W3: Hardware implementation of rule learning procedures

W3-1: Implementation of interface between FPGA and software realization of machine learning algorithms

Evaluation of usefulness and correctness of the selected routines in software (W3-1-4)

Sažetak

Analizirane su rutine koje su u prijašnjim aktivnostima projekta identificirane kao kandidati za izvedbu u FPGA sklopovlju. Tri rutine (*Construct feature*, *Count coverage* i *Update oob matrix*) paralelizirane su pomoću OpenMP, te je provedeno ispitivanje utjecaja paralelizacije na vrijeme izvršavanja algoritma za učenje pravila (*rule learning*). Preostala rutina (*Compute confusion matrix*) nije dalje analizirana zbog premalog udjela u vremenu izvršavanja algoritma. Najizgledniji kandidat za izvedbu u FPGA sklopovlju je rutina *Construct feature*, čija je paralelizacija polučila dobar rezultat (ubrzanje konstrukcije jednog pravila do 4,146× na 6 dretvi).

Uvod

U prijašnjoj aktivnosti W3-1-3 „Specification of the routines that are potential candidates for implementation in hardware“ definirane su 4 rutine koje su kandidati za implementaciju u FPGA sklopovlju:

1. *Construct feature* – rutina za konstrukciju značajke (*feature*).
2. *Count coverage* – rutina za ocjenu kvalitete pojedine značajke.
3. *Compute confusion matrix* – rutina za ocjenu kvalitete naučenih pravila u smislu ispravnosti klasifikacije.
4. *Update oob matrix* – rutina za ažuriranje *out-of-bag* matrice.

U ovom dijelu aktivnosti ocjenjuje se potencijalna korist od implementacije navedenih rutina u FPGA sklopovlju.

Metodologija

Potencijalni dobitak od izvedbe navedenih rutina u FPGA sklopovlju ocjenjuje se kroz njihovu paralelizaciju u softveru i ispitivanjem dobitka u vremenu izvršavanja glavne rutine za konstrukciju pravila (*CPU rule*).

Paralelizacija rutina provodila se pomoću OpenMP [1]. za svaku rutinu identificirana je jedna ili više petlji u kojim se izvršava glavnina posla. Petlje su prema potrebi modificirane kako bi se omogućilo njihovo više-dretveno izvršavanje (npr. preformulacijom petlje da bi se izbacila *break* naredba). Nužno je bilo modificirati i niz funkcija koje se pozivaju iz petlji kako bi postale sigurne za više-dretveno izvršavanje (najčešće pretvorbom globalnih varijabli u lokalne), te se osigurala ispravnost dobivenih rezultata.

Pri ispitivanju mjeri se vrijeme izvršavanja glavne funkcije za izgradnju pravila *cpu_rule*, te vremena izvršavanja za pojedinu rutina koja se paralelizira. Mjerena vremena izvršavanja se akumuliraju, tj. ona su zbroj vremena izvršavanja svakog pojedinog poziva funkcija/rutina.

Mjerenja se vrše za izvršavanje programa na različitom broju dretvi (*thread*), koji se mijenja od 1 do 6. Kao ulazni podaci korišteni su sintetski skupovi podataka generirani u sklopu aktivnosti iz W2 „*Evaluation on synthetic data*“. Skupovi pokrivaju sve kombinacije {35, 100, 350, 1000} atributa i {350, 1000, 3500, 10000} primjera.

Iz izmjerenih vremena izračunava se ubrzanje (*speedup*) prema:

$$S_n = \frac{t_1}{t_n}$$

gdje je t_n vrijeme izvršavanja na n dretvi.

Za različite veličina skupova, sakupljeni su i analizirani podaci za izvršavanje s fiksiranim brojem atributa, odnosno primjera. Za uvid u utjecaj broja primjera, broj atributa fiksiran je na 1000, dok je za uvid u utjecaj broja atributa, broj primjera fiksiran na 10000.

