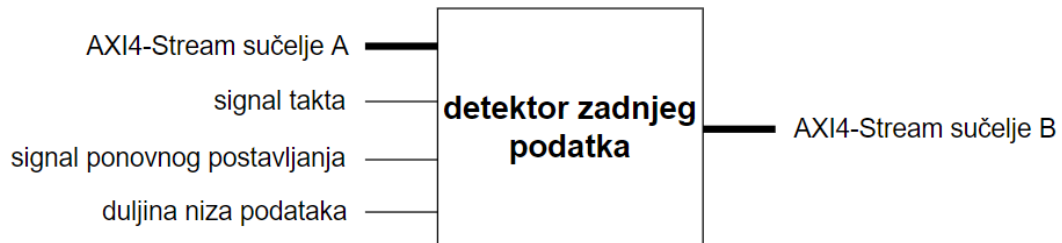
Tablica 5.2 - Parametri IP jezgre „Floating point“

PARAMETAR	ODABRANA OPCIJA
Odabir operacije	Množenje
Tip preciznosti	Jednostruka preciznost
Korištenje DSP blokova	Potpuno korištenje
Kontrola toka	Ne-blokirajući način
Latencija i konfiguracija brzine	Koristi maksimalnu latenciju (8)

Operacija množenja započinje nakon što se na podatkovne linije *AXI4-Stream sučelja A* i *AXI4-Stream sučelja B* postave željeni operandi te se tijekom jednog perioda takta aktiviraju linije koje signaliziraju da su podaci validni. Potrebno je 8 perioda takta kako bi se izračunao rezultat, no struktura ovog modula je cjevovod pa se na ulaz mogu uzastopno dovoditi podaci. Ako se na ulaz bloka uzastopno dovode podaci, prvi rezultat će kasniti 8 perioda takta, a zatim, kad se cjevovod napuni, rezultati će se pojavljivati uzastopno, svaki period takta. Aktiviranjem odgovarajuće linije, *AXI4-Stream sučelja C* signalizira da je rezultat validan.

### 5.4.3. Detektor zadnjeg podatka

Detektor zadnjeg podatka s pripadajućim ulazima i izlazima je prikazan na slici 5.12.

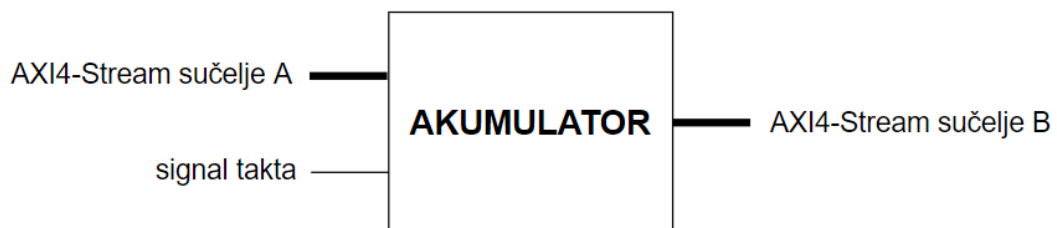


Slika 5.12 – Detektor zadnjeg podatka s pripadajućim ulazima i izlazima

Zadaća ovog modula je brojati koliko podataka je prosljedio s ulaza na izlaz te u periodu takta kada se na *AXI4-Stream sučelju A* pojavi zadnji podatak, aktivirati odgovarajuću liniju na sučelju *B* koja signalizira *akumulatoru* da će primiti zadnji podatak u nizu. Ovakvo ponašanje modula je uvjetovano ponašanjem *akumulatora*. *Duljina niza podataka* omogućuje zadavanje broja podataka u nizu.

### 5.4.4. Akumulator

*Akumulator* s pripadajućim ulazima i izlazima je prikazan na slici 5.13.



Slika 5.13 -Akumulator s pripadajućim ulazima i izlazima

Njegova zadaća je akumulirati sve umnoške težina i izlaza iz neurona iz prethodnog sloja te na izlazu dati validni rezultat akumulacije. Parametri modula (IP jezgre „Floating point“) su prikazani tablicom 5.3.

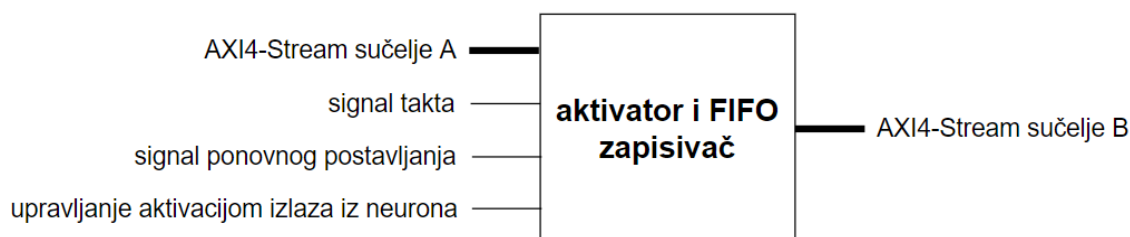
Tablica 5.3 - Parametri IP jezgre „Floating point“

PARAMETAR	ODABRANA OPCIJA
Odabir operacije	Akumulator
Zbrajanje/Oduzimanje	Zbrajanje
Tip preciznosti	Jednostruka preciznost
Optimizacije arhitekture	Velika brzina
Korištenje DSP blokova	Potpuno korištenje
Kontrola toka	Ne-blokirajući način
Latencija i konfiguracija brzine	Latencija: 8

Akumulacija se izvodi za svaki neuron u trenutno računatom sloju prema algoritmu propagacije unaprijed. Akumulator je u početku postavljen na 0. Ako se na podatkovne linije *AXI4-Stream sučelja A* postavi validni podatak, iznos akumulatora se poveća za iznos podatka na ulazu. Zadnji podatak u nizu ulaznih podataka, označen aktivnim stanjem odgovarajućeg signala, signalizira *akumulatoru* da završi postupak akumulacije. Kada je postupak akumulacije završen, na podatkovnim linijama *AXI4-Stream sučelja B* pojavljuje se validni iznos akumulatora. Nakon toga se iznos akumulatora ponovno postavlja na 0 te može započeti sljedeći postupak akumulacije.

#### 5.4.5. Aktivator i FIFO zapisivač

*Aktivator i FIFO zapisivač* s pripadajućim ulazima i izlazima je prikazan na slici 5.14.



Slika 5.14 – Aktivator i FIFO zapisivač s pripadajućim ulazima i izlazima

Zadaća ovog modula je aktiviranje rezultata akumulacije i spremanje izračunatih izlaza iz neurona u FIFO memoriju iz koje se isti, preko DMA, spremaju u DDR3 memoriju. Kad se na podatkovnim linijama *AXI4-Stream sučelja A* pojavi validni podatak, izvodi se aktivacija ako je

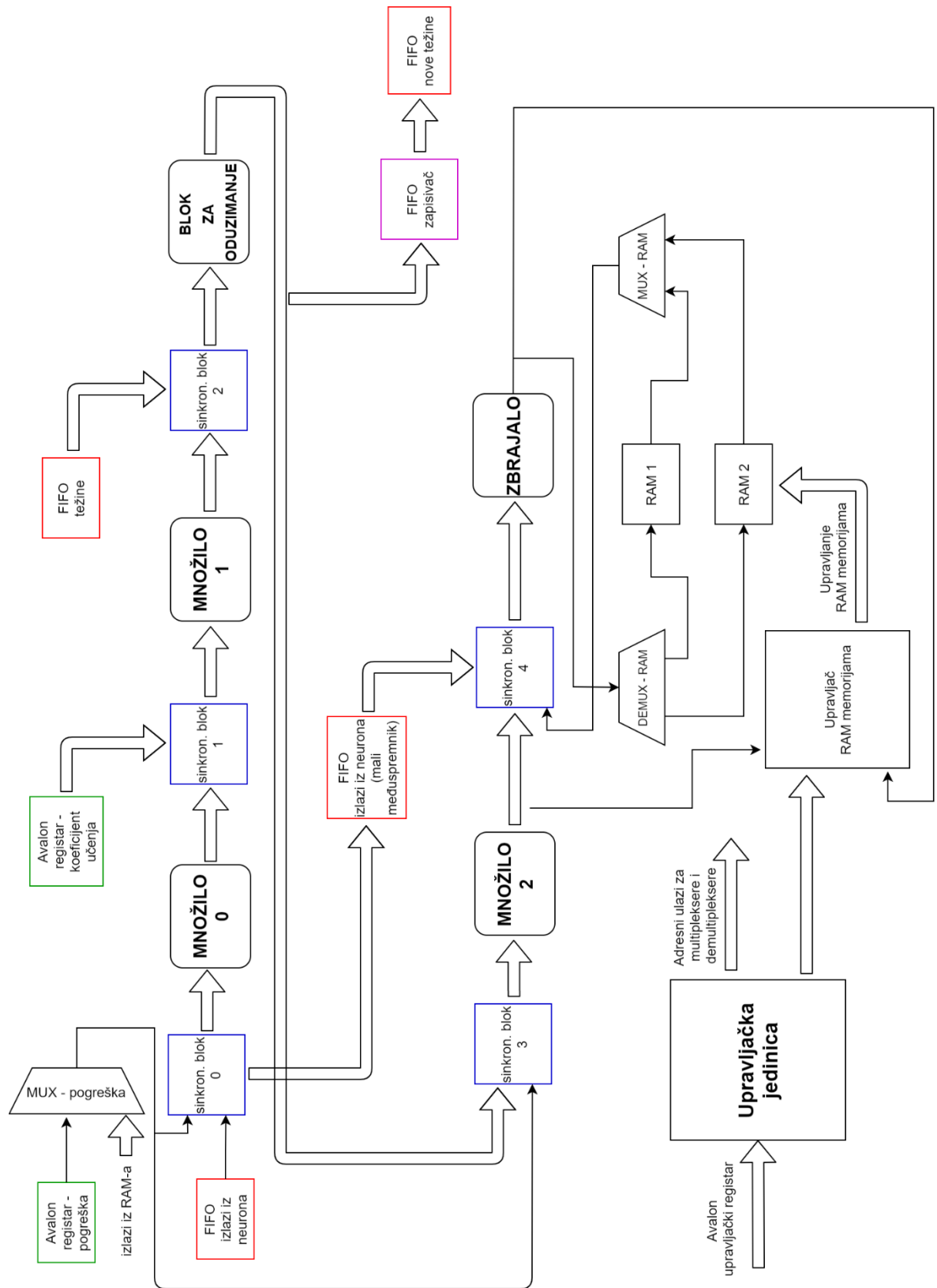
signal *upravljanje aktivacijom izlaza iz neurona* aktivan. Zatim se aktivirani podatak sprema u FIFO memoriju. Ako signal *upravljanje aktivacijom izlaza iz neurona* nije aktivan, podatak se sprema u FIFO memoriju u originalnom stanju. Signal *upravljanje aktivacijom izlaza iz neurona* odgovara LSB-u (engl. *Least Significant Bit - LSB*) *Avalon upravljačkog registra* koji je detaljnije opisan u poglavlju 5.6.

## 5.5. Blok za propagaciju unazad

Blok za propagaciju unazad implementira funkcionalnosti za izvršavanje algoritma propagacije unazad. Osnovna mu je zadaća izračunati nove iznose težina u trenutno obrađivanom sloju mreže te gradijente pogreške za idući obrađivani sloj. Prethodno spomenuti izlazi iz bloka računaju se na temelju starih iznosa težina i izlaza iz neurona zapamćenih u zadnjoj iteraciji propagacije unaprijed. Komunikacija između pojedinih modula je ostvarena AXI4-Stream protokolom. Blokovska shema bloka za propagaciju unazad je prikazana na slici 5.15.

Na ulaz modula *množilo 0* dovode se pripadajući izlaz iz neurona i pogreška prema izrazima (4-6), (4-11) i (4-13). Modul na izlazu daje gradijent pogreške potreban za izračun iznosa korekcije težine. Korekcija težine se zatim izračunava u *množilu 1*, na temelju koeficijenta učenja i gradijenta pogreške, prema izrazu (4-8). *Blok za oduzimanje* uzima iz FIFO memorije stari iznos težine te ga umanjuje za korekciju težine i kao rezultat daje novi iznos težine. *FIFO zapisivač* zapisuje nove iznose težina u *FIFO nove težine*.

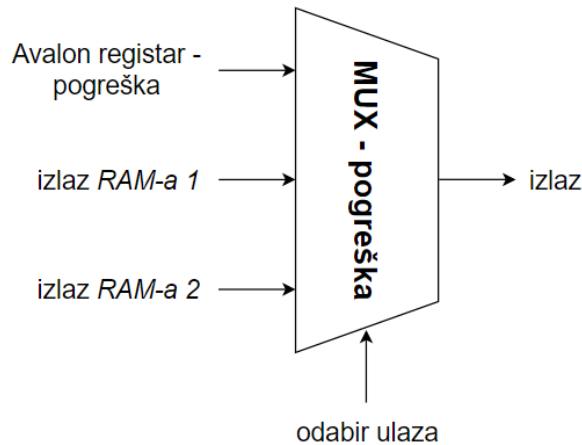
Sljedeći moduli u lancu služe za računanje gradijenata pogreške. *Množilo 2* množi pogrešku (izlaz modula *MUX – pogreška*) s novim iznosom težine, a rezultat je gradijent pogreške. Taj rezultat se akumulira s prethodnima prema izrazima (4-10) do (4-13). U akumulaciji sudjeluju moduli: *sinkronizacijski blok 4*, *zbrajalo*, *DEMUX-RAM*, *MUX-RAM*, *RAM 1*, *RAM 2* i *Upravljač RAM memorijama*. U RAM memorije se naizmjenično zapisuje i iz njih čita na sljedeći način. Ako se u *RAM 1* trenutno zapisuju gradijenti pogreške za idući obrađivani sloj, iz *RAM 2* se preko *MUX-pogreška* čitaju gradijenti pogreške. Vrijedi i obrnuto. Poseban slučaj je kada se obrađuje izlazni sloj mreže. Tada se u *RAM 1* zapisuju gradijenti pogreške za idući obrađivani sloj, a iz *Avalon registra – pogreške* se čitaju pogreške koje trenutno sudjeluju u izračunima u bloku.



