W3: Hardware implementation of rule learning procedures

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

# A6.3 Implementation in hardware (M11-M13)

Tri jezgre (engl. *kernel*) specificirane u A6.2 implementirane su u sklopovlju. Specifikacije jezgara nadopunjene su detaljima koji pokrivaju konkretne primjene.

## Jezgra *Count conditional*

Jezgra *Count conditional* izvršava prebrojavanje pojava određenih vrijednosti, odnosno kombinacija vrijednosti, u ulaznim vektorima. Jezgra ima slijedeće četiri primjene:

* *Count coverage*
* *Eliminate covered*
* *Update total coverage*
* *Update OOB matrix*

**Ulazi jezgre:**

* Vektori
  + sIn[0..3] : 32-bit numeric
* Skalari
  + useCase : 3-bit integer : primjena (do osam mogućnosti, iskorišteno četiri)
  + cmpVal : 32-bit numeric : vrijednost za usporedbu (iz *feature*-a) / klasa
  + cmpFn : 2-bit integer : funkcija za usporedbu (iz *feature*-a)

**Izlazi jezgre:**

* Skalari
  + count[0..3] : 64-bit integer

**Primjene:**

*Eliminate covered*

* Ulazi
  + sIn[0] ⇐ selN
  + sIn[1..3] ⇐ NULL
* Izlazi
  + count[0] ⇒ remCount
  + cound[1..3] ⇒ NULL
* Funkcije
  + count[0]++ ← sIn[0] != 0

*Count coverage*

* Ulazi
  + sIn[0] ⇐ selN
  + sIn[1] ⇐ selP
  + sIn[2] ⇐ eU
  + sIn[3] ⇐ e
* Izlazi
  + count[0] ⇒ mm
  + count[1] ⇒ NULL
  + count[2] ⇒ m1
  + count[3] ⇒ m2
* Funkcije
  + count[0]++ ← sIn[2] == 0 && sIn[0] !=0 && !cmp
  + count[2]++ ← sIn[2] != 0 && sIn[1] !=0
  + count[3]++ ← sIn[2] == 0 && sIn[1] !=0 && !cmp
  + case cmpFn
    - f0 : cmp = sIn[3] > cmpVal /\* float > \*/
    - f1 : cmp = !(sIn[3] > cmpVal) /\* float <= \*/
    - f2 : cmp = sIn[3] == cmpVal /\* int == \*/
    - f3 : cmp = !(sIn[3] == cmpVal) /\* int != \*/

*Update total coverage*

* Ulazi
  + sIn[0] ⇐ atrue
  + sIn[1,2] ⇐ NULL
  + sIn[3] ⇐ cls
* Izlazi
  + count[0] ⇒ k4
  + count[1] ⇒ kkk
  + count[2,3] ⇒ NULL
* Funkcije
  + count[0]++ ← sIn[0] != 0 && sin[3] != cmpVal
  + count[1]++ ← sIn[0] != 0 && sIn[3] == cmpVal

*Update OOB matrix*

* Ulazi
  + sIn[0] ⇐ atrue
  + sIn[1] ⇐ selected
  + sIn[2] ⇐ NULL
  + sIn[3] ⇐ cls
* Izlazi
  + count[0] ⇒ k5
  + count[1] ⇒ k6
  + count[2] ⇒ k
  + count[3] ⇒ kk
* Funkcije
  + count[0]++ ← sIn[0] != 0
  + count[1]++ ← sIn[0] != 0 && sIn[3] == cmpVal
  + count[2]++ ← sIn[0] != 0 && sIn[1] == 0
  + count[3]++ ← sIn[0] != 0 && sIn[1] == 0 && sIn[3] == cmpVal

## Jezgra *Logic simple*

Ova jezgra izvršava logičke funkcije na elementima ulaznih vektora, te time formira nove vrijednosti vektora. Jezgra ima samo jednu primjenu, ali podijeljenu u dva slučaja: *training* i *test*.

