TEHNIČKO REŠENJE

IP jezgra za hardversku realizaciju ansambala stabala odluka

M-85: Prototip, nova metoda, softver**,** standardizovan ili atestiran instrument, nova genetska proba, mikroorganizmi

Autori:

Rastislav Struharik, Fakultet tehničkih nauka (FTN), Novi Sad,
Ladislav Novak, Fakultet tehničkih nauka (FTN), Novi Sad.

1. Kratak opis

Hardverske implementacije ansambala stabala odluka mogu predstavljati jedino rešenje u slučajevima kada je potrebno izvršiti klasifikaciju instance u vrlo kratkom vremenu, ili kada je potrebno adaptivno formiranje prediktivnog modela, u toku rada sistema. Ovo su tipični zahtevi koji se sreću prilikom projektovanja savremenih a pogotovo budućih *embedded* sistema. Imajući u vidu ove činjenice, razvijena su IP jezga za hardversku realizaciju ansambala stabala odluka koja u mnogome olakšavaju integraciju i rada sa stablima odluka prilikom projektovanja *embedded* sistema.

Tehničke karakteristike:

IP jezgra za hardversku realizaciju ansambala stabala odluka modelovana su pomoću standardnog jezika za specifikaciju hardvera, VHDL. Razvijeni modeli su parametrizovani što omogućava jednostavno prilagođavanje arhitekture trenutnim potrebama korisnika.

Tehničke mogućnosti:

Korišćenjem konfiguracionih parametara definisanih unutar VHDL modela arhitektura za realizaciju ansambala stabala odluka (broj bitova koji se koristi za predstavu vednosti atributa problema, koeficijenata razdvajajućih hiperpovrši, ciljnih klasa; broja atributa preko kojih je opisan klasifikacioni problem; broj stabala u ansamblu; maksimalnog broja čvorova u realizovanom stablu odluke, itd.) moguće je prilagoditi VHDL model potrebama tekuće aplikacije. Pilikom sinteze hardvera ovi parametri se koriste kako bi se automatski generisala optimalna hardverska implementacija za tekuću aplikaciju.

Realizator:

Fakultet tehničkih nauka – FTN.

Korisnik:

Fakultet tehničkih nauka – FTN.

Podtip rešenja:

Softver -M85

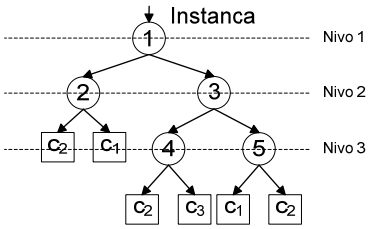
Projekat u okviru koga je realizovano tehničko rešenje:

Program istraživanja u oblasti tehnološkog razvoja za period 2011.-2014.

Tehnološka oblast:	Elektronika, telekomunikacije i informacione tehnologije
Rukovodilac projekta:	dr Ljiljana Živanov, redovni profesor
Naziv projekta:	Nove generacije ugrađenih elektronskih komponenti i sistema u neorganskim i organskim tehnologijama za uređaje široke potrošnje
Broj projekta:	TR 32016

2. Stanje u svetu

Posmatrajmo binarno stablo odluke prikazano na slici 2.1. Ovo stablo bi moglo biti rezultat izvršavanja nekog od algoritama za formiranje stabala odluke [1], [2].



Slika 2.1 Binarno stablo odluke

Stablo odluke klasifikuje instance prolazeći kroz stablo počevši od korena do nekog od listova, kome je asocirana jedna od mogućih klasa, i na taj način obezbeđuje klasifikaciju tekuće instance. Svaki čvor u stablu specificira test nekog od atributa date instance (mada se test može sastojati i od više od jednog atributa), a svaka od grana koja polazi od tog čvora predstavlja jedan od mogućih ishoda testa. Instanca se klasifikuje počevši od korena stabla, izvodeći test koji je pridružen korenu, a zatim se kreće duž grane koja odgovara rezultatu izvedenog testa. Ovaj proces se zatim rekurzivno ponavlja uzimajući podstablo čiji koren predstavlja čvor do kojeg se stiže krećući se duž odabrane grane.