Slika 5.15 - Blokovski prikaz bloka za propagaciju unazad

### 5.5.1. MUX – pogreška

Modul *MUX – pogreška* s pripadajućim ulazima i izlazima je prikazan na slici 5.16.



Slika 5.16 - Modul MUX - pogreška s pripadajućim ulazima i izlazima

*MUX – pogreška*, tj. multiplekser pogreške, omogućuje odabir između tri izvora pogreške: *Avalon registar – pogreška*, *izlaz RAM-a 1* i *izlaz RAM-a 2*. Tablica istinitosti prema kojoj se odabire ulaz koji se prosljeđuje na izlaz, prikazana je tablicom 5.4.

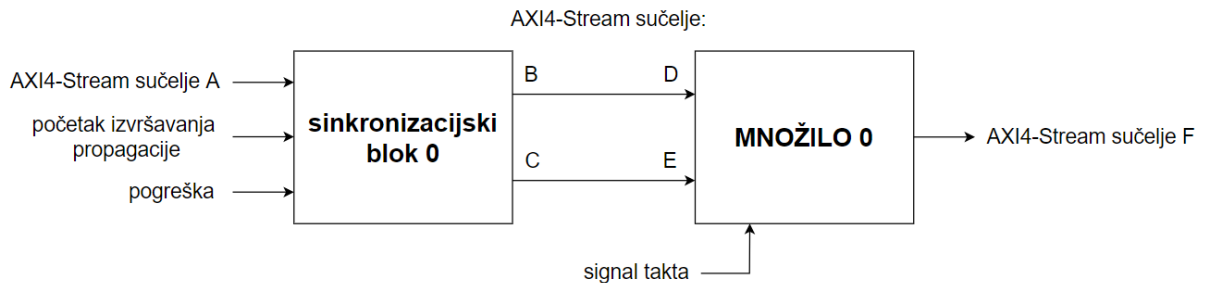
Tablica 5.4 - Tablica istinitosti multipleksera pogreške

ODABIR ULAZA	IZLAZ
„00“	Avalon registar – pogreška
„01“	izlaz RAM-a 1
„10“	izlaz RAM-a 2
inače	0

Zadaća multipleksera je omogućiti odabir izvora pogreške koji je različit u svakom sloju mreže. U slučaju obrađivanja izlaznog sloja mreže, na ulaz bloka za propagaciju unazad se preko *MUX – pogreška* dovodi podatak iz *Avalon registra – pogreška*. Te su pogreške izračunate u PS-u. Gradijenti pogreške, izračunati u izlaznom sloju za idući obrađivani sloj, spremljeni su u *RAM 1*. Iz prethodno navedenog slijedi da će se pri obradi idućeg sloja na ulaz bloka za propagaciju unazad odabirati *izlaz RAM-a 1*. U tom će se sloju računati i u *RAM 2* spremati gradijenti pogreške za idući obrađivani sloj pa će se u idućem sloju multiplekserom odabirati *izlaz RAM-a 2* itd.

## 5.5.2. Sinkronizacijski blok 0 i množilo 0

Blokovski dijagram spoja *sinkronizacijskog bloka 0* i *množila 0* s pripadajućim ulazima i izlazima je prikazan na slici 5.17.



Slika 5.17 - Blokovski dijagram spoja sinkronizacijskog bloka 0 i množila 0

Zadaća *sinkronizacijskog bloka 0* je prosljediti podatke s *AXI4-Stream sučelja A* na sučelje *B* te s ulaza *pogreška* na *AXI4-Stream sučelje C*. Prosljeđivanje podataka se izvršava ako je aktivan signal *početak izvršavanja propagacije* te ako je podatak na sučelju *A* validan.

*Množilo 0* započinje s operacijom množenja kada se na sučeljima *D* i *E*, u istom periodu takta, pojavi validan podatak. Ovaj modul za aritmetičke operacije je, kao i oni u poglavlju 5.4, preuzet iz Xilinx-ovog IP repozitorija pod nazivom „Floating point“ te mu je struktura cjevovod. Iz prethodno navedenog slijedi da se na ulaz mogu uzastopno dovoditi validni podaci, a prvi rezultat pojaviti će se na izlazu za onoliko perioda takta na koliko je postavljen parametar latencije. Parametri IP jezgre „Floating point“ kojima se realizira modul *množilo 0*, prikazani su tablicom 5.5.

Tablica 5.5 - Parametri IP jezgre "Floating point"

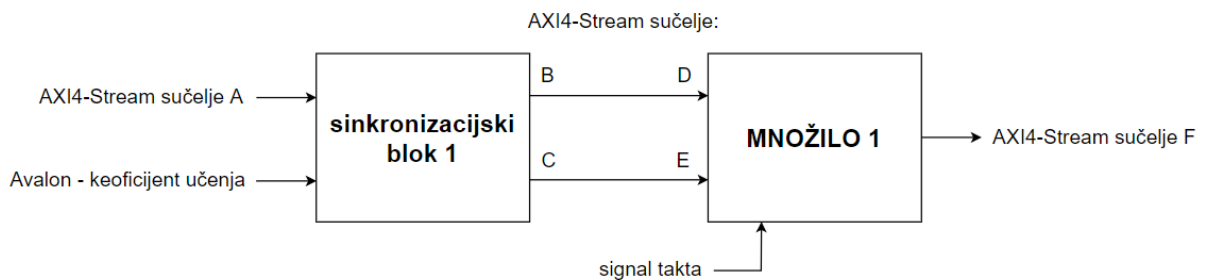
PARAMETAR	ODABRANA OPCIJA
Odabir operacije	Množenje
Tip preciznosti	Jednostruka preciznost
Korištenje DSP blokova	Potpuno korištenje
Kontrola toka	Ne-blokirajući način
Latencija i konfiguracija brzine	Koristi maksimalnu latenciju (8)



Podsustav sa slike 5.17 ima zadatau pomnožiti odgovarajuće izlaze iz neurona, pohranjene u *FIFO* izlazi iz neurona, s odgovarajućom pogreškom te na izlazu dati gradijent pogreške. Gradijent pogreške nadalje služi za izračun korekcije odgovarajuće težine.

### 5.5.3. Sinkronizacijski blok 1 i množilo 1

Blokovski dijagram spoja *sinkronizacijskog bloka 1* i *množila 1* s pripadajućim ulazima i izlazima je prikazan na slici 5.18.



Slika 5.18 - Blokovski dijagram spoja sinkronizacijskog bloka 1 i množila 1

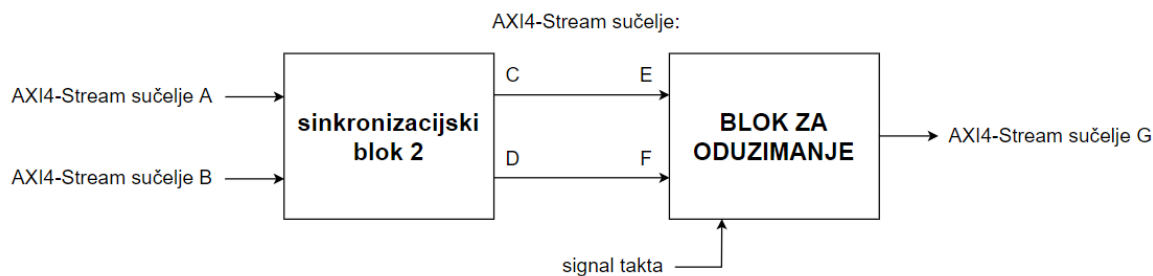
Zadaća *sinkronizacijskog bloka 1* je prosljediti podatke s *AXI4-Stream sučelja A* na *AXI4-Stream sučelje B* te s ulaza *Avalon registar – koeficijent učenja* na *AXI4-Stream sučelje C*. Podaci se prosljeđuju s ulaza na izlaz ako je podatak na sučelju *A* validan.

*Množilo 1* funkcioniра na isti način kao i *množilo 0*, što znači da operacija množenja započinje kad su oba podatka na ulazu u modul validni te se podaci na ulaz mogu dovoditi uzastopno. Ovaj modul je također preuzet u obliku IP jezgre iz Xilinx-ovog IP repozitorija pod nazivom „Floating point“ te su parametri IP jezgre jednaki onima u tablici 5.5.

Zadaća podsustava sa slike 5.18 je pomnožiti koeficijent učenja s odgovarajućim gradijentom pogreške te na izlazu dati korekciju odgovarajuće težine.

### 5.5.4. Sinkronizacijski blok 2 i blok za oduzimanje

Blokovski dijagram spoja *sinkronizacijskog bloka 2* i *bloka za oduzimanje* s pripadajućim ulazima i izlazima je prikazan na slici 5.19.



Slika 5.19 - Blokovski dijagram spoja sinkronizacijskog bloka 2 i bloka za oduzimanje

Zadaća sinkronizacijskog bloka 2 je proslijediti podatke s AXI4-Stream sučelja A na sučelje C te s AXI4-Stream sučelja B na sučelje D. Modul je preko sučelja A spojen na izlaz FIFO težine, a preko sučelja B na izlaz iz množila 1. Ako je podatak na sučelju B validan, modul čita jedan podatak iz FIFO težine te njega i podatak sa sučelja B prosljeđuje na izlaz modula.

Blok za oduzimanje izvršava operaciju oduzimanja nakon što se na sučeljima A i B, u istom periodu takta pojave validni podaci. Modul je kao i oni u prethodnim potpoglavljima preuzet u obliku IP jezgre iz Xilinx-ovog IP repozitorija pod nazivom „Floating point“ te funkcioniра na isti način, samo što umjesto množenja izvršava operaciju oduzimanja. Parametri IP jezgre „Floating point“ kojima se realizira blok za oduzimanje, prikazani su tablicom 5.6.

Tablica 5.6 - Parametri IP jezgre "Floating point"

PARAMETAR	ODABRANA OPCIJA
Odabir operacije	Zbrajanje/Oduzimanje
Zbrajanje/Oduzimanje	Oduzimanje
Optimizacije arhitekture	Velika brzina
Korištenje DSP blokova	Potpuno korištenje
Kontrola toka	Ne-blokirajući način
Latencija i konfiguracija brzine	Latencija: 8

Zadaća podsustava sa slike 5.19 je oduzeti korekciju težine od odgovarajuće težine iz FIFO težine. Rezultat je novi iznos težine koji se može spremi u DDR3 memoriju.

### 5.5.5. FIFO zapisivač

Modul *FIFO zapisivač* s pripadajućim ulazom i izlazom je prikazan na slici 5.20.

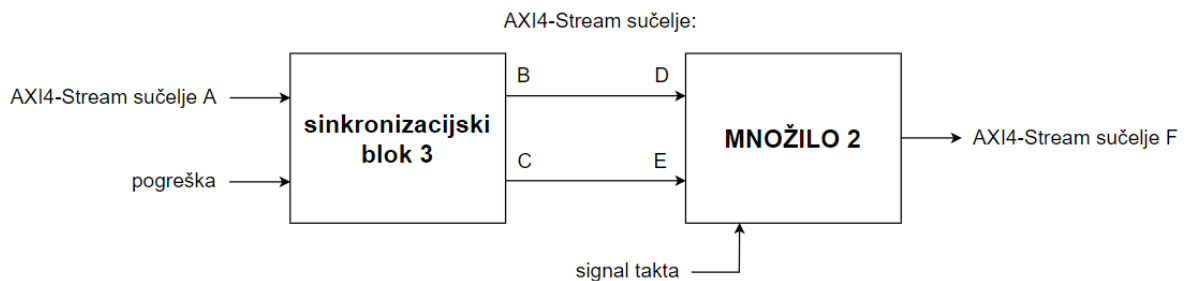


Slika 5.20 - Modul FIFO zapisivač s pripadajućim ulazom i izlazom

Zadaća ovog modula je zapisati iznos nove težine u *FIFO nove težine* kad se na *AXI4-Stream sučelju A* pojavi validan podatak. Zapisivanje se vrši putem *AXI4-Stream sučelja B*.

### 5.5.6. Sinkronizacijski blok 3 i množilo 2

Blokovski dijagram spoja *sinkronizacijskog bloka 3* i *množila 2* je prikazan na slici 5.21.



Slika 5.21 - Blokovski dijagram spoja sinkronizacijskog bloka 3 i množila 2

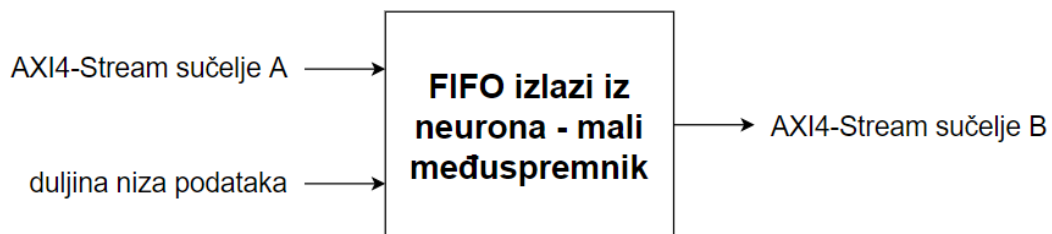
*Sinkronizacijski blok 3* ima zadaću proslijediti podatak s *AXI4-Stream sučelja A* na sučelje *B* te podatak s ulaza *pogreška* na sučelje *C* kad se na sučelju *A* pojavi validan podatak. Na sučelje *A* se dovode novi iznosi težina, a na ulaz *pogreška* se dovodi multipleksirana pogreška iz *MUX – pogreška*.

