Projekat iz predmeta

Projektovanje složenih digitalnih sistema

Autori:
Maša Jeličić EE36/2015
Marko Nikić EE86/2015

Mentor: prof Vuk Vranjković

13.2.2020

Sadržaj

0 Zadatak projekta **2**

1 Teorijska analiza **2**

2 Projektovanje IP jezgra **2**
2.1 Opis SystemC modela 2
2.2 Eliminacija naredbi ponavljanja 5
2.3 Definisanje interfejsa 7
2.4 Izrada ASMD dijagrama 8
2.5 Pisanje HDL modela glavnog modula 10
2.6 Pakovanje modula u IP jezgro 10

3 Simulaciona provera funkcionalnosti IP jezgra **11**

0 Zadatak projekta

- napraviti ekvivalentan ASMD dijagram
- implementirati algoritam u HDL jeziku
- oklopiti moduo kao AXI periferiju
- testirati osnovnu funkcionalnost modula

1 Teorijska analiza

Višeslojni perceptron (MLP) je klasifikator u obliku višeslojne mreže neurona. Neuron je komponenta sa više ulaza i jednim izlazom, koja svaki od ulaza množi sa odgovarajućom težinom, sabira dobijene proizvode i bias. Taj zbir se propagira kroz nelinearnu funkciju aktivacije. Izlaz funkcije aktivacije jeste izlaz neurona.

Neuroni u MLP-u su fully connected, što znači da je izlaz svakog neurona iz jednog sloja povezan na ulaz svakog neurona iz narednog sloja. Svaki od izlaznih neurona predstavlja jednu od kategorija klasifikacije: ako k-ti neuron u poslednjem sloju ima najveću izlaznu vrednost, ulazni vektor pripada k-toj kategoriji.

U našem slučaju MLP ima 3 sloja: ulazni, skriveni i izlazni. Funkcija aktivacije je LeakyReLu, data sledećom jednačinom

$$f(x) = \begin{cases} 0,001 \cdot x, & x < 0 \\ x, & x \geq 0. \end{cases} \tag{1}$$

Ulazni vektori su slike iz MNIST baze veličine 28x28 piksela, zbog čega ima 28x28=784 ulaznih neurona, a ulazi u neuron su vrednosti piksela od 0 do 1. U skrivenom sloju ima 30 neurona, a kako je u pitanju klasifikacija cifara 0-9, u poslednjem sloju ima 10 neurona.

2 Projektovanje IP jezgra

Pri projektovanju jezgra prvi korak je razvoj SystemC modela. Nakon toga je pristupljeno definisanju interfejsa i izradi ASMD dijagrama RTL modela. Konačno, napisan je VHDL kôd svih modula i testbench.

2.1 Opis SystemC modela

```
while (1)
{
    while (start == SC_LOGIC_0);
    start = SC_LOGIC_0;
    toggle = SC_LOGIC_1;
    p_out->write(toggle);
    for (int p = 0; p < NUM_INPUTS; p++)
    {
        while (!p_fifo->nb_read(fifo_tmp));
        image_v[p] = fifo_tmp;
        toggle = SC_LOGIC_0;
        p_out->write(toggle);
    }
    res_v.clear();
    for (unsigned int layer = 1; layer < LAYER_NUM; layer++) {
        for (unsigned int neuron = 0; neuron < neuron_array [layer];
            ↪ neuron++)
        {
            acc = 0;
            for ( int i=0; i < neuron_array [layer-1]; i++)
            {
                toggle = SC_LOGIC_1;
                p_out->write(toggle);
                while (!p_fifo->nb_read(fifo_tmp));
                acc += image_v[i] * fifo_tmp;
                toggle = SC_LOGIC_0;
                p_out->write(toggle);
            }
            toggle = SC_LOGIC_1;
            p_out->write(toggle);
            while (!p_fifo->nb_read(fifo_tmp));
            toggle = SC_LOGIC_0;
            p_out->write(toggle);
            acc += fifo_tmp;
            if (acc<0)
            acc*=0.001;
            res_v.push_back (acc);
        }
        if (layer != LAYER_NUM-1) {
            image_v.clear();
```

```
40              image_v = res_v;
41              res_v.clear();
42          }
43      }
44      max_res = res_v[0];
45      cl_num = 0;
46      for (int i = 1; i < 10; i++)
47      {
48          if (max_res < res_v[i])
49          {
50              max_res = res_v[i];
51              cl_num = (num_t)i;
52          }
53      }
54      toggle = SC_LOGIC_1;
55      p_out->write(toggle);
56      toggle = SC_LOGIC_0;
57      p_out->write(toggle);
58  }
```

Kôd koji implementira rad MLP-a se nalazi u beskonačnoj while petlji. Jedan prolazak kroz petlju predstavlja jednu klasifikaciju.

