W3: Hardware implementation of rule learning procedures

W3-2: Implementation of basic rule learning procedures in hardware

Selection of the set of procedures for implementation in hardware (W3-2-1)

Sažetak

Analizirana je rutina *Construct feature*, i njena podrutina *Count coverage*, koja je probnom paralelizacijom u softveru identificirana kao najizgledniji kandidat za izvedbu u sklopovlju. Uz nju u analizu su uključene i tri dodatne rutine – *Eliminate covered*, *Update total coverage* i *Update OOB matrix* – koje bi kao ulaz (i izlaz) imale podatke pohranjene u memoriji spojenoj na FPGA. Iz ulaznih i izlaznih parametara rutina, te operacija koje se izvršavaju nad podacima, specificirane su tri procedure koje će se implementirati u sklopovlju prema modelu podatkovnog toka: *Count conditional*, *Increment conditional* i *Logic simple*. Navedene procedure zajedno pokrivaju sve operacije koje se vrše nad podacima u analiziranim rutinama (prebrojavanje, uvećavanje za 1, logička funkcija, usporedba, itd.), te će se njihovim povezivanjem u sklopovlju replicirati rad svake analizirane rutine.

Uvod

U prijašnjim aktivnostima projekta identificirane su četiri rutine kao potencijalni kandidati za implementaciju u FPGA sklopovlju. Te četiri rutine dodatno su analizirane, a tri od njih su paralelizirane u svrhu ocijene mogućeg dobitka izvedbom u sklopovlju. Kao najizglednija prepoznata je rutina *Construct feature*, čijom su paralelizacijom dobiveni dobri rezultati (ubrzanje konstrukcije jednog pravila do 4,146× na 6 dretvi).

U ovoj aktivnosti specificira se skup procedura koji će u narednim aktivnostima biti implementirani u FPGA sklopovlju. Polazište je rutina *Construct feature*. Ona se u ovoj aktivnosti dodatno analizira s ciljem identifikacije dijelova pogodnih za preformulaciju prema modelu toka podataka (*dataflow*). Prema saznanjima dobitnih iz analize specificiraju se rutine za implementaciju u FPGA sklopovlju.

Ključne rutine u programskoj implementaciji

Construct feature

Rutinu *Construct feature* čini petlja *LOOP4*, u kojoj se ispituju pojedini slučajno odabrani atributi. Broj iteracija petlje ograničen je s dvije vrijednosti:

- *mflag* – broj uspješno ispitanih značajki,
- *maxFeat* – maksimalni broj pokušaja ispitivanja značajki (iznosi 20*mflag).

Petlja završava kada je uspješno ispitano bar *mflag* značajki, te kada je napravljeno *maxFeat* iteracija. Ukoliko ja nakon *maxFeat* iteracija ispitano manje od *mflag* značajki, izgradnja pravila zaustavlja se i proglašava neuspješnom.

Unutar petlje:

- slučajno se odabere atribut,

- poziva se funkcija *construct_feature* koja atribut prima kao parametar.

Unutar funkcije *construct_feature*:

- ispita se atribut na kontradikciju - *wex* i *wey* su u kontradikciji ako ih se navedenim atributom ne može razlikovati (vrijednosti atributa za *wex* i *wey* su iste/bliske),
- ako su *wex* i *wey* u kontradikciji, funkcija završava bez značajke,
- inače:
    - stvara se značajka (*feature*),
    - ispita se pokrivenost skupa tom značajkom (rutina *Count coverage*),
    - izračuna se kvaliteta značajke

Ova rutina sama po sebi nije pogodna za izvedbu u sklopovlju. No, pod-rutina *Count coverage,* koja je njen dio, jest pogodna.

Count coverage

Rutina *Count coverage* (ispitivanje pokrivenosti skupa određenom značajkom) sastoji se od petlje koja iterirajući kroz sve primjere iz skupa za učenje:

- prebrojava pozitivne primjere nepoznate vrijednosti atributa,
- prebrojava pozitivne primjere (poznate vrijednosti atributa) koji su (neispravno) eliminirani
- prebrojava negativne primjere (poznate vrijednosti atributa) koji su (ispravno) eliminirani

Ova rutina vremenski je najintenzivniji proces koji se izvršava kao dio rutine *Construct feature*.

Rutina kao ulaz uzima četiri polja (vektora), te vraća dva skalara. Izvršavaju se logičke operacije, usporedba, te uvjetno brojanje (*count*).

Ostale rutine

Budući da opetovani prijenos podataka između glavne memorije računala u memoriju FPGA u pravilu ima negativne posljedice na performanse programa, prijenos skupa za učenje trebao bi se provesti samo jedanput, na početku programa. Stoga je praktično implementirati u sklopovlju i sve rutine koje kao ulaz prihvaćaju skup za učenje, ili neko pomoćno polje koje se koristi u algoritmu (npr. polje koje označava odabrane pozitivne i negativne primjere).