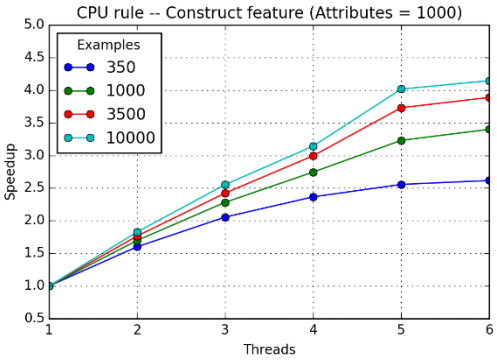
Ispitni program kompiliran je gcc kompajlerom, verzija 4.4.7. Program je izvršavan na radnoj stanici s Intel Xeon E5-1650 CPU (6 jezgri, 3,2 GHz) i 16 GiB DDR3-1600 RAM, pod operacijskim sustavom Linux (Centos 6.7, kernel 2.6.32).

Opcije kompajlera: -fopenmp -Wall -g -O3 -fno-omit-frame-pointer

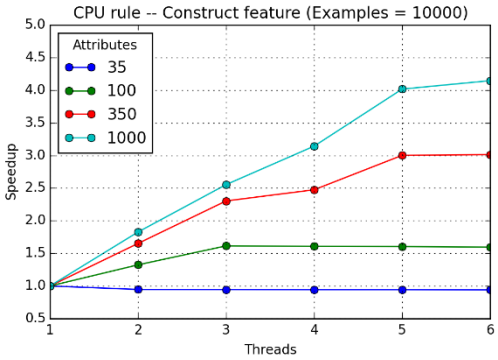
Rutina *Construct feature*

Glavnina posla izvršava se u petlji *LOOP4* iz koje se poziva funkcija *construct_feature*. Paralelizira se petlja *LOOP4*, čime se postiže više istovremenih izvršavanja funkcije *construct_feature*. Navedena funkcija, kao i funkcije koje s pozivaju iz nje, značajno su modificirane kako bi se učinile više-dretveno sigurnima. Sama petlja *LOOP4* modificirana je da se izbjegne korištenje *break* naredbe. Uz vrijeme izvršavanja funkcije *cpu_rule*, mjereno je i vrijeme izvršavanja same petlje *LOOP4*.

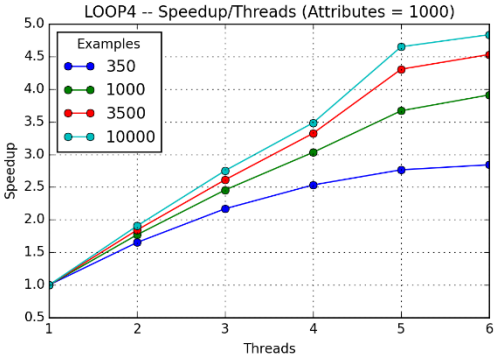
Izmjerena ubrzanja dana su u tablicamaTablica 1 1-4, te su prikazana na slikama 1-4.



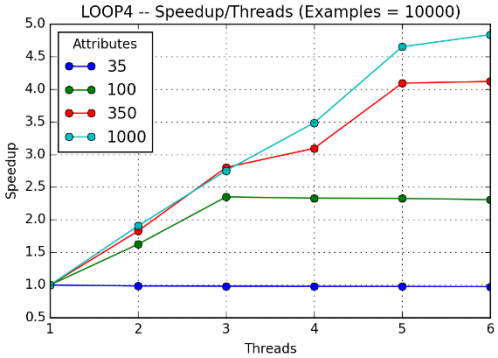
Slika 1. Ubrzanje funkcije cpu_rule*, utjecaj broja primjera*



Slika 2. Ubrzanje funkcije cpu_rule*, utjecaj broja atributa*



Slika 3. Ubrzanje petlje LOOP4*, utjecaj broja primjera*



Slika 4. Ubrzanje petlje LOOP4*, utjecaj broja atributa*

Tablica 1. Ubrzanje funkcije cpu_rule*, utjecaj broja primjera*

N_A = 1000	Primjera / Ubrzanje			
Dretvi	**350**	**1000**	**3500**	**10000**
1	1,000	1,000	1,000	1,000
2	1,601	1,697	1,766	1,827
3	2,056	2,281	2,423	2,553
4	2,366	2,745	2,995	3,143
5	2,556	3,231	3,731	4,017
6	2,617	3,400	3,890	4,146