U liniji 3 algoritam čeka na postavljanje kontrolnog start signala na '1'. Nakon toga sledi prihvat slike preko fifo bafera, implementirano for petljom na liniji 7.

U liniji 14 se prazni vektor res_v koji sadrži rezultate klasifikacije iz prethodne iteracije petlje.

Na liniji 15 započinje klasifikacija. Prva for petlja predstavlja iteraciju po slojevima, a druga, unutrašnja predstavlja iteraciju po neuronima iz tog sloja. Za svaki neuron vrše se sledeće radnje: prihvataju se težine preko fifo bafera, množe se sa ulazima neurona sadržanim u image_v vektoru i na kraju se dodaje bias dopremljen preko fifo bafera. U liniji 34 implementirana je LeakyReLu funkcija. Rezultati trenutnog sloja se smeštaju u vektor res_v.

Broj neurona u svakom sloju se nalazi u nizu neuron_array. U našem slučaju on ima vrednosti {784, 30, 10}.

Na liniji 44 počinje pronalaženje maksimalne izlazne vrednosti poslednjeg sloja neurona. Indeks maksimuma predstavlja klasifikovanu cifru cl_num.

Signal prekida se aktivira svaki put kada MLP zahteva nove podatke (sliku, težine neurona ili bias neurona), linije 5, 21 i 28, i kada se dobije rezultat klasifikacije, linija 54.