Eliminate covered

Ovu rutinu čini petlja koja iterira kroz skup za učenje i pri tome:

- označava kao eliminirane sve pozitivne primjere koji imaju nepoznatu vrijednost atributa određenog značajkom,
- označava kao eliminirane sve pozitivne i negativne primjere za koje odabrana značajka evaluira u logičku neistinu,
- prebrojava preostale (ne-eliminirane) negativne primjere.

Rutina kao ulaz uzima četiri polja (vektora), te modificira tri polja (vraća tri nova vektora) i vraća jedan skalar. Izvršavaju se usporedba, uvjetno brojanje (*count*), uvjetno uvećavanje za 1 (inkrement), te logička funkcija.

Update total coverage

Ova rutina ispituje pokrivenost značajkom svih pozitivnih primjera skupa za učenje. Pokrivenost svakog primjera pohranjuje se u posebnom polju. Rutinu čine dvije petlje. Prva petlja iterira kroz sve primjere skupa za učenje i pri tome:

- prebrojava pozitivne primjere koji su (ispravno) pokriveni značajkom,
- uvećava pokrivenost primjera koji su (ispravno) pokriveni značajkom (modifikacija sadržaja ulaznog polja),
- prebrojava negativne primjere koji su (neispravno) pokriveni značajkom,

Rutina kao ulaz uzima tri polja (vektora), te jednom od njih mijenja sadržaj (vraća novi vektor). Izvršavaju se usporedba, uvjetno brojanje, te uvjetno uvećavanje za 1 (inkrement).

Update OOB matrix

Ova rutina ažurira *out-of-bag* matricu, a čine ju dvije petlje. Prva petlja iterira kroz skup za učenje i za *out-of-bag* primjere (primjere isključene iz skupa za učenje jednog pravila):

- prebroji ih,
- uveća za jedan elemente u stupcu koji odgovara klasi *wex,*
- prebroji primjere koji su iste klase kao *wex*.

Rutina kao ulaz uzima četiri polja (vektora), jednom od njih mijenja sadržaj (vraća novi vektor) i vraća četiri skalara. U rutini se izvršavaju usporedba, uvjetno brojanje (*count*), te uvjetno uvećavanje za 1 (inkrement).

Specifikacija procedura za implementaciju u sklopovlju

Navedene rutine imaju zajedničke dva procesa: uvjetno prebrojavanje i uvjetna inkrementacija. Ostale operacije (usporedba, logičke operacije) izvršavaju se sa svrhom određivanja uvjeta pod kojim se izvršavaju glavne operacije – brojanje i inkrementacija. Jedna rutina (*Eliminate covered*) izvršava logičke operacije nad vektorima, te daje nove vektore kao rezultat.

Count conditional

Procedura *Count conditional* implementira operaciju uvjetnog prebrojavanja, koja je zajednička za sve rutine.

- Ulazi:
  - 4 vektora
  - 4 skalara
  - Parametar: izbor jedne od 8 funkcija koje definiraju uvjete za prebrojavanje
- Izlazi:
  - 4 skalara

Procedura u sklopovlju, ovisno o funkciji, može koristiti i manje od 4 ulazna vektora, i manje od 4 skalara. Također nije nužno vratiti vrijednost za sva 4 izlazna skalara. Koliko ulaznih vektora i skalara se koristi, te koliko se izlaznih skalara dobiva kao rezultat, ovisno je o odabranoj funkciji. Izvedene funkcije moraju biti definirane tako da pokrivaju sve slučajeve korištenja prisutne u opisanim rutinama.

Increment conditional

Procedura *Increment conditional* implementira operaciju uvjetne inkrementacije. Ta je procedura zajednička za rutine *Update total coverage* i *Update OOB matrix*.

- Ulazi:
  - 4 vektora
  - 2 skalara

    - Parametar: izbor jedne od 8 funkcija koje definiraju uvjete za inkrement
- Izlazi:
    - 4 vektora

Izvede funkcije moraju biti definirate tako da pokrivaju sve slučajeve korištenja prisutne u navedenim rutinama. Pojedine funkcije ne moraju koristiti sve ulazne vektore i skalare, niti definirati sve izlazne vektore.

Logic simple

Procedura *Logic simple* implementira jednostavne logičke funkcije s 4 ulaza i 4 izlaza. Logičke funkcije primjenjuju se na svakom elementu ulaznih vektora i njihovi rezultati čine izlazne vektore. Ova je procedura centralni dio rutine *Eliminate covered*.

- Ulazi:
    - 4 vektora
    - 2 skalara
    - Parametar: izbor jedne od 8 logičkih funkcija
- Izlazi:
    - 4 vektora (rezultat logičke funkcije između vektora)

Izvedene funkcije moraju biti definirane tako da pokrivaju sve slučajeve korištenja prisutne u opisanim rutinama. Pojedine logičke funkcije ne moraju koristiti sva 4 ulazna vektora, niti definirati sve izlaze.