*Množilo 2* započinje s operacijom množenja kad su podaci na sučeljima *D* i *E*, u istom periodu takta, validni. Više detalja o množilu i IP jezgri „Floating point“ iz koje je množilo izvedeno, može se pročitati u potpoglavlju 5.5.2. Parametri IP jezgre „Floating point“, kojima se realizira *množilo 2*, jednaki su onima u tablici 5.5.

Zadaća podsustava sa slike 5.21 je pomnožiti odgovarajuću pogrešku i iznos nove težine kako bi se na izlazu dobio gradijent pogreške za idući obrađivani sloj. Taj gradijent pogreške još nije konačan, nego je utjecaj samo jedne veze između trenutno obrađivanog sloja i idućeg obrađivanog sloja. Iduća potpoglavlja opisuju sklopovlje za akumulaciju gradijenata koje za rezultat daje konačne gradijente pogreške u idućem obrađivanom sloju.

### 5.5.7. FIFO izlazi iz neurona – mali međuspremnik

Modul *FIFO izlazi iz neurona – mali međuspremnik* s pripadajućim ulazima i izlazima je prikazan na slici 5.22.



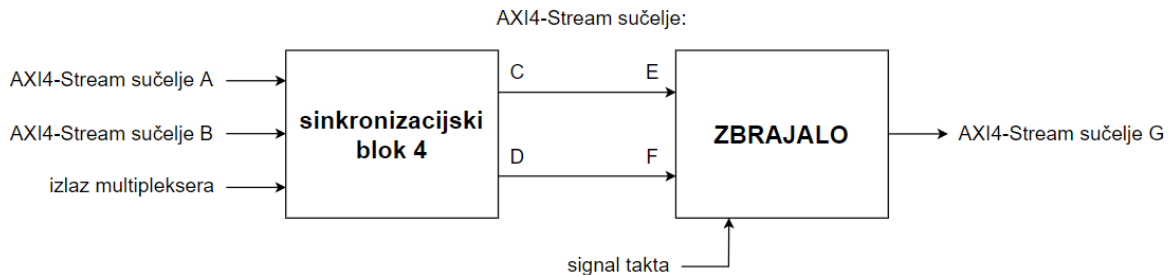
Slika 5.22 - Modul FIFO izlazi iz neurona - mali međuspremnik s pripadajućim ulazima i izlazima

Na sučelje A se dovodi podatak iz modula *FIFO izlazi iz neurona*, preko *sinkronizacijskog bloka 0*. Zapisivanje podataka u ovaj modul se izvršava kad je podatak na *AXI4-Stream sučelju A* sa slike 5.17 validan. Modul je preko *AXI4-Stream sučelja B* povezan sa *sinkronizacijskim blokom 4* koji iz njega, prema potrebi, čita podatke. Ulazom *duljina niza podataka* se zadaje broj podataka koje modul treba primiti i poslati. Zadnji podatak u nizu je označen odgovarajućim signalom u aktivnom stanju.

*Sinkronizacijski blok 0* i *sinkronizacijski blok 4* čitaju podatke iz *FIFO izlazi iz neurona* sa slike 5.15. Struktura bloka za propagaciju unazad je cjevovod, a ukupna latencija podataka do *sinkronizacijskog bloka 4* je zbroj latencija svih modula za aritmetičke operacije do *sinkronizacijskog bloka 4*. Iz prethodno navedenog slijedi da je latencija do *sinkronizacijskog bloka 4* jednaka 32 perioda takta. Budući da *sinkronizacijski blok 0* počinje čitati podatke iz *FIFO izlazi iz neurona* 32 perioda takta prije *sinkronizacijskog bloka 4*, potrebno je u *FIFO izlazi iz neurona – mali međuspremnik* spremati ta 32 podatka, što omogućuje *sinkronizacijskom bloku 4* da uvijek čita validne podatke. Slijedi da je zadaća modula sa slike 5.22 omogućiti *sinkronizacijskom bloku 4* čitanje validnih podataka iz *FIFO izlazi iz neurona*.

### 5.5.8. Sinkronizacijski blok 4 i zbrajalo

Blokovski dijagram spoja *sinkronizacijskog bloka 4* i *zbrajala* s pripadajućim ulazima i izlazima je prikazan na slici 5.23.



Slika 5.23 - Blokovski dijagram spoja sinkronizacijskog bloka 4 i zbrajala s pripadajućim ulazima i izlazima

Modul *sinkronizacijski blok 4* je povezan s *FIFO* izlazi iz neurona (*mali međuspremnik*) preko sučelja *A*. Preko sučelja *B* je povezan na izlaz *množila 2*. Također, na ulaz modula se preko *izlaza multipleksera* može dovesti izlaz bilo koje RAM memorije. *Sinkronizacijski blok 4* propušta podatke sa sučelja *B* na sučelje *C* i s ulaza *izlaz multipleksera* na sučelje *D* ako je podatak na sučelju *B* validan. Također, kad je podatak na sučelju *B* validan, modul čita podatak iz *FIFO* izlazi iz neurona (*mali međuspremnik*) preko sučelja *A*. Ako je pročitana vrijednost 0, na sučelje *C* se prosljeđuje 0, a inače se prosljeđuje podatak sa sučelja *B*. Na sučelje *D* se uvijek prosljeđuje podatak s *izlaza multipleksera*.

*Zbrajalo* je kao i u prethodnim poglavljima preuzeto u obliku IP jezgre pod nazivom „Floating point“ iz Xilinx-ovog IP repozitorija te počinje s operacijom zbrajanja kad su oba podatka na sučeljima *E* i *F* validni. Parametri IP jezgre „Floating point“ su prikazani tablicom 5.7. Rezultati zbrajanja spremaju se u jednu od RAM memorija preko sučelja *G* i demultipleksera *DEMUX – RAM*.

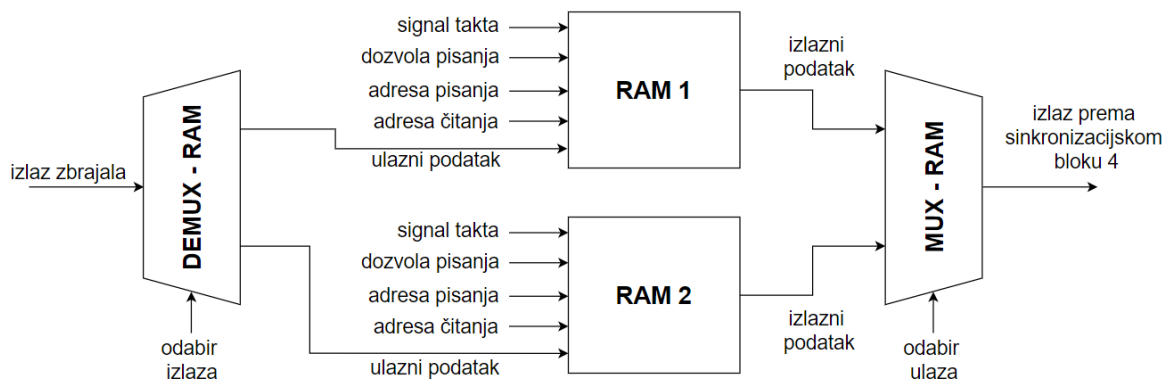
Zadaća podsustava prikazanog na slici 5.23 je akumulirati gradijente pogreške za idući obrađivani sloj mreže. Podsustav uzima podatak iz odgovarajuće RAM memorije koji predstavlja akumulirani gradijent pogreške za odgovarajući neuron u idućem obrađivanom sloju te njemu pribraja novi izračunati gradijent pogreške prema izrazima (4-10) i (4-12).

Tablica 5.7 - Parametri IP jezgre "Floating point"

PARAMETAR	ODABRANA OPCIJA
Odabir operacije	Zbrajanje/oduzimanje
Zbrajanje/Oduzimanje	Zbrajanje
Optimizacije arhitekture	Velika brzina
Korištenje DSP blokova	Potpuno korištenje
Kontrola toka	Ne-blokirajući način
Latencija i konfiguracija brzine	Latencija: 8

### 5.5.9. Demultiplekser, multiplekser i RAM memorije

Blokovski dijagram spoja demultipleksera, multipleksera i dvaju RAM memorija s pripadajućim ulazima i izlazima je vidljiv na slici 5.24.



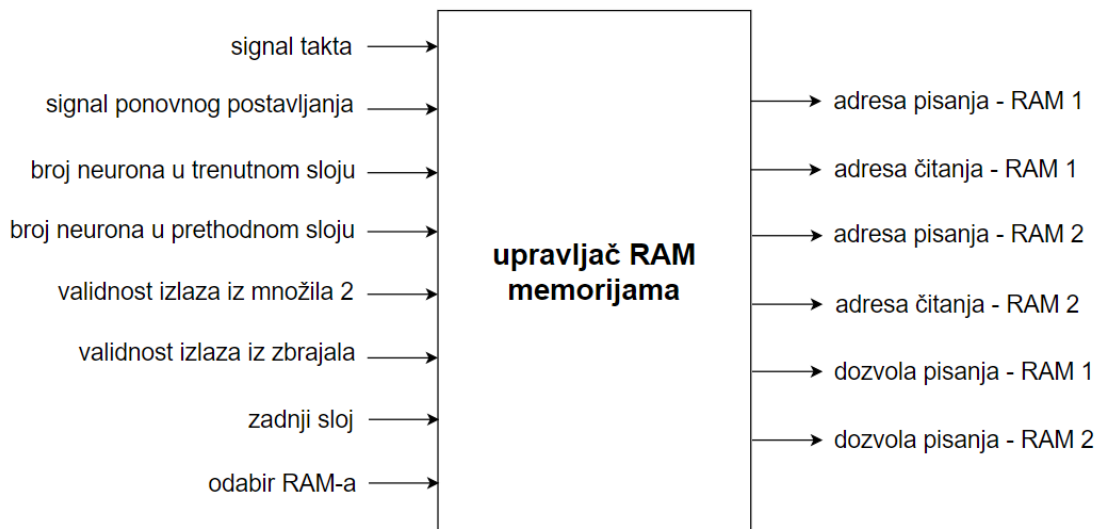
Slika 5.24 - Blokovski dijagram spoja demultipleksera, multipleksera i dvaju RAM memorija

Budući da zbrajalo mora imati mogućnost spremanja rezultata u obje RAM memorije, kreiran je demultiplekser *DEMUX – RAM* koji prema linijama *odabir izlaza*, postavlja *izlaz zbrajala* na ulaz *RAM-a 1* ili *RAM-a 2*. Obje RAM memorije u ovom podsustavu su tzv. „dual-port“ RAM memorije. Podatak se u RAM zapisuje tako da se na linije *ulazni podatak* postavi podatak koji se želi zapisati te se tijekom jednog perioda takta aktivira signal *dozvola pisanja*. Izlaz memorije je kombinacijski, što znači da se prema postavljenoj *adresi čitanja*, trenutno (ako zanemarimo vrijeme propagacije signala kroz kombinacijsku mrežu) postavlja *izlazni podatak*. Multiplekser *MUX – RAM* pruža mogućnost odabiranja *izlaznog podatka* iz jedne od dvaju RAM memorija.

Upravljanje RAM memorijama, u vidu dozvole pisanja i adresiranja memorije, čini *upravljач RAM memorijama* čiji je opis dan u sljedećem potpoglavlju.

### 5.5.10. Upravljač RAM memorijama

Modul *upravljач RAM memorijama* s pripadajućim ulazima i izlazima je prikazan na slici 5.25.



Slika 5.25 - Modul upravljач RAM memorijama s pripadajućim ulazima i izlazima

Zadaća ovog modula je upravljanje zapisivanjem i čitanjem podataka iz obje RAM memorije te adresiranje istih. Upravljanje ovisi o sljedećim ulazima u modul: *broj neurona u trenutnom sloju*, *broj neurona u prethodnom sloju*, *validnost izlaza iz množiла 2*, *validnost izlaza iz zbrajala*, *zadnji sloj* i *odabir RAM-a*. Opis ponašanja modula prema zadanim ulazima je prikazan tablicom 5.8. Napomena: *broj neurona u prethodnom sloju* se odnosi na broj neurona u sloju koji je obrađivan u prethodnom koraku.

Ako se trenutno obrađuje zadnji sloj, gradijenti pogreške za idući sloj se zapisuju u *RAM 1*. *Adresa čitanja – RAM 1* se inkrementira ako je *validan izlaz iz množiла 2*. Također, *adresa pisanja – RAM 1* se inkrementira ako je *validan izlaz iz zbrajala* te se u istom periodu takta aktivira *dozvola pisanja – RAM 1*. Obje adrese se ponovno postavljaju na 0 kad postignu vrijednost zadanu *brojem neurona u trenutnom sloju*.