Kao što je rečeno algoritmi za formiranje stabala odluka mogu da rade sa problemima koji su opisani kategoričkim i numeričkim atributima. Pošto su praktični problemi najčešće predstavljeni pomoću numeričkih atributa, detaljno će biti razmotreni mogući tipovi testova koji se mogu koristiti u ovom slučaju.

Ako se u svakom čvoru koristi test samo jednog atributa onda taj test u opštem slučaju ima oblik

$$A_i > a_i \tag{2.1}$$

Ovaj test ekvivalentan je hiperravni koja je ortogonalna u odnosu na osu koja predstavlja atribut A_i. Stabla koja koriste ovakve testove zovu se *stabla odluka sa ortogonalnim hiperravnima*. Ovo je najčešći tip stabala odluka koja se koriste u praksi. Kada se formira stablo odluke sa ovakvim testovima, ono će izvršiti particiju hiperprostora koristeći samo ovakve, ortogonalne hiperravni.

Međutim, moguće je formirati i znatno složenije testove. Testovi mogu da predstavljaju linearnu kombinaciju atributa problema,

$$\sum_{i=1}^{n} a_i A_i + a_{n+1} > 0 \tag{2.2}$$

Kao što se može videti iz prethodne jednačine ovakvi testovi zapravo definišu hiperravni potpuno proizvoljnog položaja u hiperprostoru definisanom atributima problema. Stabla koja koriste ovakve testove zovu se *stabla odluka sa neortogonalnim hiperravnima*. Ovakvi testovi mogu biti vrlo pogodni ako je priroda problema takva da se particija instanci može znatno lakše izvesti pomoću neortogonalnih hiperravni, jer će tada formirana stabla odluka biti znatno manja nego u slučaju da se koriste testovi samo pojedinačnih atributa. Sa druge strane, problem nalaženja optimalnog položaja neortogonalne hiperravni je znatno teži jer on zahteva pretragu u $n+1$ dimenzionalnom prostoru koeficijenata za razliku od slučaja kada se koriste ortogonalne hiperravni kada je potrebno vršiti pretragu u 2 dimenzionalnom prostoru pretrage. U literaturi je pokazano da je problem pronalaženja optimalnog položaja neortogonalne hiperravni NP težak problem, [3].

Na kraju, mogu se koristiti još opštiji testovi. Najopštiji oblik testa u nekom od čvorova stabla ima sledeći oblik.

$$f(A_1, A_2, ..., A_n) > 0 \tag{2.3}$$

Funkcija f može biti bilo koja funkcija n atributa. Sada se prilikom particije hiperprostora atributa koristi hiperpovrš definisana funkcijom f. Ako je funkcija f nelinearna funkcija, stabla koja koriste takve testove se zovu *stabla odluka sa nelinearnim hiperpovršima*.

Za razliku od ortogonalnih hiperpovrši koje se mogu opisati jednoznačno korišćenjem izraza (2.2), nelinearne hiperpovrši se ne mogu jednoznačno opisati. U ovom radu razmatrane su nelinearne hiperpovrši bazirane na korišćenju polinoma drugog i trećeg reda.

$$\sum_{i=1}^{n} a_i A_i^2 + \sum_{i=1}^{n} a_i A_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{i,j} A_i A_j + c > 0$$
$$\sum_{i=1}^{n} a_i A_i^3 + \sum_{i=1}^{n} a_i A_i^2 + \sum_{i=1}^{n} a_i A_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{i,j} A_i A_j + \tag{2.4}$$
$$+ \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j} A_i^2 A_j + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} a_{i,j,k} A_i A_j A_k + c > 0$$

Stablo prikazano na slici 2.1 sadrži ukupno pet čvorova i šest listova raspoređenih u četiri nivoa. U svakom od čvorova definisan je po jedan test atributa klasifikacionog problema. Ovi testovi mogu biti ortogonalni, (2.1), neortogonalni, (2.2), ili nelinearni, (2.4). Instanca problema koju je potrebno klasifikovati (u slučaju stabla sa slike 2.1 u jednu od tri moguće klase) dovodi se u koren stabla. U zavisnosti od ishoda testa specificiranog u korenu stabla, instanca se prosleđuje jednom od dva naslednika (čvorovi 2 i 3). Čitav postupak se zatim ponavlja sve dok se ne dođe do nekog od listova stabla. U tom trenutku instanca se klasifikuje u klasu koja je pridružena posmatranom listu. Tipično se čitav ovaj postupak realizuje u vidu programa koji se zatim izvršava na nekom od procesora opšte namene.