2.2 Eliminacija naredbi ponavljanja

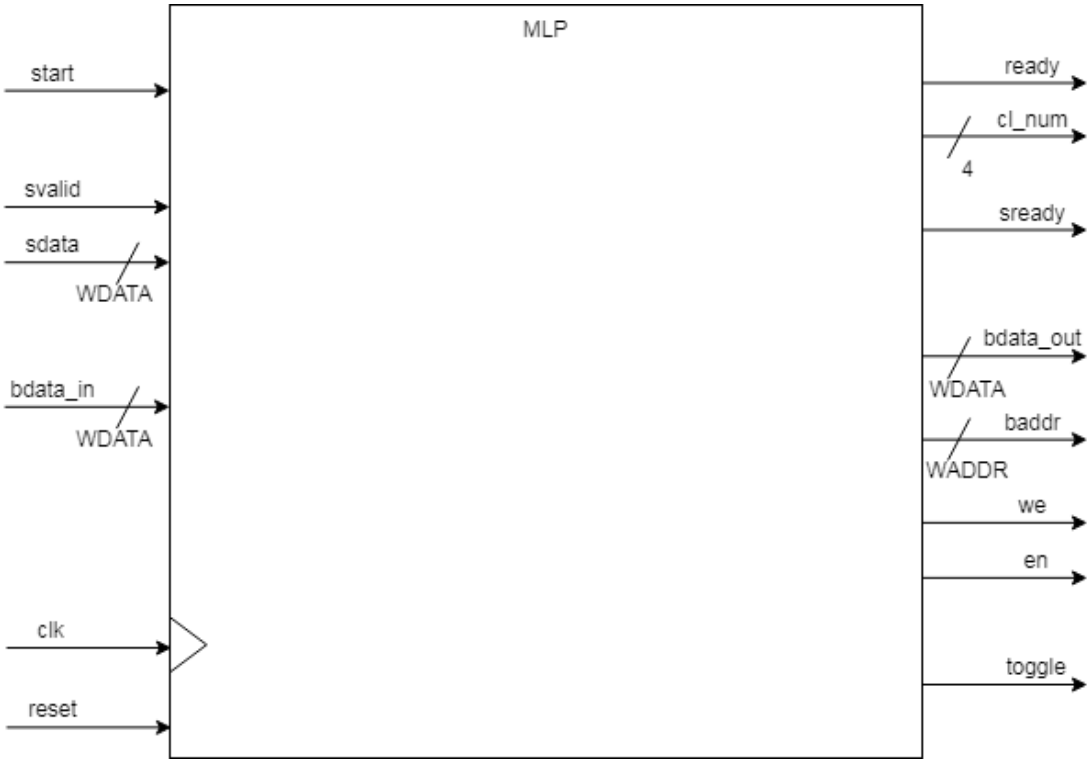
Za generisanje ASMD dijagrama neophodno je u izvornom kodu eliminisati naredbe ponavljanja i zameniti ih *goto* naredbama.

```
idle:
    if (start != SC_LOGIC_0)
        goto start_state;
    else
        goto idle;
start_state:
    start = SC_LOGIC_0;
    toggle = SC_LOGIC_1;
    p_out->write(toggle);
    int p = 0;
wait_pixel:
    if (p_fifo->nb_read(fifo_tmp))
        goto load_pixel;
    else
        goto wait_pixel;
load_pixel:
    image_v[p] = fifo_tmp;
    toggle = SC_LOGIC_0;
    p_out->write(toggle);
    p ++;
    if (p < NUM_INPUTS)
        goto wait_pixel;
    else {
        res_v.clear();
        int layer = 1;
        goto layer_state;
    }
layer_state:
    unsigned int neuron = 0;
neuron_state:
    acc = 0;
    int i = 0;
synapse_state:
    toggle = SC_LOGIC_1;
    p_out->write(toggle);
wait_weight:
    if (p_fifo->nb_read(fifo_tmp))
        goto load_weight;
    else
        goto wait_weight;
load_weight:
    acc += image_v[i] * fifo_tmp;
    toggle = SC_LOGIC_0;
    p_out->write(toggle);
    i ++;
    if (i < neuron_array[layer-1])
        goto synapse_state;
    else {
        toggle = SC_LOGIC_1;
        p_out->write(toggle);
        goto wait_bias;
    }
wait_bias:
    if (p_fifo->nb_read(fifo_tmp))
        goto load_bias;
    else
        goto wait_bias;
load_bias:
    toggle = SC_LOGIC_0;
    p_out->write(toggle);
    acc += fifo_tmp;
    if (acc < 0)
        acc *= 0.001;
    res_v.push_back (acc);
    neuron ++;
    if (neuron < neuron_array[layer])
        goto neuron_state;
    else
        goto cont;
cont:
    if (layer != LAYER_NUM-1) {
        image_v.clear();
        image_v = res_v;
        res_v.clear();
    }
    layer ++;
    if (layer < LAYER_NUM)
        goto layer_state;
    else
        goto cont_1;
cont_1:
    max_res = res_v[0];
    cl_num = 0;
    int j = 1;
find_res:
    if (max_res < res_v[j])
    {
        max_res = res_v[j];
        cl_num = (num_t)j;
    }
    j++;
    if (j < 10)
        goto find_res;
    else {
        toggle = SC_LOGIC_1;
        p_out->write(toggle);
        toggle = SC_LOGIC_0;
        p_out->write(toggle);
        goto idle;
    }
```

2.3 Definisanje interfejsa

Interfejs se sastoji od:

- sinhronizacionog i reset porta *clk* i *reset*
- kontrolnog porta *start*, čijim setovanjem počinje klasifikacija
- statusnog porta *ready* koji MLP aktivira kada je spreman za novu klasifikaciju
- izlaznog porta *cl_num* koji predstavlja rezultat klasifikacije
- portova *sdata*, *svalid* i *sready*, za komunikaciju sa fifo kanalom
- portova *bdata_in*, *bdata_out*, *baddr*, *en* i *we* za komunikaciju sa blok RAM-om
- izlaznog porta *toggle* koji MLP aktivira da označi kraj klasifikacije