Tablica 2. Ubrzanje funkcije cpu_rule*, utjecaj broja atributa*

N_E = 10000	Atributa / Ubrzanje			
Dretvi	**35**	**100**	**350**	**1000**
1	1,000	1,000	1,000	1,000
2	0,947	1,324	1,652	1,827
3	0,945	1,615	2,303	2,553
4	0,944	1,607	2,474	3,143
5	0,944	1,604	3,002	4,017
6	0,940	1,596	3,013	4,146

Tablica 3. Ubrzanje petlje LOOP4*, utjecaj broja primjera*

N_A = 1000	Primjera / Ubrzanje			
Dretvi	**350**	**1000**	**3500**	**10000**
1	1,000	1,000	1,000	1,000
2	1,652	1,770	1,842	1,905
3	2,168	2,456	2,612	2,752
4	2,533	3,032	3,324	3,485
5	2,765	3,670	4,306	4,650
6	2,841	3,912	4,532	4,835

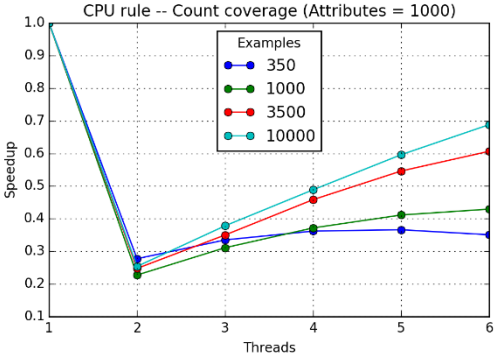
Tablica 4. Ubrzanje petlje LOOP4*, utjecaj broja atributa*

N_E = 10000	Atributa / Ubrzanje			
Dretvi	**35**	**100**	**350**	**1000**
1	1,000	1,000	1,000	1,000
2	0,985	1,625	1,827	1,905
3	0,983	2,350	2,801	2,752
4	0,980	2,330	3,094	3,485
5	0,979	2,327	4,093	4,650
6	0,974	2,307	4,119	4,835

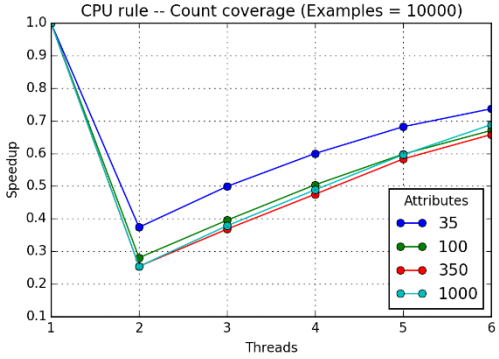
Rutina *Count coverage*

Rutinu *Count coverage* čini četiri petlje unutar funkcije *construct_feature*, od kojih se izvršava samo po jedna pri svakom izvršavanju funkcije. Sve četiri petlje strukturno su identične, a koja od petlji se izvršava ovisi o vrsti značajke. U svrhu paralelizacije, rutina je izdvojena u novu funkciju *count_coverage*, gdje je svaka od petlji paralelizirana raspodjelom iteracija petlji na dretve. Funkcija *construct_feature* poziva se iz petlje *LOOP4*, tj. u sklopu rutine *Construct feature*. Ukoliko je ta rutina paralelizirana, implicitno je paralelizirana i rutina *Count coverage*. U ovom slučaju ispituje se eksplicitna paralelizacija rutine. Uz vrijeme izvršavanja funkcije *cpu_rule*, mjereno je i vrijeme izvršavanja funkcije *count_coverage*.

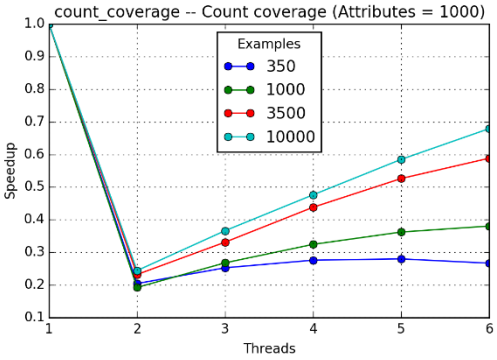
Izmjerena ubrzanja dana su u tablicamaTablica 1 5-8, te su prikazana na slikama 5-8.