Ako se ne obrađuje zadnji sloj, a *odabir RAM-a* je postavljen u logičko stanje '0', gradijenti pogreške za idući sloj se spremaju u *RAM 1*, a iz *RAM-a 2* se čitaju gradijenti pogreške izračunati u prethodnom sloju. *Adresa pisanja – RAM 1* se inkrementira ako je *validan izlaz iz množiла 2* te

se u istom periodu takta aktivira *dozvola pisanja – RAM 1*, a *adresa čitanja – RAM 1* se inkrementira ako je *validan izlaz iz zbrajala*. *Adresa čitanja – RAM 2* se inkrementira ako je *validan izlaz iz množila 2* te ako je *adresa čitanja – RAM 1* jednaka *broju neurona u trenutnom sloju*. Obje adrese *RAM-a 1* se ponovno postavljaju na 0 kad postignu vrijednost zadanu *brojem neurona u trenutnom sloju*, a *adresa čitanja – RAM 2* se ponovno postavlja na 0 ako se trenutno ponovno postavlja *adresa čitanja – RAM 1* i ako je *adresa čitanja – RAM 2* postigla vrijednost zadanu *brojem neurona u prethodnom sloju*. Vrijedi obrnuto ako je *odabir RAM-a* u logičkom stanju '1'.

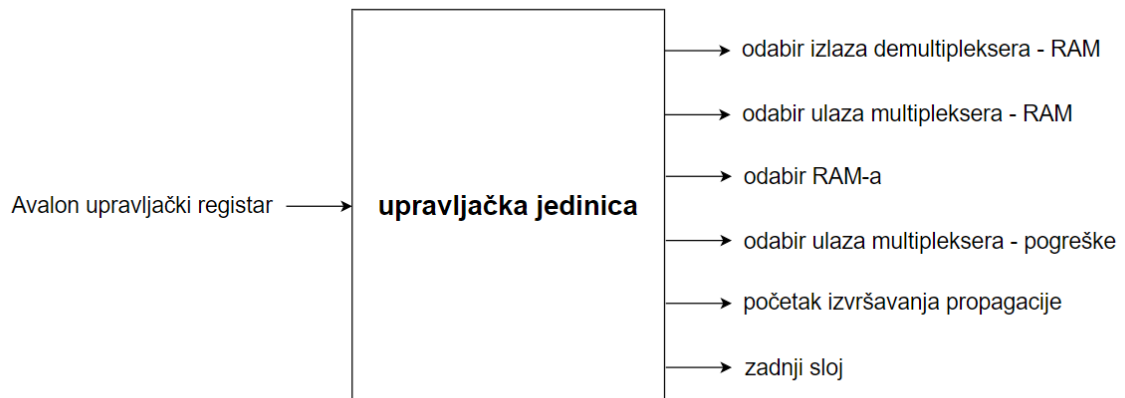
Tablica 5.8 - Opis ponašanja upravljača RAM memorijama prema zadanim ulazima

ULAZI		OPIS PONAŠANJA
<i>zadnji sloj</i>	<i>odabir RAM-a</i>	
'1'	X	Gradijenti pogreške za idući sloj se zapisuju u <i>RAM 1</i> .
'0'	'0'	Gradijenti pogreške za idući sloj se zapisuju u <i>RAM 1</i> , a iz <i>RAM-a 2</i> se čitaju gradijenti izračunati u prethodnom sloju (preko <i>MUX - pogreška</i> ).
'0'	'1'	Gradijenti pogreške za idući sloj se zapisuju u <i>RAM 2</i> , a iz <i>RAM-a 1</i> se čitaju gradijenti izračunati u prethodnom sloju (preko <i>MUX - pogreška</i> ).

### 5.5.11. Upravljačka jedinica

Modul upravljačka jedinica s pripadajućim ulazima i izlazima je prikazan na slici 5.26. Upravljačka jedinica u ovom slučaju je kombinacijski sklop, koji prema 4-bitnom ulaznom vektoru *Avalon upravljački registar* generira izlaze.





Slika 5.26 - Modul upravljачka jedinica s pripadajućim ulazima i izlazima

Procesor pristupa *Avalon upravljачkom registru* te postavlja određene bitove tako da se ostvari željena funkcionalnost bloka za propagaciju unazad. Tablica 5.9 prikazuje mapu ulaza *Avalon upravljачki registar*. Važno je naglasiti da je prethodno spomenuti ulaz samo dio upravljачkog registra koji je rezerviran za upravljanje blokom za propagaciju unazad. Točnije bitovi s rasponom indeksa [4,1] odgovaraju bitovima s rasponom [3,0] iz tablice 5.9. Mapa registra u cjelini je dana u poglavlju 5.6.

Tablica 5.9 – Mapa ulaza Avalon upravljачki registar

3	2	1	0
inicijalizacija gradijenata	zadnji sloj	odabir RAM-a	početak izvršavanja propagacije

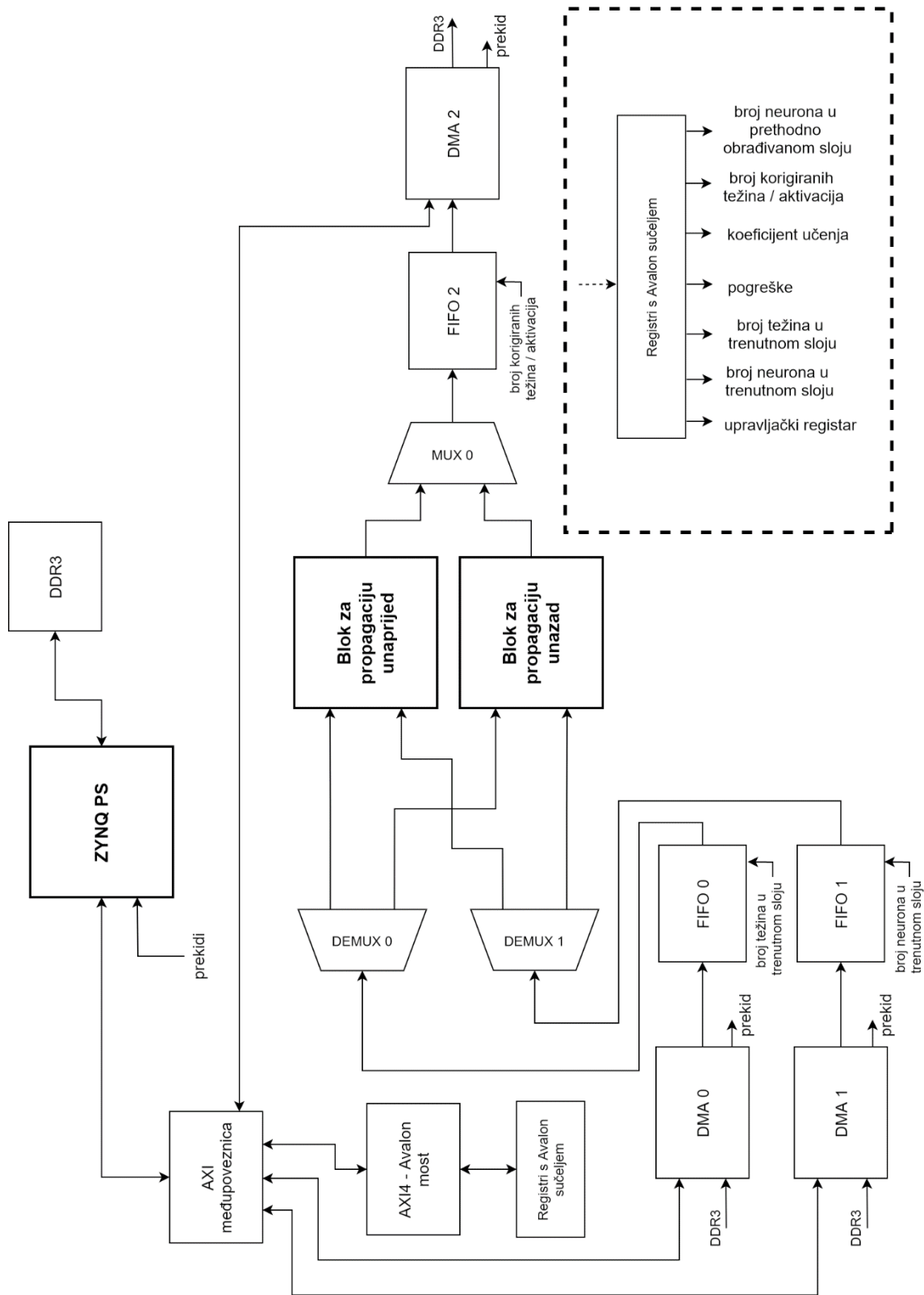
Neki bitovi upravljачkog registra, kao što su: *početak izvršavanja propagacije*, *zadnji sloj* i *odabir RAM-a*, se jednostavno prosljeđuju s ulaza na izlaz modula, a izlazi: *odabir izlaza demultipleksera – RAM*, *odabir ulaza multipleksera – RAM* i *odabir ulaza multipleksera – pogreške* dobivaju se kombinacijama određenih ulaza koje su prikazane tablicom istinitosti 5.10.

Tablica 5.10 - Tablica istinitosti upravljačke jedinice

IZLAZ UPRAVLJAČKE JEDINICE	ULAZI			LOGIČKO STANJE IZLAZA
	inicijalizacija gradijenata	zadnji sloj	odabir RAM-a	
odabir izlaza demultipleksera – RAM	X	'1'	X	'0'
	X	'0'	'0'	'1'
	X	'0'	'1'	'1'
	inače			'0'
odabir izlaza multipleksera – RAM	'0'	'1'	X	„00“
	'0'	'0'	'0'	
	'0'	'0'	'1'	„01“
	'1'	X	X	„10“
	inače			„11“
odabir ulaza multipleksera - pogreške	X	'1'	X	„00“
	X	'0'	'1'	„01“
	X	'0'	'0'	„10“
	inače			„00“

## 5.6. Pregled potpunog rješenja

U ovome poglavlju je u cijelosti opisan način funkcioniranja sustava za prepoznavanje znamenki. Blokovski dijagram sustava je vidljiv na slici 5.27. Na prethodno opisane blokove za propagaciju unaprijed i unazad, dodan je omotač kojim se ostvaruje put podataka iz DDR3 memorije prema prethodno spomenutim blokovima te iz blokova prema DDR3 memoriji. Omotač također omogućuje komunikaciju PS-PL koja je detaljnije objašnjena u poglavlju 5.3.



Slika 5.27 - Blokovski dijagram potpunog sustava za prepoznavanje znamenki

Putem *upravljačkog registra*, sa slike 5.27, upravlja se cijelim sustavom, tj. zapisivanjem određenog podatka u isti se izvršava određena funkcija sustava. Tablica 5.11 prikazuje mapu *upravljačkog registra s Avalon sučeljem*, gdje se bit 0 odnosi isključivo na *blok za propagaciju unaprijed*, a bitovi s rasponom indeksa [4,1] na *blok za propagaciju unazad*. Bit 5, s oznakom: *odabir funkcije sustava*, omogućuje odabir puta podataka u sustavu. Točnije, ako se bit postavi u logičko stanje '0', podaci preko *DEMUX 0* i *DEMUX 1* ulaze u *blok za propagaciju unaprijed*, a izlaze iz istog preko *MUX 0* te se nadalje spremaju u *DDR3* memoriju. Ako se bit postavi u logičko stanje '1', podaci na isti način ulaze i izlaze iz *bloka za propagaciju unazad*. Iz prethodno navedenog slijedi da bit s oznakom *odabir funkcije sustava* služi kao selekcijski ulaz za module *DEMUX 0*, *DEMUX 1* i *MUX 0*. Pri *upravljanju blokom za propagaciju unazad*, bit 1 treba aktivirati kada se želi započeti s izračunima u bloku. Postavljanjem bita 2 u odgovarajuće stanje, odabire se RAM memorija u *bloku za propagaciju unazad* u koju se zapisuju gradijenti pogreške za idući obrađivani sloj. Nadalje, aktiviranjem bita 3, *blok za propagaciju unazad* dobiva informaciju da se trenutno obrađuje zadnji sloj mreže (izlazni) te se ponaša sukladno tome. Aktiviranjem bita 4 omogućuje se ponovno postavljanje vrijednosti u RAM memorijama za gradijente pogreške, tj. ako se trenutno započinju računati i akumulirati gradijenti pogreške za idući obrađivani sloj, aktivira se ovaj bit i time se postiže akumuliranje novih gradijenata s nulom. Jednostavnije rečeno, na ovaj način je izvedena funkcija postavljanja svih podataka u memoriji na vrijednost nula koja se izvodi u toku spremanja novih gradijenata. Funkcija bita 0 je objašnjena u poglavlju 5.4.5.

Tablica 5.11 - Mapa upravljačkog registra s Avalon sučeljem

5	4	3	2	1	0
odabir funkcije sustava	inicijalizacija gradijenata	zadnji sloj	odabir RAM-a	početak izvršavanja propagacije	upravljanje aktivacijom izlaza iz neurona

Algoritam propagacije unaprijed izvršava se na sljedeći način. Odabire se put podataka kroz *blok za propagaciju unaprijed* te se pomoću bita *upravljanje aktivacijom izlaza iz neurona* određuje želi li se aktivirati izlaze trenutno obrađivanih neurona, a zatim se pokreću DMA transferi težina (*DMA 0*), izlaza iz neurona (*DMA 1*) i izračunatih izlaza iz neurona (*DMA 2*). *Blok za propagaciju unaprijed* zatim uzima podatke iz *FIFO 0* i *FIFO 1* te sprema rezultate u *FIFO 2* koji se nadalje preko *DMA 2* spremaju u *DDR3* memoriju.

Algoritam propagacije unazad izvršava se na sljedeći način. Odabire se put podataka kroz *blok za propagaciju unazad* te se prema potrebi postavljaju bitovi *upravljačkog registra* s rasponom indeksa [4,1] te registar *pogreška*. Zatim se pokreću DMA transferi težina (*DMA 0*), izlaza iz neurona (*DMA 1*) i korigiranih težina (*DMA 2*). *Blok za propagaciju unazad* zatim uzima podatke iz *FIFO 0* i *FIFO 1* te sprema korigirane težine u *FIFO 2* koji se nadalje preko *DMA 2* spremaju u DDR3.