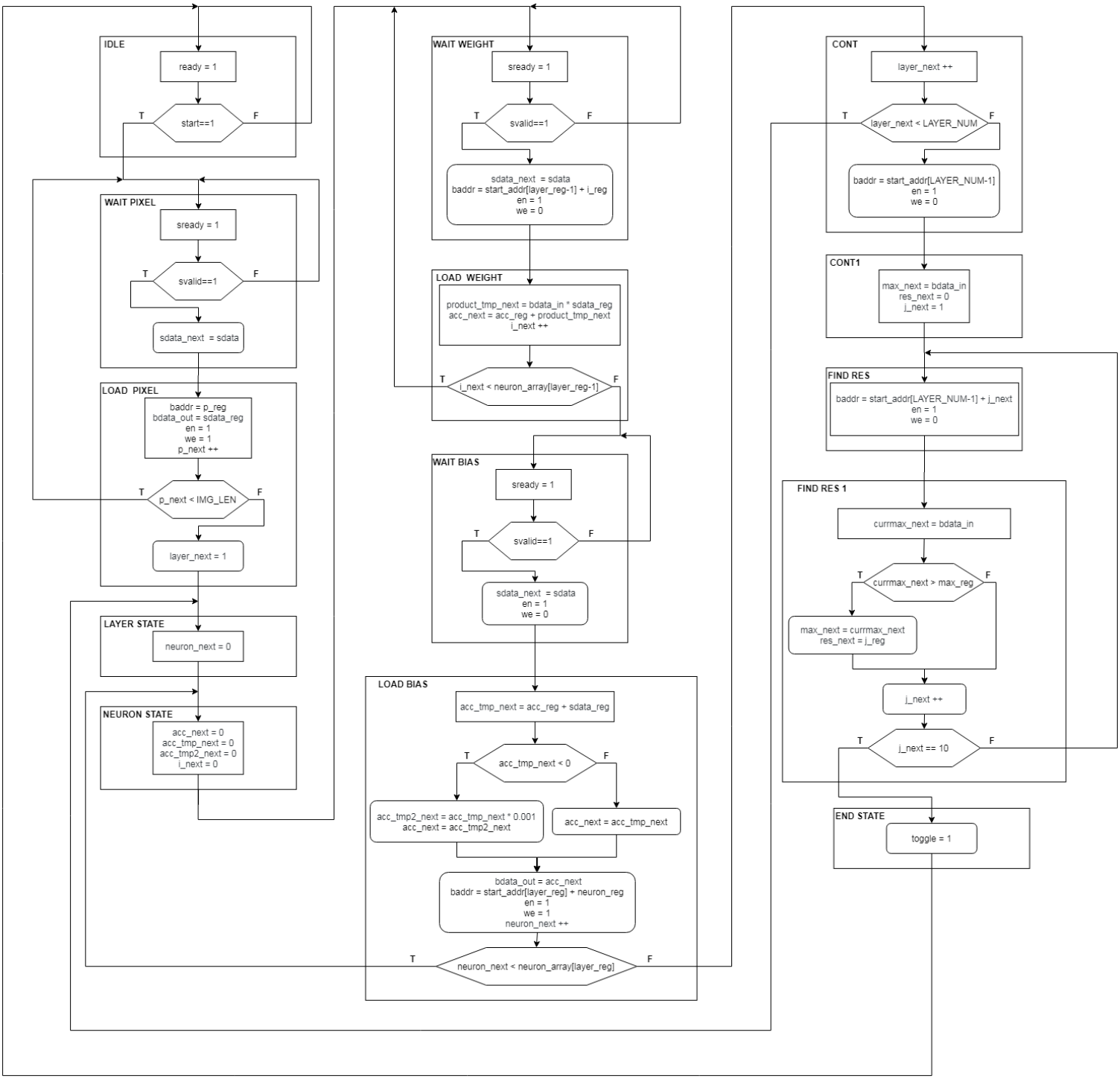
Slika 1: Interfejs MLP modula

2.4 Izrada ASMD dijagrama

Nakon što su naredbe ponavljanja eliminisane i interfejs definisan, izrađuje se ASMD dijagram koji opisuje rad kontrolne jedinice i datapath modula. Dijagram je dat na slici 2. Načinjene su izvesne izmene u odnosu na SystemC model, i to u pogledu: stanja upravljačke jedinice i trenutka aktiviranja *toggle* signala.

Toggle se aktivira samo po završetku klasifikacije, a ne i pri potraživanju podataka kao u SystemC modelu. Ova promena je izvršena zato što unapred definisani parametri poput broja neurona i broja slojeva, zajedno sa AXI Stream protokolom, pružaju sasvim dovoljno informacija za uspešan transfer podataka.