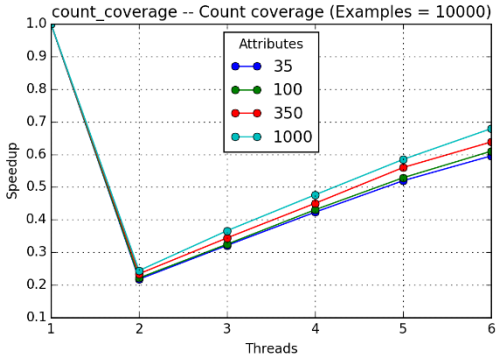
Slika 5. Ubrzanje funkcije cpu_rule*, utjecaj broja primjera*



Slika 6. Ubrzanje funkcije cpu_rule*, utjecaj broja atributa*



Slika 7. Ubrzanje funkcije count_coverage*, utjecaj broja primjera*



Slika 8. Ubrzanje funkcije count_coverage*, utjecaj broja atributa*

Tablica 5. Ubrzanje funkcije cpu_rule*, utjecaj broja primjera*

N_A = 1000	Primjera / Ubrzanje			
Dretvi	**350**	**1000**	**3500**	**10000**
1	1,000	1,000	1,000	1,000
2	0,277	0,227	0,248	0,254
3	0,335	0,311	0,350	0,378
4	0,362	0,372	0,458	0,489
5	0,366	0,411	0,546	0,596
6	0,351	0,429	0,607	0,688

Tablica 6. Ubrzanje funkcije cpu_rule*, utjecaj broja atributa*

N_E = 10000	Atributa / Ubrzanje			
Dretvi	**35**	**100**	**350**	**1000**
1	1,000	1,000	1,000	1,000
2	0,374	0,280	0,254	0,254
3	0,499	0,395	0,368	0,378
4	0,600	0,504	0,475	0,489
5	0,682	0,598	0,583	0,596
6	0,737	0,670	0,658	0,688

Tablica 7. Ubrzanje funkcije count_coverage*, utjecaj broja primjera*

N_A = 1000	Primjera / Ubrzanje			
Dretvi	**350**	**1000**	**3500**	**10000**
1	1,000	1,000	1,000	1,000
2	0,205	0,192	0,232	0,244
3	0,253	0,268	0,331	0,366
4	0,276	0,325	0,438	0,476
5	0,280	0,362	0,526	0,585
6	0,267	0,381	0,588	0,679

Tablica 8. Ubrzanje funkcije count_coverage*, utjecaj broja atributa*

N_E = 10000	Atributa / Ubrzanje			
Dretvi	**35**	**100**	**350**	**1000**
1	1,000	1,000	1,000	1,000
2	0,218	0,222	0,234	0,244
3	0,321	0,325	0,344	0,366
4	0,423	0,431	0,450	0,476
5	0,520	0,529	0,560	0,585
6	0,596	0,610	0,638	0,679

Rutina *Compute confusion matrix*

Rutina *Compute confusion matrix* izvedena je u funkciji *comp_confusion.* Dodatnom analizom utvrđeno je da ta funkcija nema značajan utjecaj na ukupno vrijeme izvršavanja programom. Funkcija se izvršava jednom za svako generirano pravilo, tj. jednom nakon svakog izvršavanja funkcije *cpu_rule*. Mjerenjem vremena izvršavanja potvrđeno je da rutina ne uzima dovoljno vremena da bi njena paralelizacija bila isplativa. Akumulirano vrijeme izvršavanje rutine ne prelazi 52 µs (najveće vrijeme uz 1000 atributa i 10000 primjera), dok pod istim uvjetima vrijeme funkcije *cpu_rule* iznosi 12,5 s. Omjer akumuliranog vremena izvršavanja rutine i vremena izvršavanja funkcije *cpu_rule*, dani u tablici 9, ne prelazi 75 ppm.

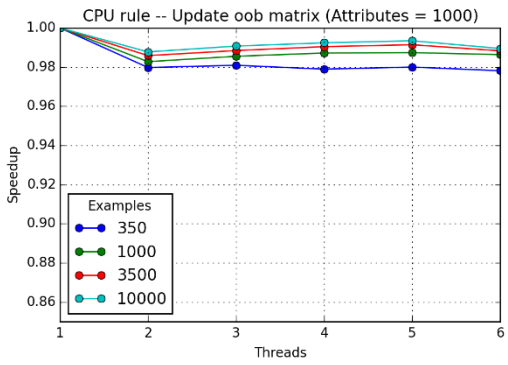
Tablica 9. Omjeri vremena izvršavanja rutine Compute confusion *i vremena izvršavanja funkcije* cpu_rule