## **5.7. Programaska podrška**

U ovome poglavlju je opisana programaska podrška sustava za prepoznavanje znamenki. Upravljački program (engl. *driver*) je opisan pomoću dijagrama toka, a spomenuti su i najvažniji parametri aplikacije. Budući da je pri programiranju FPGA moguće odabrati između dvije opcije: klasifikacijski način rada i trenirajući način rada, u nastavku su nakon opisa glavnih parametara aplikacije zasebno opisani upravljački programi za oba načina rada.

### **5.7.1. Glavni parametri aplikacije**

Prije pokretanja aplikacije, u DDR3 memoriju je potrebno zapisati sljedeće podatke: skup slika za treniranje, skup slika za testiranje, skup inicijaliziranih težina, očekivane izlaze skupa za treniranje i očekivane izlaze skupa za testiranje. Prethodno spomenuti podaci se zapisuju u velikoj količini pa je za svaku skupinu podataka rezerviran određen dio memorije. Tablica 5.12 prikazuje početne adrese svakog skupa podataka u DDR3 memoriji ZYBO razvojne ploče. U tablici se također nalaze i adrese skupova podataka u koje se spremaju međurezultati: izlazi prvog sloja, izlazi drugog sloja, izlazi trećeg sloja i skup korigiranih težina.

Tablica 5.12 – Početne adrese skupova podataka u DDR3 memoriji

<b>SKUP PODATAKA</b>	<b>POČETNA ADRESA</b>
skup slika za treniranje	0x01B00000
skup slika za testiranje	0x15000000
skup inicijaliziranih težina	0x01100000
očekivani izlazi skupa za treniranje	0x01860000
očekivani izlazi skupa za testiranje	0x01840000
izlazi prvog sloja	0x01600000
izlazi drugog sloja	0x01700000
izlazi trećeg sloja	0x01800000
skup korigiranih težina	0x01900000

U prethodnim poglavljima dana je informacija da se PS-PL komunikacija ostvaruje uglavnom preko registara s Avalon sučeljem te da procesor prema potrebi upisuje podatak u neki od tih registara. Xilinx SDK daje baznu adresu AXI-Avalon mosta koja će u nastavku biti označena s „AXI\_AVALON\_BAZNA\_ADR“, a relativna adresa pojedinog registra određena je samim VHDL opisom modula *registri s Avalon sučeljem* sa slike 5.27. Relativne adrese pojedinih registara s Avalon sučeljem prikazane su u tablici 5.13, a iste se nalaze u virtualnom adresnom prostoru.

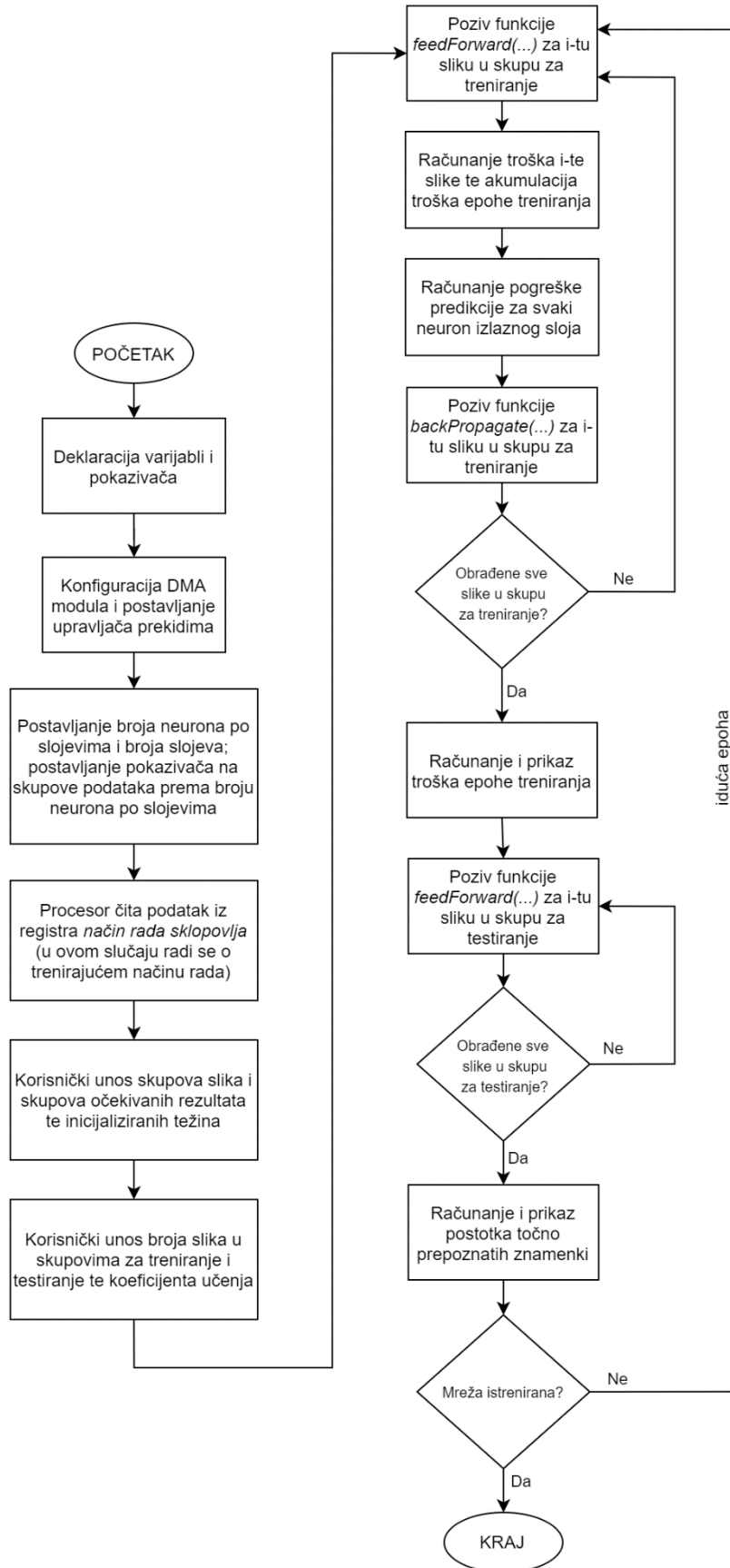
Tablica 5.13 - Relativne adrese registara s Avalon sučeljem

<b>NAZIV REGISTRA</b>	<b>RELATIVNA ADRESA</b>
upravljački registar	AXI_AVALON_BAZNA_ADR + 0x0
broj neurona u trenutno obrađivanom sloju	AXI_AVALON_BAZNA_ADR + 0x4
broj težina	AXI_AVALON_BAZNA_ADR + 0x8
pogreška	AXI_AVALON_BAZNA_ADR + 0xC
koeficijent učenja	AXI_AVALON_BAZNA_ADR + 0x10
broj korigiranih težina	AXI_AVALON_BAZNA_ADR + 0x14
broj neurona u prethodnom sloju	AXI_AVALON_BAZNA_ADR + 0x18
način rada sklopovlja	AXI_AVALON_BAZNA_ADR + 0x1C

Zbog mogućnosti odabiranja klasifikacijskog i trenirajućeg načina rada, u sustav je dodan registar *način rada sklopovlja* koji nema utjecaja na sustav u PL-u. Pri odabiranju načina rada sklopovlja i programiranja FPGA, ovaj registar se postavlja u određeno stanje, a procesor u određenom trenutku čita sadržaj tog registra te se prema postavkama sklopovlja izvršava određeni dio programa. Npr. ako procesor iz registra pročita vrijednost 1, procesor će upravljati sklopovljem tako da se neuronska mreža trenira.

### **5.7.2. Upravljački program za trenirajući način rada sklopovlja**

Dijagram toka upravljačkog programa za trenirajući način rada sklopovlja je vidljiv na slici 5.28. Ovaj upravljački program, kao i većina drugih za ugradbene računalne sustave, započinje općim postavljanjem sustava kao što je deklariranje i inicijalizacija varijabli, konfiguracija modula povezanih na procesor, korisnički unos parametara itd. Nakon uspješnog postavljanja sustava slijedi glavna petlja u kojoj se izvršava treniranje mreže, tj. jednom iteracijom petlje izvršava se jedna epoha treniranja. Glavna petlja sadržava dvije petlje, gdje se u prvoj izvršava treniranje mreže nad skupom slika za treniranje. To podrazumijeva izvršavanje algoritma propagacije unaprijed, računanje troška slike, akumulaciju troška epohe treniranja, računanje pogreške prepoznavanja i izvršavanje algoritma propagacije unazad. Nakon izvršene epohe treniranja u prvoj petlji, u drugoj petlji se izvršava testiranje mreže nad skupom slika za testiranje. To podrazumijeva izvršavanje algoritma za propagaciju unaprijed za svaku sliku u skupu za testiranje te po završetku petlje računanje i prikazivanje postotka točno prepoznatih znamenaka. Ovime je izvršena jedna epoha treniranja neuronske mreže te se izvršava sljedeća ako mreža nije dovoljno utrenirana. Ako je dovoljno utrenirana, program završava.

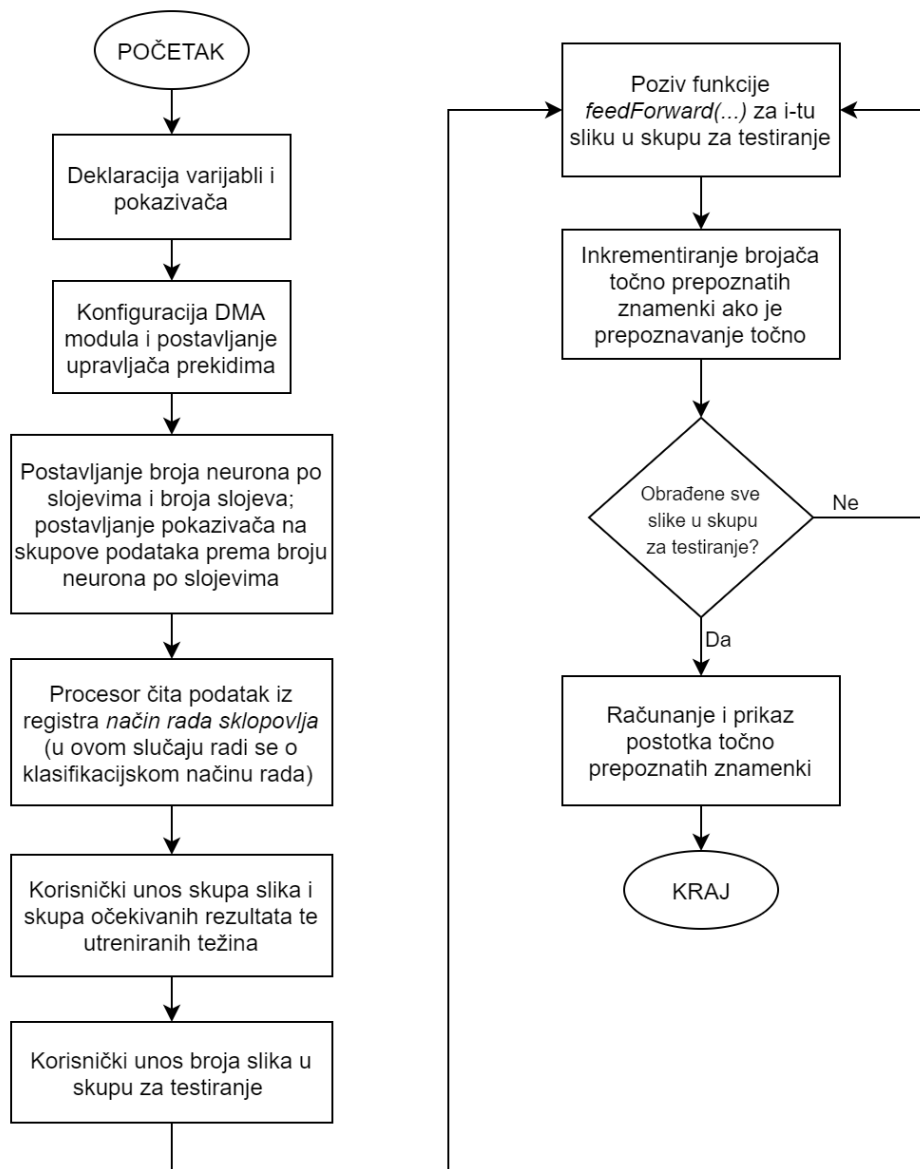


Slika 5.28 - Dijagram toka upravljačkog programa za trenirajući način rada sklopovlja



### 5.7.3. Upravljački program za klasifikacijski način rada sklopovlja

Dijagram toka upravljačkog programa za klasifikacijski način rada sklopovlja je vidljiv na slici 5.29. Upravljački program, kao i onaj sa slike 5.28, započinje općim postavljanjem sustava te unosom parametara. Zatim se u petlji izvršava algoritam propagacije unaprijed i inkrementiranje brojača točno prepoznatih znamenki za svaku sliku iz skupa za testiranje. Program završava nakon što su obrađene sve slike iz skupa za testiranje te je izračunat i prikazan postotak točno prepoznatih znamenki.



Slika 5.29 - Dijagram toka upravljačkog programa za klasifikacijski način rada sklopovlja

#### 5.7.4. Glavne funkcionalnosti aplikacije

Funkcije *feedForward(...)* i *backPropagate(...)* glavni su dio upravljačkog programa sustava za prepoznavanje znamenki, a glavna im je zadaća omogućiti izvršavanje algoritma za propagaciju unaprijed i unazad. Jedan poziv funkcije odgovara izvršavanju odgovarajućeg algoritma za jednu sliku.