Ansambli prediktivnih modela bazirani su na ideji kombinovanja predikcija većeg broja individualno obučenih modela (tipično stabala odluke ili veštačkih neuronskih mreža) prilikom klasifikovanja novih instanci. Ova ideju koristimo u svakodnevnom životu prilikom donošenja važnih odluka koje mogu imate značajne finansijske, medicinske ili socijalne konsekvence. Konsultovanje većeg broja eksperata prilikom donošenja neke važne odluke, izvodi se u cilju učvršćivanja ubeđenja da donosimo ispravnu odluku. Stoga je bilo prirodno da se sličan koncept primeni i u oblasti mašinskog učenja.

Najraniji rad iz oblasti ansambala prediktivnih modela jeste [4], u kome se predlaže particija prostora atributa korišćenjem dva ili više klasifikatora. U radu [5] autori su pokazali da se generalizacija izolovane veštačke neuronske mreže može poboljšati korišćenjem ansambla veštačkih neuronskih mreža sa sličnom topologijom. Prekretnicu u razvoju teorije ansambala prediktivnih modela napravio je Shapire sa svojim radom [6] u kojem je dokazao da se jak klasifikator u PAC smislu (Probably Approximately Correct, PAC) može dobiti kombinovanjem slabih klasifikatora. Nakon ovog ključnog rada, istraživanja u oblasti ansambala prediktivnih modela su se intenzivirala, iako su se rezultati u literaturi pojavljivali pod različitim imenima. Dugačka lista uključuje: smeše eksperata [7-8], sjedinjavanje konsenzusa [9], kombinaciju višestrukih klasifikatora [10-14], fuziju klasifikatora [15-17], komitete neuronskih mreža [18], glasajući skup klasifikatora [19], ansamble klasifikatora [20], itd.

Svi ovi pristupi se međusobno razlikuju u načinu na koji formiraju individualne prediktivne modele i kombinuju njihove predikcije. Generalno, postoje dva načina na koji se mogu kombinovati prediktivni modeli: selekcija klasifikatora i fuzija klasifikatora. U slučaju selekcije klasifikatora, svaki klasifikator je obučen tako da postane ekspert u jednom, lokalnom regionu čitavog prostora atributa. Kombinovanje klasifikatora je zatim bazirano na trenutnom ulaznom vektoru: klasifikator obučavan sa podacima koji su najbliži trenutnom ulaznom vektoru, mereno pomoću odgovarajuće metrike, dobija najveći značaj. Jedan ili više lokalnih eksperata mogu biti nominovani da donesu odluku [7], [14], [21-23]. U slučaju fuzije klasifikatora, svaki od članova ansambla obučava se sa primerima iz čitavog prostora atributa. U ovom slučaju, kombinovanje klasifikatora uključuje sjedinjavanje individualnih (slabih) prediktivnih modela u cilju formiranja jednog (jakog) eksperta sa superiornim performansama. U ovu grupu spadaju poznati Bagging [24], Boosting [6], [25] algoritmi, kao i njihove brojne varijacije.

Kombinacija prediktivnih modela može se odnositi na diskretne, ili pak na kontinualne vrednosti izlaza individualnih prediktivnih modela [17], [26-27]. U drugom slučaju, predikcije pojedinih klasifikatora su najčešće normalizovane na interval [0, 1], i interpretiraju se kao podrška klasifikatora svakoj od mogućih klasa, ili čak kao uslovne verovatnoće [17], [28]. Ovakva interpretacija omogućava formiranje ansambla korišćenjem algebarskih pravila kombinovanja (većinsko glasanje, maksimum/minimum/suma/proizvod ili neka druga kombinacija uslovnih verovatnoća), [11], [27], [29-30], različite kombinacije bazirane na fuzzy logici, [16], [31-32], Dempster-Shafer fuziju klasifikatora, [12], [33], itd.