	Atributa / $T_{comp\_confusion}/T_{cpu\_rule} \times 10^{-6}$			
Primjera	**35**	**100**	**350**	**1000**
350	74,07	35,28	13,20	3,966
1000	54,13	31,34	12,96	4,761
3500	49,04	29,09	12,18	4,546
10000	46,20	27,48	10,82	4,127

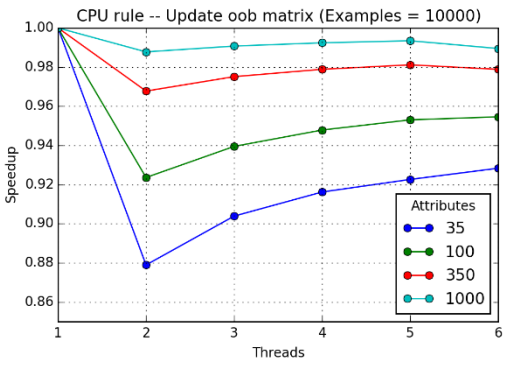
Rutina *Update oob matrix*

Rutinu *Update oob matrix* čine dvije petlje od kojih jedna obrađuje skup za učenje, a druga skup za ispitivanje. Navedene petlje izvršavaju se nakon petlje *LOOP3*, što znači da se rutina izvršava samo jednom za jedno pravilo. Petlje su paralelizirane raspodjelom iteracija petlji, tj. skupova, na dretve. Uz vrijeme izvršavanja funkcije *cpu_rule*, mjereno je i vrijeme izvršavanja rutine *Update oob matrix*.

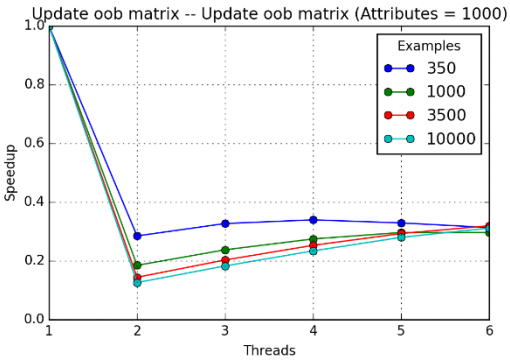
Izmjerena ubrzanja dana su u tablicama 10-13Tablica 1, te su prikazana na slikama 9-12.



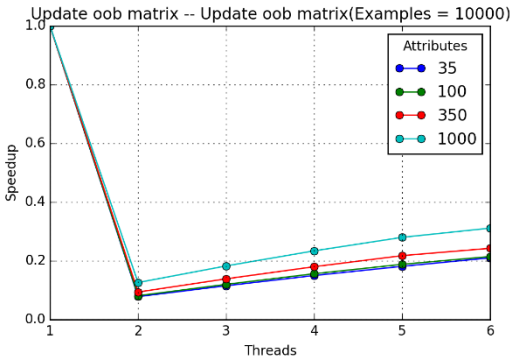
Slika 9. Ubrzanje funkcije cpu_rule*, utjecaj broja primjera*



Slika 10. Ubrzanje funkcije cpu_rule*, utjecaj broja atributa*



Slika 11. Ubrzanje rutine Update oob matrix*, utjecaj broja primjera*



Slika 12. Ubrzanje rutine Update oob matrix*, utjecaj broja atributa*

Tablica 10. Ubrzanje funkcije cpu_rule*, utjecaj broja primjera*

N_A = 1000	Primjera / Ubrzanje			
Dretvi	**350**	**1000**	**3500**	**10000**
1	1,000	1,000	1,000	1,000
2	0,980	0,983	0,986	0,988
3	0,981	0,986	0,988	0,991
4	0,979	0,987	0,990	0,992
5	0,980	0,987	0,991	0,993
6	0,978	0,986	0,988	0,989

Tablica 11. Ubrzanje funkcije cpu_rule*, utjecaj broja atributa*

N_E = 10000	Atributa / Ubrzanje			
Dretvi	**35**	**100**	**350**	**1000**
1	1,000	1,000	1,000	1,000
2	0,879	0,924	0,968	0,988
3	0,904	0,939	0,975	0,991
4	0,916	0,948	0,979	0,992
5	0,923	0,953	0,981	0,993
6	0,928	0,955	0,979	0,989