**Ulazi jezgre:**

* Vektori
  + sIn[0..3] : boolean (1-bit), cast from 32-bit integer when read from memory
  + sIn[4] : 32-bit numeric
* Skalari
  + useCase : 2-bit integer : primjena (do četiri mogućnosti, iskorištene dvije)
  + cmpVal : 32-bit numeric : vrijednost za usporedbu (iz *feature*-a) / klasa
  + cmpFn : 2-bit integer : funkcija za usporedbu (iz *feature*-a)

Izlazi jezgre:

* Vektori:
  + sOut[0..2] : boolean (1-bit), cast to 32-bit integer for writing to memory

**Primjene:**

Slučaj A - *training*

* Ulazi
  + sIn[0] ⇐ atrue
  + sIn[1] ⇐ selP
  + sIn[2] ⇐ selN
  + sIn[3] ⇐ eU
  + sIn[4] ⇐ e
* Izlazi
  + sOut[0] ⇒ atrue’
  + sOut[1] ⇒ selP’
  + sOut[2] ⇒ selN’
* Funkcije
  + sOut[0] ⇐ sIn[0] && !sIn[3] && cmp
  + sOut[1] ⇐ sIn[1] && !sIn[3] && cmp
  + sOut[2] ⇐ sIn[2] && (sIn[3] || cmp)

Slučaj B - *test*

* Ulazi
  + sIn[0] ⇐ atrue
  + sIn[1,2] ⇐ NULL
  + sIn[3] ⇐ eU
  + sIn[4] ⇐ e
* Izlazi
  + sOut[0] ⇒ atrue’
  + sOut[1,2] ⇒ NULL
* Funkcije
  + sOut[0] ⇐ sIn[0] && !sIn[3] && cmp

Zajedničko za slučaj A i B:

* Funkcije
  + case cmpFn
    - f0 : cmp = sIn[4] > cmpVal /\* float > \*/
    - f1 : cmp = !(sIn[4] > cmpVal) /\* float <= \*/
    - f2 : cmp = sIn[4] == cmpVal /\* int == \*/
    - f3 : cmp = !(sIn[4] == cmpVal) /\* int != \*/

## Jezgra *Incremenet conditional*

Ova jezgra izvršava uvećavanje-za-jedan elemenata vektora koji ispunjavaju određene uvjete. uvjeti su određeni kombinacijama vrijednosti elemeanata vektora. Jezra ima dvije primjene:

* *Update total coverage*
* *Update OOB matrix*, s dva slučaja

**Ulazi jezgre:**

Vektori

* sIn[0..2] : 32-bit integer

Skalari

* useCase : 2-bit integer : primjena (do četiri mogućnosti)
* clsWex : 32-bit integer

**Izlazi jezgre:**

Vektori

* sOut : 32-bit integer

**Primjene:**

*Update total coverage*

* Ulazi
  + sIn[0] ⇐ atrue
  + sIn[1] ⇐ cls
  + sIn[2] ⇐ totco
* Izlazi
  + sOut ⇒ totco’
* Funkcije
  + sOut ⇐ sIn[0] != 0 && sIn[1] == clsWex ? sIn[2]++ : sIn[2]

*Update OOB matrix*, slučaj A

* Ulazi
  + sIn[0] ⇐ atrue
  + sIn[1] ⇐ sel
  + sIn[2] ⇐ oob
* Izlazi
  + sOut ⇒ oob’
* Funkcije
  + sOut ⇐ sIn[0] != 0 && sIn[1] == 0 ? sIn[2]++ : sIn[2]

*Update OOB matrix*, slučaj B

* Ulazi
  + sIn[0] ⇐ atrue
  + sIn[1] ⇐ NULL
  + sIn[2] ⇐ totco
* Izlazi
  + sOut ⇒ oob’
* Funkcije
  + sOut ⇐ sIn[0] != 0 ? sIn[2]++ : sIn[2]

Sve opisane jezgre izvedene su pomoću MaxJ API-ja. Glavne funkcionalnosti jezgara izvedene su na način koji omogućuje njihovo kasnije kombiniranje u kompozitne jezgre.

Za svaku jezgru implementirana je softverska referenca. Jezgre su preliminarno ispitane na ispravnost ne uzimajući u obzir rubne slučajeve. Dubinsko ispitivanje ispravnosti, te ispitivanje performansi provodi se u sklopu aktivnosti A7.3 „Evaluation of performance, comparison with pure SW implementations“.