Prediktivni model sa savršenom generalizacijom nije moguće realizovati, usled šuma koji je prisutan u podacima, neadekvatno odabranih atributa, preklapajućih raspodela podataka iz različitih klasa, itd. U najboljem slučaju moguće je napraviti model koji će ispravno klasifikovati podatke u najvećem broju slučajeva. Osnovna ideja prediktivnih modela baziranih na korišćenju ansambala sastoji se u formiranju velikog broja individualnih klasifikatora, i kombinovanju njihovih predikcija na način koji će obezbediti da tačnost ansambla bude veća od tačnost individualnog modela. Da bi to bilo moguće, neophodno je da individualni klasifikatori prave greške na različitim instancama problema. U slučaju da svaki klasifikator pravi drugačije greške, njihovim pravilnim kombinovanjem biće moguće umanjiti ukupnu grešku klasifikacije. Stoga je osnovni uslov koji je potrebno zadovoljiti prilikom formiranja ansambla prediktivnih modela, da svaki od članova ansambla bude što je moguće više unikatan, pogotovo kada se posmatra njegovo ponašanje u slučaju pogrešno klasifikovanih instanci. Drugačije rečeno, potrebni su nam klasifikatori čije se razdvajajuće površi u značajnoj meri razlikuju od razdvajajućih površi ostalih klasifikatora iz ansambla. Za takav skup klasifikatora kažemo da je raznolik.

Raznolikost klasifikatora se može postići na nekoliko načina. Najpopularniji metod sastoji se u korišćenju različitih trening skupova za formiranje individualnih klasifikatora. Ovakvi trening skupovi se često dobijaju kao rezultat korišćenja tehnika odabiranja, kao što su Bootstraping ili Bagging, kod kojih se trening skupovi formiraju slučajnim odabirom instanci iz celokupnog trening skupa. Drugi pristup za ostvarivanje raznolikosti koristi drugačije parametre prilikom obučavanja individualnih klasifikatora. Na primer, čitav niz veštačkih neuronskih mreža može biti formiran obučavanjem koje koristi različite inicijalne vrednosti težina veza, različit broj slojeva odnosno neurona u mreži, različite ciljne funkcije, itd. Menjanjem ovih parametara može se uticati na raznolikost formiranih modela. Mogućnost da se relativno lako može kontrolisati struktura i performanse formiranog modela u slučaju veštačkih neuronskih mreža i stabala odluke, čini ih pogodnim kandidatima za korišćenje u ansamblima. Raznolikost se može postići i korišćenjem različitih tipova klasifikatora unutar jednog ansambla, na primer neuronskih mreža, stabala odluke, support vector mašina, itd. Na kraju, raznolikost se može postići i korišćenjem različitih podskupova atributa prilikom obučavanja individualnih klasifikatora [34].

Da bi se raznolikost mogla numerički izraziti, u praksi se koristi veliki broj različitih mera raznolikosti, od kojih su najpoznatije: mera bazirana na korelaciji, Q-statistika, entropijska mera, Kohavi-Wolpert varijansa, mera težine, itd.

Svaki ansambl prediktivnih modela sastoji se od dve ključne komponente. Prvo, potrebno je definisati postupak za formiranje ansambla koji će biti što je moguće više raznolik. Nakon što je ansambl formiran, potrebno je definisati način na koji će se kombinovati predikcije individualnih prediktivnih modela koji čine ansambl na takav način da se tačne odluke pojačavaju, a netačne smanjuju.

U dostupnoj literaturi postoji veliki broj predloženih postupaka za formiranje ansambla, od kojih su najpoznatiji: Bagging [24], Boosting [6], AdaBoost [25], Stacked generalization [35] i Mixture of experts [36]. Prilikom formiranja ansambla mora se odgovoriti na dva pitanja. Kako će se formirati individualni prediktivni modeli i kako će se oni međusobno razlikovati. Odgovori na ova pitanja direktno utiču na raznolikost ansambla, a time i na njegove performanse kao prediktivnog modela. Zbog toga bi svaki postupak za formiranje ansambla trebao da teži da poveća raznolikost ansambla. Međutim, najveći broj postojećih postupaka za formiranje ansambla ne pokušava da maksimizuje neku od mera raznolikosti, iako ima i suprotnih primera, [37-38]. Raznolikost ansambla se uglavnom postiže indirektno, korišćenjem odgovarajućih postupaka odabiranja trening podskupova ili odabirom drugačijih parametara prilikom obučavanja individualnih prediktivnih modela.