Zbog izmene *toggle* signala, stanje (labela u kodu sa goto naredbama) `start_state` je postalo suvišno, pa je ono uklonjeno, a dodato je stanje `end_state` u kom se aktivira *toggle*. Zbog čitanja iz blok RAM-a pri traženju maksimuma u izlaznom sloju bilo je potrebno dodati stanje `find_res_1` .



Slika 2: ASMD dijagram MLP modula

2.5 Pisanje HDL modela glavnog modula

Sada imamo sve informacije potrebne za pisanje HDL koda. VHDL kôd dat u prilogu strogo prati ASMD dijagram i rađen je u dvoprocesnom maniru, gde jedan proces opisuje registarsku logiku, a drugi kombinacionu. Izvršena je kvantizacija LeakyReLu parametra na 16 bita, 4 za ceo i 12 za razlomljeni deo. Kvantizacija menja vrednost parametra sa 0.001 na 0.0009765625, što ne degradira performanse sistema. Broj slojeva LAYER_NUM je dat kao generic, a broj neurona u svakom sloju je dat u constant nizu neuron_array.

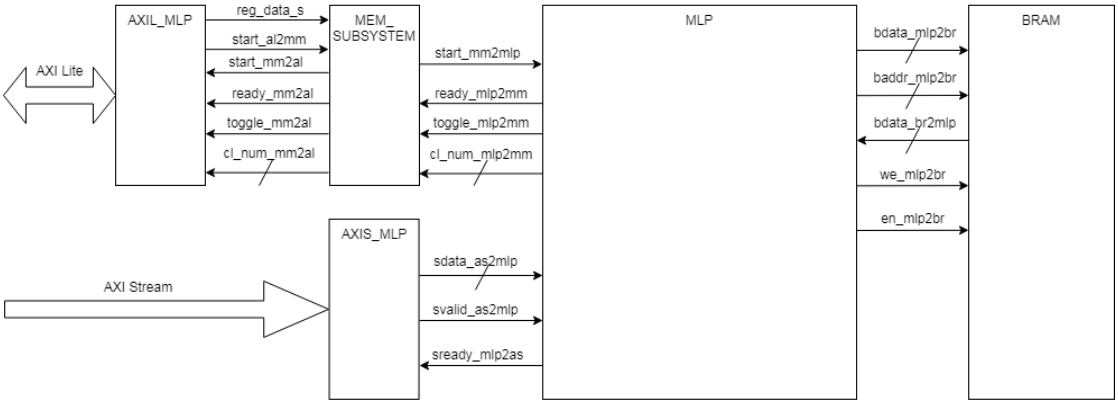
2.6 Pakovanje modula u IP jezgro

Radi lakše integracije potrebno je da moduli budu oklopljeni standardnim interfejsom. Odabran je AXI interfejs, i to varijante AXI Stream i AXI Lite. Za komunikaciju AXI interfejsom bilo je potrebno uvesti dodatne module:

- MEM_SUBSYSTEM, koji čine 4 registra u kojima se čuvaju vrednosti signala portova *start*, *ready*, *toggle* i *cl_num*
- AXIL_MLP, koji je povezan sa MEM_SUBSYSTEM-om i omogućava komunikaciju sa njegovim registrima preko AXI Lite protokola
- AXIS_MLP, koji prilagođava AXI Stream interfejs portovima za prijem podataka MLP modula (*sdata, svalid, sready*).

BRAM je dvopristupni RAM u kom se čuvaju ulazna slika, rezultati skrivenog i izlaznog sloja. Jedan pristup je direktno povezan sa MLP modulom, a drugi je onemogućen.

Na slici 3 je dat blok dijagram svih modula koji čine IP jezgro.



Slika 3: Blok dijagram IP jezgra

3 Simulaciona provera funkcionalnosti IP jezgra

Testbench je pisan u VHDL-u. Prvo se resetuje IP jezgro, nakon čega se pokreće klasifikacija 100 slika. Slike i parametri dati su u binarnom zapisu u odgovarajućim fajlovima koje testbench iščitava. Od 100 slika 98 je ispravno klasifikovano.

Pri pokretanju simulacije treba proveriti da li su putanje do fajlova slika i parametara ispravno navedene. Ako je potrebno, izmeniti ih u fajlu tb.vhd čiji je kôd dat u prilogu.