Tablica 12. Ubrzanje rutine Update oob matrix*, utjecaj broja primjera*

N_A = 1000	Primjera / Ubrzanje			
Dretvi	**350**	**1000**	**3500**	**10000**
1	1,000	1,000	1,000	1,000
2	0,285	0,185	0,144	0,126
3	0,327	0,238	0,203	0,183
4	0,340	0,275	0,253	0,234
5	0,329	0,297	0,293	0,280
6	0,313	0,297	0,320	0,312

Tablica 13. Ubrzanje rutine Update oob matrix*, utjecaj broja atributa*

N_E = 10000	Atributa / Ubrzanje			
Dretvi	**35**	**100**	**350**	**1000**
1	1,000	1,000	1,000	1,000
2	0,078	0,081	0,094	0,126
3	0,115	0,120	0,139	0,183
4	0,150	0,156	0,180	0,234
5	0,181	0,188	0,218	0,280
6	0,210	0,215	0,243	0,312

Diskusija rezultata

Sve ispitivane rutine i nakon paralelizacija daju ispravne rezultate, tj. rezultati su identični onima bez paralelizacije, osim u slučaju rutine *Construct feature*. Razlika u rezultatima kod te rutine posljedica je modifikacije petlje *LOOP4*, kod koje je promijenjen način ranijeg završetka petlje (uklanjanje *break* naredbe i podjela iteracija petlje u blokove).

Za sve paralelizirane rutine pokazalo se da veličina ulaznog skupa za obradu ima značajan utjecaj na dobitak pralelizacije. U pravilu, što je veći broj primjera i atributa u skupu to je veće dobiveno ubrzanje. Ubrzanje je ovisno o količini operacija koje se izvode u jednoj iteraciji pojedine rutine. Što je broj operacija veći, to je veće ubrzanje. To je posljedica činjenice da višedretveno izvršavanje unosi određeni trošak (overhead) stvaranja i sinkroniziranja dretvi. Da bi se postiglo ubrzanje, ušteda u vremenu od pralelizacije mora pokriti dodatni vremenski trošak upravljanjem dretvama. U protivnom umjesto ukupnog ubrzanje dobije se usporavanje izvršavanja rutine.

Paralelizacija rutina *Count coverage* i *Update oob matrix* nema značajan utjecaj na brzinu izvršavanja centralne funkcije *cpu_rule*, a time ni na program u cjelini. Također, paraleliziranje tih rutina nije ubrzalo njihovo izvršenje. Više-dretveno izvršavanje uzrokovalo je sporije izvršavanje rutine, što je posljedica premalog broja primjera u skupovima za učenje. S obzirom na mali broj operacija unutar petlji koje se čine rutine, potreban je izrazito velik broj primjera u skupu da bi se kompenzirao trošak stvaranja i upravljanja dretvama. Ispitni skupovi nisu bili dovoljno veliki da bi se pronašla točka jediničnog ubrzanja. Iz pogleda dobitka u vremenu izvršavanja, ove rutine same po sebi nisu izgledan kandidat za izvedbu u FPGA sklopovlju.

Rutina *Compute confusion matrix* uzima premali udio u vremenu izvršavanja da bi se razmatrala paralelizacija. Stoga nije provedena njena paralelizacija, ni daljnja analiza.

Rutina *Construct feature* polučila je najbolji dobitak od paralelizacije, što je očekivano s obzirom na to da se unutar njene glavne petlje izvršava značajna količina posla, u koji je uključena i petlja *count_coverage*. Najbolje ubrzanje za samu rutinu je 4,835× (10000 primjera i 1000 atributa, na 6 dretvi). Za centralnu funkciju *cpu_rule* ono iznosi 4,146× (10000 primjera i 1000 atributa, na 6 dretvi). Taj rezultat upućuje da je ta rutina, a posljedično i rutine koji su njen sastavni dio, glavni kandidat za izvedbu u FPGA sklopovlju.

Bibliografija

[1] OpenMP Architecture Review Board, “OpenMP Application Program Interface,” 2011. [Online]. Available: http://www.openmp.org/mp-documents/OpenMP3.1.pdf. [Accessed: 05-Feb-2014].