Druga ključna komponenta svakog ansambla prediktivnih modela jeste način na koji se predikcije individualnih prediktivnih modela kombinuju u jednu, kolektivnu odluku. Algoritmi za kombinovanje prediktivnih modela mogu se podeliti na algoritme koji zahtevaju obučavanje i algoritme koji ne zahtevaju obučavanje. Algoritme za kombinovanje prediktivnih modela takođe možemo podeliti i na algoritme koji se koriste za kombinovanje prediktivnih modela sa diskretnim izlazom i na algoritme koji se koriste za kombinovanje prediktivnih modela sa kontinualnim izlazom.

U slučaju algoritama za kombinovanje prediktivnih modela koji zahtevaju dodatno obučavanje, odgovarajući parametri sistema koji se koristi za kombinovanje, tipično su to neke vrste težina, se optimizuju pomoću posebnog postupka obučavanja. Ovako određeni parametri obično zavise od skupa instanci koji je korišćen za njihovo određivanje.

U slučaju druge podele, algoritmi za kombinovanje prediktivnih modela sa diskretnim izlazom kombinuju klase prognozirane od strane individualnih prediktivnih modela, $c_i \in C = \{c_1, c_2\}$. Sa druge strane, algoritmi za kombinovanje prediktivnih modela sa kontinualnim izlazom koriste kontinualne vrednosti generisane od strane individualnih prediktivnih modela. Ove kontinualne vrednosti često predstavljaju stepen podrške koju svakoj od mogućih klasa daje individualni prediktivni model, ili čak mogu biti interpretirane kao aposteriorne uslovne verovatnoće, $P(c_i \mid x)$. Ovo je moguće ukoliko su izlazi prediktivnih modela normalizovani tako da u ukupnoj zbiru daju broj jedan i ukoliko su individualni prediktivni modeli obučavani sa dovoljno reprezentativnim trening skupom. Mnogi od prediktivnih modela, kao što su veštačke neuronske mreže, generišu kontinualne izlaze koji se mogu interpretirati kao aposteriorne uslovne verovatnoće.

Obzirom da stabla odluka generišu diskretne izlaze, prilikom njihovog kombinovanja mogu se koristiti samo algoritmi iz prve grupe. Zbog toga će u nastavku biti predstavljeni samo najpoznatiji algoritmi za kombinovanje prediktivnih modela sa diskretnim izlazom.

Pretpostavimo da su nam od svakog individualnog prediktivnog modela iz ansambla dostupne samo predikcije klase kojoj tekuća instanca pripada, pri čemu je broj različitih klasa je konačan. Označimo predikciju *i*-tog prediktivnog modela sa vektorom $d_i \in \{0,1\}^C$, *i=1, ..., T*, gde je *T* broj prediktivnog modela u ansamblu, a *C* je broj različitih klasa. Ako *i*-ti prediktivni model klasifikuje tekuću instancu u klasu c_k tada je $d_i = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases}, j = 1,...,C$.

Većinsko odlučivanje

Postoje tri varijante većinskog odlučivanja::

- *jednoglasno odlučivanje* – bira se klasa za koju glasaju svi članovi ansambla,

- *odlučivanje prostom većinom* - bira se klasa za koju glasa 50% +1 član ansambla,
- *većinsko odlučivanje* - bira se klasa za koju glasa najveći broj članova ansambla, bez obzira da li je taj broj veći od 50%.