Funkcija *feedForward(...)* omogućuje upravljanje sklopovljem tako da se izvrši algoritam propagacije unaprijed. Funkcija se izvršava na sljedeći način. Aktivacija se na izlazima iz neurona omogućuje u svim slojevima osim izlaznog na način da se bit 0 *upravljačkog registra* iz tablice 5.11 postavi u logičko stanje '1'. Prije pokretanja DMA transfera podataka za svaki pojedini sloj, u odgovarajuće registre se zapisuju broj neurona i težina u trenutnom sloju. Zatim se iterativno pokreću DMA transferi. Tablica 5.14 prikazuje koji se podaci prenose DMA transferima u kojem sloju neuronske mreže. Važno je naglasiti da procesor nakon pokretanja DMA transfera čeka da DMA izvrši prekid, a zatim se u prekidnoj rutini podigne zastavica koja signalizira da je DMA transfer podataka završio. Kada se odgovarajući DMA transferi završe, započinju se novi DMA transferi sljedećih podataka u nizu itd. Nakon uspješne obrade izlaznog sloja u PL-u, procesor izvršava Softmax aktivacijsku funkciju nad izlazima neurona izlaznog sloja te provjerava koja znamenka ima najveću vjerojatnost pojavljivanja na ulaznoj slici. Funkcija vraća 1 ako je prepoznata znamenka jednaka očekivanoj znamenki, a u suprotnom vraća 0. Također, na mjesta u memoriji zadana pokazivačima u parametarskoj listi, vraća vjerojatnosti pojavljivanja pojedinih znamenaka te znamenku s najvećom vjerojatnošću pojavljivanja.

Tablica 5.14 – Ovisnost sloja neuronske mreže i podataka u DMA transferima

SLOJ	DMA 0	DMA 1	DMA 2
prvi skriveni sloj	težine ( <i>ulazni</i> → <i>prvi skriveni sloj</i> )	pikseli ulazne slike	izlazi neurona prvog skrivenog sloja
drugi skriveni sloj	težine ( <i>prvi skriveni sloj</i> → <i>drugi skriveni sloj</i> )	izlazi neurona prvog skrivenog sloja	izlazi neurona drugog skrivenog sloja
izlazni sloj	težine ( <i>drugi skriveni sloj</i> → <i>izlazni sloj</i> )	izlazi neurona drugog skrivenog sloja	izlazi neurona izlaznog sloja

Funkcija *backPropagate(...)* omogućuje upravljanje sklopovljem tako da se izvrši algoritam propagacije unazad. Funkcija se izvršava na sljedeći način. Prije izvršavanja propagacije unazad,

u registar *koeficijent učenja* sa slike 5.27 se zapisuje koeficijent učenja. Zatim se u petlji izvršavaju izračuni za svaki sloj, gdje jedna iteracija petlje odgovara jednom sloju mreže. Također, za svaki sloj se iterativno pokreću DMA transferi te se u prvoj iteraciji ponovno postavljaju akumulirani gradijenti pogreške u odgovarajućoj RAM memoriji, kao što je opisano u poglavlju 5.6. Tablica 5.15 prikazuje koji se podaci prenose DMA transferima te koji je izvor gradijenata pogrešaka u kojem sloju neuronske mreže. Kada odgovarajući DMA transferi podataka završe, započinju se DMA transferi sljedećih podataka u nizu. Okidanje prekidnih rutina i pokretanje sljedećih DMA transfera se odvija na isti način kao što je opisano u prethodnom odjeljku, a upravljanje sklopovljem, tj. *blokom za propagaciju unazad*, se izvršava pristupanjem *upravljačkom registru* na način opisan u poglavlju 5.6. Funkcija ne vraća ništa, a završava kad se u DDR3 memoriju spremne korigirane težine svih slojeva neuronske mreže.

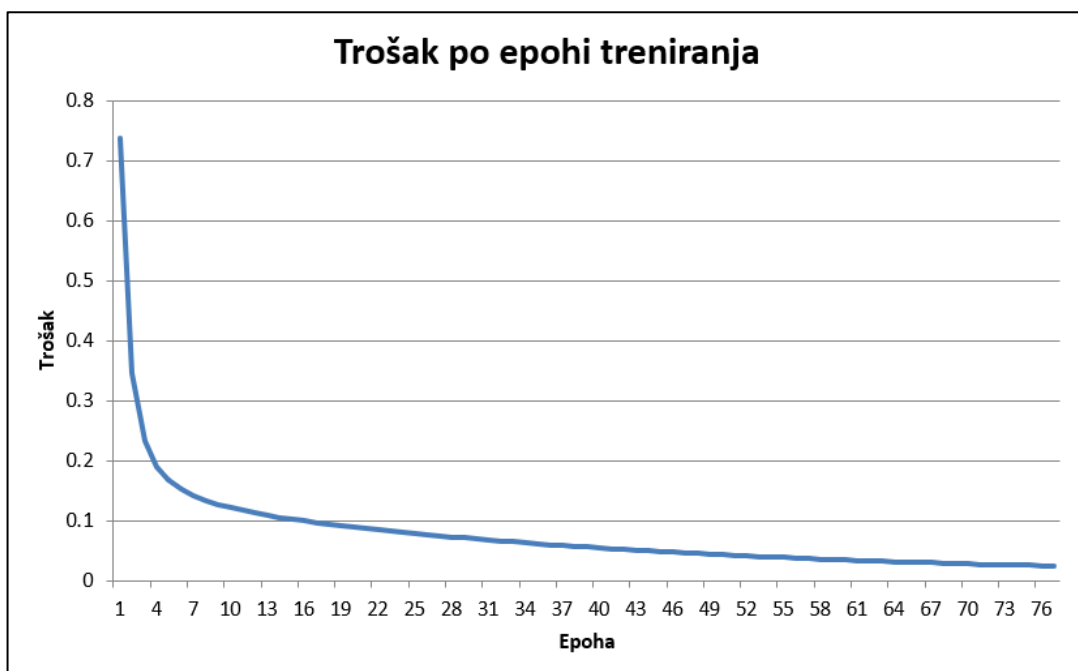
Tablica 5.15 - Ovisnost sloja neuronske mreže, podataka u DMA transferima i izvora gradijenata pogreške

SLOJ	DMA 0	DMA 1	DMA 2	IZVOR GRADIJENATA POGREŠAKA
izlazni sloj	težine ( <i>drugi skriveni sloj</i> → <i>izlazni sloj</i> )	izlazi neurona drugog skrivenog sloja	korigirane težine ( <i>drugi skriveni sloj</i> → <i>izlazni sloj</i> )	registar s Avalon sučeljem - pogreška
drugi skriveni sloj	težine ( <i>prvi skriveni sloj</i> → <i>drugi skriveni sloj</i> )	izlazi neurona prvog skrivenog sloja	korigirane težine ( <i>prvi skriveni sloj</i> → <i>drugi skriveni sloj</i> )	izlaz RAM-a 1
prvi skriveni sloj	težine ( <i>ulazni sloj</i> → <i>prvi skriveni sloj</i> )	piksli ulazne slike	korigirane težine ( <i>ulazni sloj</i> → <i>prvi skriveni sloj</i> )	izlaz RAM-a 2

## 6. REZULTATI MJERENJA

U ovome poglavlju dan je pregled rezultata mjerenja dobivenih nakon sinteze i implementacije sustava te nakon uspješnog treniranja neuronske mreže. U nastavku je ukratko opisan postupak treniranja mreže. Prvo je isprogramirana programibilna logika, a zatim procesorski sustav. Zatim su pomoću *Dump/Restore Data File* opcije Xilinx SDK razvojnog okruženja u memoriju upisani sljedeći podaci: skup težina inicijaliziranih „He“ metodom, skup slika za treniranje, skup referentnih podataka za treniranje, skup slika za testiranje i skup referentnih podataka za testiranje. Nakon unosa ulaznih podataka zadan je broj slika nad kojim će se mreža trenirati, broj slika nad kojim će se mreža testirati i koeficijent učenja te je pokrenuto treniranje mreže. Postupak treniranja je zaustavljen nakon što je postotak točnih prepoznavanja počeo konstantno padati, a zatim je u binarnu datoteku spremljen skup istreniranih težina nakon epohe s najvećim postotkom točnih prepoznavanja. Treniranje je provedeno dva puta nad različitom količinom slika u skupovima za treniranje i testiranje te s različitim koeficijentom učenja.

U prvom slučaju treniranje je obavljeno nad 5000 slika iz skupa za treniranje, a testiranje nad 1000 slika iz skupa za testiranje. Pritom je odabran koeficijent učenja 0,0003. Treniranje je završeno nakon 77 epoha, a vrijeme treniranja je 2 sata i 49 minuta. Postotak točnih prepoznavanja jest 91,1%. Slika 6.1 prikazuje trošak po epohi treniranja mreže. Sa slike je vidljivo da trošak naglo opada prve 4 epohe, a zatim polako konvergira nuli.



Slika 6.1 - Trošak po epohi treniranja (5000 trening slika, 1000 test slika)

Testiranjem prethodno istrenirane mreže nad 1000 slika iz skupa za testiranje, dobiveni su podaci iz kojih je kreirana konfuzijska matrica prikazana tablicom 6.1. Iz matrice se može zaključiti da mreža u 99% slučajeva točno prepoznaje znamenku „1“ iz čega slijedi da je najbolje istrenirana za prepoznavanje te znamenke, a u 86% slučajeva točno prepoznaje znamenku „3“ koju najčešće zamjenjuje sa znamenkom „5“ iz čega slijedi da je mreža najlošije istrenirana za prepoznavanje znamenke „3“.

Tablica 6.1 - Konfuzijska matrica (5000 trening slika, 1000 test slika)

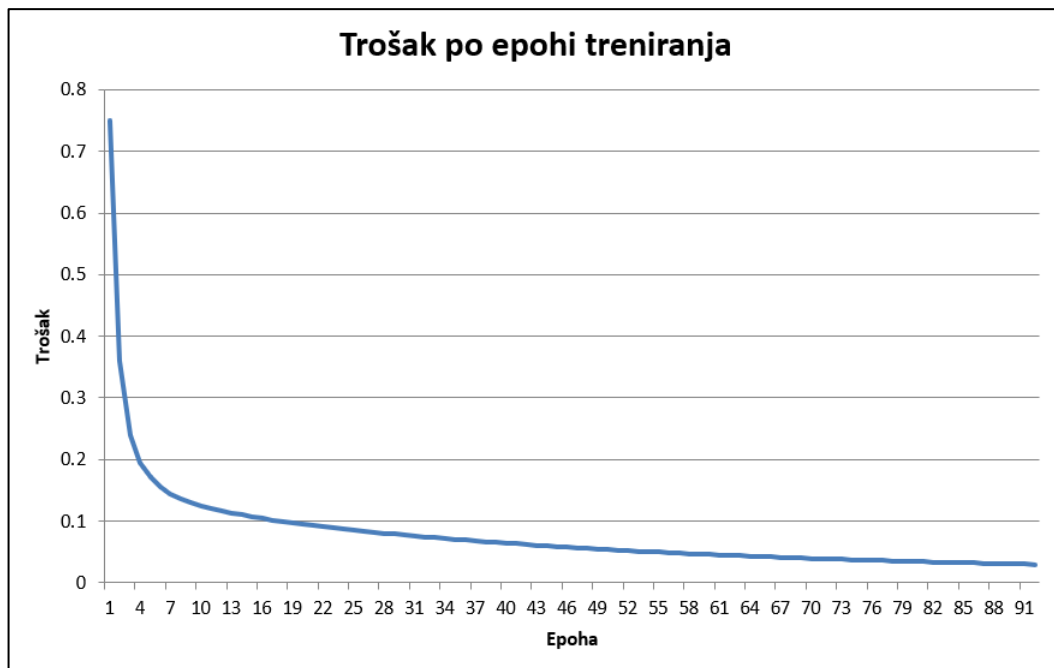
		PREPOZNATO										
		0	1	2	3	4	5	6	7	8	9	
OČEKIVANO	0	<b>0,95</b>	0,00	0,00	0,00	0,00	0,00	0,00	0,02	0,00	0,00	0,02
	1	0,00	<b>0,99</b>	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00
	2	0,00	0,01	<b>0,90</b>	0,02	0,01	0,01	0,00	0,03	0,03	0,03	0,01
	3	0,00	0,00	0,01	<b>0,86</b>	0,00	0,07	0,01	0,03	0,03	0,03	0,00
	4	0,00	0,01	0,01	0,00	<b>0,92</b>	0,00	0,02	0,00	0,00	0,00	0,05
	5	0,02	0,00	0,00	0,01	0,01	<b>0,89</b>	0,01	0,01	0,01	0,03	0,01
	6	0,03	0,00	0,00	0,00	0,01	0,01	<b>0,93</b>	0,01	0,01	0,00	0,00
	7	0,00	0,02	0,07	0,00	0,01	0,00	0,00	<b>0,87</b>	0,00	0,00	0,03
	8	0,00	0,00	0,01	0,03	0,02	0,02	0,00	0,01	<b>0,90</b>	0,00	0,00
	9	0,00	0,00	0,00	0,01	0,04	0,00	0,00	0,03	0,02	<b>0,89</b>	0,00

Iz skupa slika za testiranje slučajno je odabrana po jedna točno i jedna pogrešno prepoznata znamenka te su iste prikazane u tablici 6.2. Neke od znamenaka iz tablice teško je prepoznati čak i ljudskim okom dok se u slučaju nekih drugih znamenaka može zaključiti da sustav za prepoznavanje znamenaka ne funkcionira savršeno, tj. moguće je unaprijediti sustav s ciljem veće točnosti prepoznavanja.

Tablica 6.2 – Primjeri točnih i pogrešnih prepoznavanja (5000 trening slika, 1000 test slika)

TOČNO PREPOZNAVANJE		POGREŠNO PREPOZNAVANJE	
SLIKA	PREPOZNATA ZNAMENKA	SLIKA	PREPOZNATA ZNAMENKA
	0		6
	1		6
	2		7
	3		5
	4		9
	5		8
	6		0
	7		2
	8		3
	9		4

U drugom slučaju treniranje je obavljeno nad 7000 slika iz skupa za treniranje, a testiranje nad 1500 slika iz skupa za testiranje. Pritom je odabran koeficijent učenja 0,0002. Treniranje je završeno nakon 93 epohe, a vrijeme treniranja je 4 sata i 46 minuta. Postotak točnih prepoznavanja jest 91,4%. Slika 6.2 prikazuje trošak po epohi treniranja mreže. Sa slike je vidljivo da trošak naglo opada prvih 5 epoha, a zatim polako konvergira nuli.



Slika 6.2 - Trošak po epohi treniranja (7000 trening slika, 1500 test slika)

Testiranjem prethodno istrenirane mreže nad 1500 slika iz skupa za testiranje, dobiveni su podaci iz kojih je kreirana konfuzijska matrica prikazana tablicom 6.3. Iz matrice se može zaključiti da mreža u 99% slučajeva točno prepoznaje znamenku „1“ iz čega slijedi da je najbolje istrenirana za prepoznavanje te znamenke, a u 83% slučajeva točno prepoznaje znamenku „9“ koju najčešće zamjenjuje sa znamenkom 4 iz čega slijedi da je mreža najlošije istrenirana za prepoznavanje znamenke „9“.





















Tablica 6.3 - Konfuzijska matrica (7000 trening slika, 1500 test slika)

		PREPOZNATO									
		0	1	2	3	4	5	6	7	8	9
OČEKIVANO	0	<b>0,98</b>	0,00	0,00	0,01	0,00	0,00	0,01	0,00	0,00	0,01
	1	0,00	<b>0,99</b>	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00
	2	0,00	0,01	<b>0,91</b>	0,01	0,01	0,01	0,02	0,02	0,02	0,00
	3	0,00	0,00	0,03	<b>0,92</b>	0,00	0,03	0,01	0,01	0,01	0,00
	4	0,01	0,01	0,01	0,00	<b>0,95</b>	0,00	0,01	0,00	0,00	0,02
	5	0,01	0,00	0,00	0,04	0,01	<b>0,88</b>	0,02	0,01	0,03	0,01
	6	0,02	0,01	0,01	0,00	0,01	0,02	<b>0,92</b>	0,01	0,01	0,00
	7	0,00	0,02	0,07	0,00	0,01	0,00	0,00	<b>0,88</b>	0,00	0,02
	8	0,00	0,01	0,02	0,06	0,02	0,00	0,01	0,01	<b>0,86</b>	0,00
	9	0,01	0,00	0,00	0,02	0,08	0,00	0,00	0,06	0,01	<b>0,83</b>

Iz skupa slika za testiranje slučajno je odabrana po jedna točno i jedna pogrešno prepoznata znamenka te su iste prikazane u tablici 6.4. Kao i u prethodnom slučaju, neke od znamenaka iz tablice teško je klasificirati čak i ljudskim okom, a na temelju nekih drugih znamenaka zaključuje se da je sustav moguće unaprijediti s ciljem veće točnosti prepoznavanja znamenaka.



Tablica 6.4 - Primjeri točnih i pogrešnih prepoznavanja (7000 trening slika, 1500 test slika)

TOČNO PREPOZNAVANJE		POGREŠNO PREPOZNAVANJE	
SLIKA	PREPOZNATA ZNAMENKA	SLIKA	PREPOZNATA ZNAMENKA
	0		9
	1		6
	2		6
	3		2
	4		9
	5		3
	6		0
	7		2
	8		3
	9		4

Uz prethodno spomenute rezultate mjerenja, u nastavku su dani rezultati prikupljeni neposredno nakon sinteze i implementacije u Xilinx Vivado softverskom paketu. Zauzeto je: 10020 (57%) „lookup“ tablica, 7781 (22%) registara, 17 (21%) DSP blokova i 3 (5%) BRAM blokova. Maksimalna snaga razvojnog sustava iznosi 1,601 W, a maksimalna frekvencija rada sustava iznosi 100 MHz. Nakon uspješnog ispitivanja točnosti rada sustava izmjereno je vrijeme izvođenja algoritma propagacije unaprijed i algoritma propagacije unazad. Vrijeme izvršavanja algoritma propagacije unaprijed iznosi 7,7 ms, a algoritma propagacije unazad 16,3 ms. Teorijski broj perioda takta potrebnih za izvršavanje algoritma propagacije unaprijed iznosi 274080, a u stvarnoj implementaciji algoritma iznosi 772310. Također, teorijski broj perioda takta potrebnih

za izvršavanje algoritma propagacije unazad iznosi 289680, a u stvarnoj implementaciji algoritma iznosi 1637748. Do razlike dolazi zato što se u teorijski broj perioda takta ne uračunavaju: prazan hod u DMA transferima, broj perioda takta potreban da se izvrši upisivanje podatka u neki od registara s Avalon sučeljem te izvršavanje ostatka programa na procesoru.

U nastavku je dana usporedba rezultata ovog rada sa znanstvenim radovima [19], [20], [21] i [39]. Rezultati su prikazani u tablicama 6.5, 6.6 i 6.7. Svaki od prethodno navedenih znanstvenih radova predstavlja različitu implementaciju sustava za prepoznavanje znamenaka iz MNIST baze podataka. Rad [19] predstavlja implementaciju konvolucijske neuronske mreže na FPGA te je zbog strukture mreže koja je uspješnija u rješavanju problema prepoznavanja znamenaka, točnost prepoznavanja veća od točnosti u ovome radu. Također, treba naglasiti da sustav u radu [19] zauzima 6 puta više registara i 16 puta više DSP blokova što tu izvedbu čini složenijom. Vrijeme izvođenja propagacije unaprijed kraće je nego u ovom radu, a razlog tome je veća frekvencija rada i paralelizacija sustava te manji broj neurona u potpuno povezanim slojevima mreže. Rad [19], međutim, ne implementira algoritam propagacije unazad pa treniranje nije izvedivo na FPGA sustavu. Glavna prednost ovog rada u odnosu na rad [19] je implementiran algoritam propagacije unazad te manje zauzeće sklopovlja FPGA sustava. Rad [20] predstavlja implementaciju binarne konvolucijske neuronske mreže za prepoznavanje znamenaka. Prednost ovog tipa mreže je jednostavnost matematičkih operacija koje se izvode nad 1-bitnim binarnim brojevima, stoga aritmetičko-logičke jedinice zauzimaju malo resursa FPGA. Vrijeme izvođenja propagacije unaprijed u radu [20] manje je nego u ovom radu zbog paralelizacije sustava i manjeg broja neurona u potpuno povezanim slojevima, međutim, točnost prepoznavanja u radu [20] je manja u usporedbi s ovim radom jer se 1-bitnim binarnim brojem ne mogu dovoljno detaljno prikazati informacije u mreži. Također, rad [20] ne implementira algoritam propagacije unazad za razliku od ovog rada. Rad [21] predstavlja implementaciju umjetne neuronske mreže – višeslojni perceptron za prepoznavanje znamenaka. Predstavljene su dvije strukture mreže: dvoslojna i troslojna, a rezultati su dani za dvoslojnu mrežu čiji su parametri trenirani na razvojnoj ploči te za dvoslojnu i troslojnu mrežu čiji su parametri trenirani na računalu. Aritmetičko-logičke jedinice jednostavnije su nego u ovom radu jer su podaci 8-bitni s nepomičnim zarezom. Točnost prepoznavanja u radu [21] manja je nego u ovom radu u slučaju dvoslojne mreže, a veća u slučaju troslojne mreže. Vrijeme izvođenja propagacije unaprijed u ovom radu manje je nego u slučajevima u radu [21] gdje je treniranje obavljeno na računalu. Rad [21] implementira algoritam propagacije unazad u slučaju dvoslojne mreže, a vrijeme izvođenja treniranja nad 50000 slika za treniranje i 10000 slika za testiranje iznosi 3,8 sekundi, gdje u vrijeme izvođenja nije uračunato

vrijeme prijenosa podataka UART-om. Rad [39] predstavlja implementaciju umjetne neuronske mreže – višeslojni perceptron za prepoznavanje znamenki. Sustav iz prethodno spomenutog rada zauzima 4 puta više „lookup“ tablica, 2 puta više registara i 35 puta više DSP blokova što tu izvedbu čini složenijom. Sustav točnije i brže prepoznaje znamenke od sustava u ovome radu, ali mu je snaga 353 puta veća. Rad [39] ne implementira algoritam za propagaciju unazad.

Tablica 6.5 – Usporedba radova po zauzeću sklopova unutar FPGA

	<b>LUT</b>	<b>FF</b>	<b>DSP</b>	<b>LE (ALM)</b>	<b>BRAM</b>
<b>RAD [19]</b>	12588	48765	274	-	-
<b>RAD [20]</b>	-	-	-	-	-
<b>RAD [21] – 2 sloja, trening na ploči</b>	-	-	-	34000	-
<b>RAD [21] – 2 sloja, trening na računalu</b>	-	-	-	183*	-
<b>RAD [21] – 3 sloja</b>	-	-	-	428*	-
<b>RAD [39]</b>	44668	14274	604	-	-
<b>OVAJ RAD</b>	10020	7781	17	-	12 KB (3/60)

\*zauzeće logičkih elemenata po aritmetičko-logičkoj jedinici

Tablica 6.6 – Usporedba radova po snazi, točnosti prepoznavanja, frekvenciji rada sustava i širini i tipu podataka

	<b>SNAGA [W]</b>	<b>TOČNOST PREPOZNAVANJA [%]</b>	<b>FREKVENCIJA [MHz]</b>	<b>ŠIRINA I TIP PODATAKA</b>
<b>RAD [19]</b>	-	97,57	150	18-bitni s nepomičnim zarezom
<b>RAD [20]</b>	0,136	85	100	1 bit
<b>RAD [21] – 2 sloja, trening na ploči</b>	-	89	25	8-bitni s nepomičnim zarezom
<b>RAD [21] – 2 sloja, trening na računalu</b>	-	89,6	250	8-bitni s nepomičnim zarezom
<b>RAD [21] – 3 sloja</b>	-	92,7	60	8-bitni s nepomičnim zarezom
<b>RAD [39]</b>	566,313	93,25	100	32-bitni s pomičnim zarezom
<b>OVAJ RAD</b>	1,601	91,4	100	32-bitni s pomičnim zarezom

Tablica 6.7 – Usporedba radova po vremenu izvođenja propagacije unaprijed i unazad

	<b>VRIJEME IZVOĐENJA PROPAGACIJE UNAPRIJED [ms]</b>	<b>SUSTAV PODRŽAVA PROPAGACIJU UNAZAD</b>	<b>VRIJEME IZVOĐENJA PROPAGACIJE UNAZAD [ms]</b>
<b>RAD [19]</b>	0,0176	NE	-
<b>RAD [20]</b>	0,018	NE	-
<b>RAD [21] – 2 sloja, trening na ploči</b>	N/A	DA	N/A
<b>RAD [21] - 2 sloja, trening na računalu</b>	20	NE	-
<b>RAD [21] – 3 sloja</b>	150	NE	-
<b>RAD [39]</b>	0,00155	NE	-
<b>OVAJ RAD</b>	7,7	DA	16,3**

\*\*vrijeme izvođenja algoritma propagacije unazad (jedno podešavanje svih težina u mreži)

## 7. ZAKLJUČAK

U ovome je radu na temelju postojećih rješenja predloženo vlastito rješenje sustava za prepoznavanje znamenki. Sustav podrazumijeva četveroslojnu umjetnu neuronsku mrežu čiji je ulaz slika razlučivosti 28x28 piksela iz MNIST baze podataka ručno pisanih znamenki, a izlaz skup vjerojatnosti pojavljivanja pojedine znamenke na ulazu. Model neuronske mreže, uključujući algoritam propagacije unaprijed i algoritam propagacije unazad, prvo je implementiran u C programskom jeziku na računalu, a zatim je na temelju C implementacije osmišljen i implementiran u FPGA sustavu.

Sustav postiže zadovoljavajuće rezultate u vidu točnosti prepoznavanja znamenaka, zauzeća sklopovlja FPGA i brzine izvršavanja algoritma propagacije unaprijed i algoritma propagacije unazad. Za razliku od većine znanstvenih radova na ovu temu, ovaj rad implementira algoritam propagacije unazad te je treniranje neuronske mreže u potpunosti izvedeno na FPGA razvojnom sustavu. Računalo je korišteno jedino za potrebe programiranja FPGA i procesora te za unos parametara u DDR3 memoriju razvojnog sustava. Zauzeće sklopovlja FPGA manje je u odnosu na ostale znanstvene radove što ovu izvedbu čini jednostavnijom. Rad je moguće poboljšati u vidu povećanja brzine izvođenja propagacije unaprijed i propagacije unazad tako da se uvede paralelno izvođenje aritmetičkih operacija. Također, moguće je u neuronsku mrežu dodati konvolucijske i udruživačke slojeve ili implementirati neki drugi algoritam treniranja mreže što bi moglo rezultirati većom točnosti prepoznavanja znamenaka.

## LITERATURA

- [1] “Autopilot and Full Self-Driving Capability | Tesla.” <https://www.tesla.com/support/autopilot> (pristupljeno 13.04.2021.).
- [2] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.
- [3] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesus, *Neural Network Design 2nd Edition (2014)*. 2014.
- [4] “The mostly complete chart of Neural Networks, explained | by Andrew Tch | Towards Data Science.” <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464> (pristupljeno 13.04.2021.).
- [5] “ANN vs CNN vs RNN | Types of Neural Networks.” <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/> (pristupljeno 13.04.2021.).
- [6] “How does a Neural Network work intuitively in code? | by Steven Gong | Medium.” <https://medium.com/@gongster/how-does-a-neural-network-work-intuitively-in-code-f51f7b2c1e3f> (pristupljeno 13.04.2021.).
- [7] “Activation Functions | Fundamentals Of Deep Learning.” <https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/> (pristupljeno 14.04.2021.).
- [8] “Activation Functions in Neural Networks | by SAGAR SHARMA | Towards Data Science.” <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6> (pristupljeno 13.04.2021.).
- [9] “Logistic Regression — Detailed Overview | by Saishruthi Swaminathan | Towards Data Science.” <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc> (pristupljeno 13.04.2021.).
- [10] “Neural Networks: Feedforward and Backpropagation Explained.” <https://mlfromscratch.com/neural-networks-explained/#from-input-layer-to-hidden-layer> (pristupljeno 14.04.2021.).
- [11] “(2) What is meant by activation function? - Quora.” <https://www.quora.com/What-is-meant-by-activation-function> (pristupljeno 13.04.2021.).

- [12] “Convolutional Neural Network - an overview | ScienceDirect Topics.” <https://www.sciencedirect.com/topics/engineering/convolutional-neural-network> (pristupljeno 15.04.2021.).
- [13] D. Hugo and C. Perez, “A practical approach to Convolutional Neural Networks,” *Invert. Cern Sch. Comput.*, 2019.
- [14] “Building a Convolutional Neural Network (CNN) Model for Image classification. | by Shreyak | Becoming Human: Artificial Intelligence Magazine.” <https://becominghuman.ai/building-a-convolutional-neural-network-cnn-model-for-image-classification-116f77a7a236> (pristupljeno 14.04.2021.).
- [15] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018-Janua, no. August, pp. 1–6, 2018, doi: 10.1109/ICEngTechnol.2017.8308186.
- [16] “Understanding Convolution in Deep Learning — Tim Dettmers.” <https://timdettmers.com/2015/03/26/convolution-deep-learning/> (pristupljeno 15.04.2021.).
- [17] “MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges.” <http://yann.lecun.com/exdb/mnist/> (pristupljeno 30.04.2021.).
- [18] “How to Develop a CNN for MNIST Handwritten Digit Classification.” <https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/> (pristupljeno 20.04.2021.).
- [19] R. Xiao, J. Shi, and C. Zhang, “FPGA Implementation of CNN for Handwritten Digit Recognition,” *Proc. 2020 IEEE 4th Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2020*, no. Itnec, pp. 1128–1133, 2020, doi: 10.1109/ITNEC48623.2020.9085002.
- [20] P. Wang, J. Song, Y. Peng, and G. Liu, “Binarized Neural Network Based on FPGA to Realize Handwritten Digit Recognition,” *Proc. 2020 IEEE Int. Conf. Inf. Technol. Big Data Artif. Intell. ICIBA 2020*, no. Iciba, pp. 1204–1207, 2020, doi: 10.1109/ICIBA50161.2020.9276909.
- [21] J. Si, S. L. Harris, and E. Yfantis, “Neural Networks on an FPGA and Hardware-Friendly Activation Functions,” *J. Comput. Commun.*, vol. 08, no. 12, pp. 251–277, 2020, doi: 10.4236/jcc.2020.812021.



- [22] “Softmax Activation Function Explained | by Dario Radečić | Towards Data Science.”  
<https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60>  
 (pristupljeno 29.04.2021.).
- [23] “How to use Data Scaling Improve Deep Learning Model Stability and Performance.”  
<https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/> (pristupljeno 29.05.2021.).
- [24] “Understanding Backpropagation Algorithm | by Simeon Kostadinov | Towards Data Science.”  
<https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd> (pristupljeno 06.05.2021.).
- [25] “Understanding different Loss Functions for Neural Networks | by Shiva Verma | Towards Data Science.”  
<https://towardsdatascience.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718> (pristupljeno 06.05.2021.).
- [26] “How to Configure the Learning Rate When Training Deep Learning Neural Networks.”  
<https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/>  
 (pristupljeno 12.05.2021.).
- [27] “Zybo Reference Manual - Diligent Reference.”  
<https://reference.digilentinc.com/programmable-logic/zybo/reference-manual> (pristupljeno 27.05.2021.).
- [28] R. E. Haskell and D. M. Hanna, *Introduction to Digital Design Using Diligent FPGA Boards – Block Diagram / VHDL Examples*. 2009.
- [29] A. Ruede, “A Scientist’s Guide to FPGAs Content,” 2019.
- [30] “FPGA Fundamentals - NI.”  
<https://www.ni.com/en-rs/innovations/white-papers/08/fpga-fundamentals.html> (pristupljeno 01.06.2021.).
- [31] “What Is an FPGA and Why Is It a Big Deal? | Prowess Consulting.”  
<https://www.prowesscorp.com/what-is-fpga/> (pristupljeno 01.06.2021.).
- [32] “Xilinx Vivado - Wikipedia.”  
[https://en.wikipedia.org/wiki/Xilinx\\_Vivado](https://en.wikipedia.org/wiki/Xilinx_Vivado) (pristupljeno 07.06.2021.).
- [33] Xilinx, “AXI4-Lite IPIF v3.0 LogiCORE IP Product Guide Vivado Design Suite.”  
 Pristupljeno: 01.06.2021. [Online]. Dostupno: [www.xilinx.com](http://www.xilinx.com).
- [34] Xilinx, “AXI DMA v7.1 LogiCORE IP Product Guide Vivado Design Suite.” Pristupljeno:

- 27.05.2021. [Online]. Dostupno: [www.xilinx.com](http://www.xilinx.com).
- [35] Intel Corporation, “Avalon® Interface Specifications; Avalon® Interface Specifications.”
- [36] Xilinx, “AMM Slave Bridge v1.0 LogiCORE IP Product Guide Vivado Design Suite.”  
Pristupljeno: 08.06.2021. [Online]. Dostupno: [www.xilinx.com](http://www.xilinx.com).
- [37] “AMBA 4 AXI4-Stream Protocol Specification.”  
<https://developer.arm.com/documentation/ih0051/b> (pristupljeno 10.06.2021.).
- [38] Xilinx, “Floating-Point Operator v7.1 LogiCORE IP Product Guide Vivado Design Suite.”  
Pristupljeno: 10.06.2021. [Online]. Dostupno: [www.xilinx.com](http://www.xilinx.com).
- [39] I. Westby, X. Yang, T. Liu, and H. Xu, “FPGA acceleration on a multi-layer perceptron neural network for digit recognition,” *J. Supercomput.*, 2021, doi: 10.1007/s11227-021-03849-7.

## SAŽETAK

U okviru ovog rada opisana je implementacija sustava za prepoznavanje ručno pisanih znamenki u FPGA sustavu. Na početku su opisane vrste neuronskih mreža koje postižu najbolje rezultate u rješavanju problema prepoznavanja znamenaka, a zatim je dan pregled postojećih rješenja. Nakon toga je opisan model neuronske mreže te je dana teorija iza algoritma propagacije unaprijed i propagacije unazad. Zatim slijedi opis implementacije sustava koja podrazumijeva opis korištenih alata i tehnologija te opis svih komponenti sustava, sustava u cjelini i programske podrške. Na kraju rada predstavljeni su rezultati mjerenja te je provedena evaluacija.

Ključne riječi: FPGA, VHDL, MNIST, prepoznavanje ručno pisanih znamenki, ZYBO, Zynq.

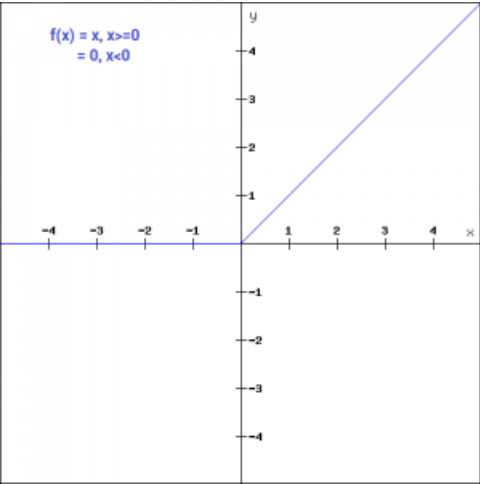
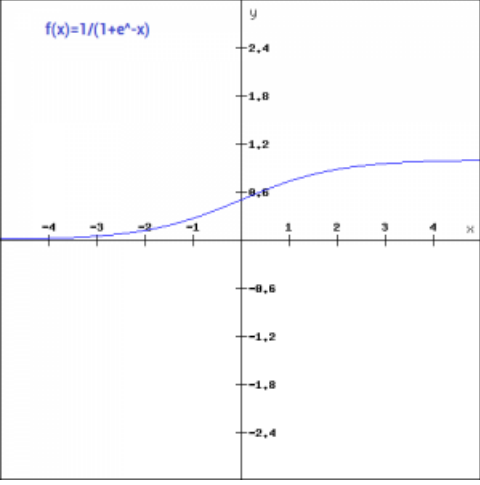
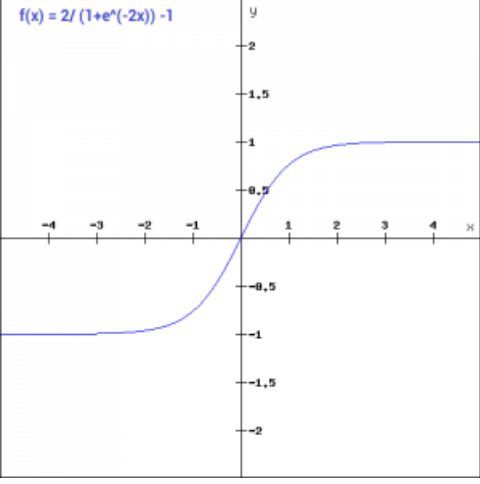
## **ABSTRACT**

This paper describes the implementation of a system for recognizing handwritten digits in an FPGA system. At the beginning, the types of neural networks that achieve the best results in solving the problem of digit recognition are described, and then an overview of existing solutions is given. After that, the neural network model is described and the theory behind the algorithm of forward propagation and backward propagation is given. This is followed by a description of the implementation of the system, which includes a description of the tools and technologies used and description of all components of the system, the system as whole and software. At the end of the paper, the measurement results were presented and an evaluation was performed.

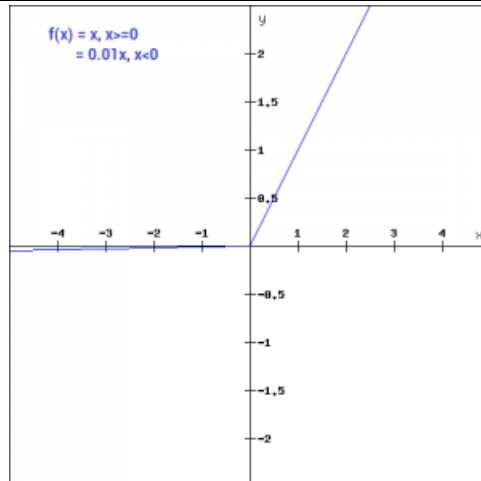
Keywords: FPGA, VHDL, MNIST, handwritten digit recognition, ZYBO, Zynq.

# PRILOG

Prilog P.1 – Tablica aktivacijskih funkcija

AKTIVACIJSKA FUNKCIJA	GRAF I FORMULA
ReLU	 <p><math>f(x) = x, x \geq 0</math> <math>= 0, x &lt; 0</math></p>
Sigmoid	 <p><math>f(x) = 1 / (1 + e^{-x})</math></p>
tanh	 <p><math>f(x) = 2 / (1 + e^{-2x}) - 1</math></p>

Leaky ReLU



Prilog P.2 – Funkcije troška

FUNKCIJA TROŠKA	IZRAZ
Srednja kvadratna pogreška	$C(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2$
Binarna unakrsna entropija	$C(y, \hat{y}) = -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$
Kategorička unakrsna entropija	$C(y, \hat{y}) = - \sum_{i=0}^{n-1} y_i \log(\hat{y}_i)$
Rijetka kategorička unakrsna entropija	$C(\hat{y}) = - \log(\hat{y}_i)$

## ŽIVOTOPIS

Matej Štajnbrikner rođen je 8. svibnja 1997. godine u Osijeku. Osnovnu je školu završio u Petrijevcima s odličnim uspjehom, a zatim upisuje Elektrotehničku i prometnu školu u Osijeku, smjer elektrotehničar, koju je također završio s odličnim uspjehom. Tijekom srednjoškolskog obrazovanja sudjelovao je na školskim i županijskim natjecanjima iz područja elektrotehnike i fizike. Po završetku srednjoškolskog obrazovanja upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Krajem 2018. godine postaje stipendist Instituta RT-RK Osijek te u prostorima tvrtke polaže FPGA školu i uspješno odrađuje završni rad na temu „FPGA implementacija 12-bitovnog procesora zasnovanog na Harvardskoj arhitekturi“. Na istome fakultetu upisuje diplomski studij, izborni blok računalno inženjerstvo te je tokom cijelog diplomskog studija stipendist Instituta RT-RK Osijek u kojemu se usavršava na polju dizajna digitalnih krugova u FPGA sustavima. Krajem 2020. godine, u istoj tvrtki počinje s izradom diplomskog rada na temu „Implementacija neuronske mreže za prepoznavanje znamenki u FPGA sustavu“.