Algoritam na osnovu kojega ansambl donosi odluku korišćenjem većinskog odlučivanja može se opisati na sledeći način:

Odaberi klasu c_K ako važi

$$\sum_{t=1}^{T} d_{t,K} = \max_{j=1}^{C} \sum_{t=1}^{T} d_{t,j} \tag{2.5}$$

Može se pokazati da je većinsko odlučivanje optimalno pravilo za kombinovanje izlaza prediktivnih modela pod uslovom da važe sledeće pretpostavke:

- Ansambl ima neparan broj članova u slučaju problema klasifikacije u dve različite klase,
- verovatnoća da će svaki prediktivni model tačno klasifikovati instancu x iznosi p za svaku instancu x,
- izlazi prediktivnih modela su međusobno nezavisni.

Pod ovim pretpostavkama, odlučivanje prostom većinom i većinsko odlučivanje su identični, a ansambl donosi tačnu odluku ukoliko barem $\lfloor T/2 \rfloor + 1$ prediktivnih modela izabere tačnu klasu. Verovatnoća da ansambl donese ispravnu odluku može se predstaviti pomoću binomne raspodele verovatnoće

$$P_{ans} = \sum_{k=(T/2)+1}^{T} \binom{T}{k} p^k (1-p)^{T-k} \tag{2.6}$$

P_{ans} teži ka 1 kada $T \to \infty$, ako je $p > 0.5$, a teži ka 0 ako je $p < 0.5$. Ovaj rezultat je poznat iz teorije verovatnoće i statistike [39-40]. Upravu ovde leži snaga većinskog odlučivanja: ako za svakog člana ansambla možemo garantovati da je njegova verovatnoća davanja ispravnog odgovora veća od 0.5, tada za dovoljno veliki ansambl (recimo od 100 ili više članova), verovatnoća donošenja ispravne odluke čitavog ansambla teži ka 1. Ova analiza odnosi se na slučaj klasifikacionih problema sa samo dve moguće klase. Može se pokazati da u slučaju klasifikacionih problema sa više od dve moguće klase, verovatnoća ispravne klasifikacije individualnih klasifikatora ne mora biti veća od 0.5, već zapravo može biti i znatno manja, a da tačnost ansambla i dalje teži ka 1. Za odličnu analizu većinskog odlučivanja čitalac može pogledati [41].

Težinsko većinsko odlučivanje

Ukoliko postoje dokazi da su neki od članova ansambla pouzdaniji u donošenju ispravnih odluka od drugih, davanje veće težine njihovim prognozama moglo bi dodatno popraviti performanse čitavog ansambla. Težinsko većinsko odlučivanje dodeljuje težinu w_j prediktivnom modelu h_j koja je proporcionalna poverenju koje imamo njega. Algoritam težinskog većinskog odlučivanja bira klasu c_K ako važi

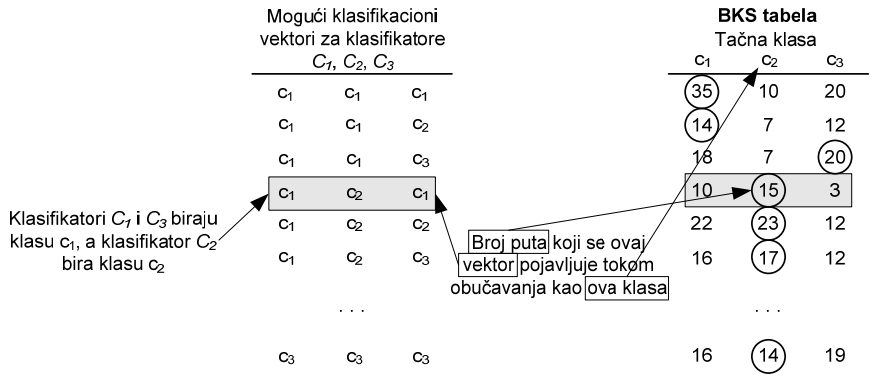
$$\sum_{t=1}^{T} w_t \cdot d_{t,K} = \max_{j=1}^{C} \sum_{t=1}^{T} w_t \cdot d_{t,j} \tag{2.7}$$

Odnosno, bira klasu c_K ako je ukupna težinska suma dodeljena klasi c_K veća od ukupne težinske sume dodeljene bilo kojoj drugoj klasi.

Osnovno pitanje koje se postavlja kod težinskog većinskog odlučivanja jeste na koji način pridružiti težine pojedinim prediktivnim modelima. Ukoliko bi smo posedovali informaciju o tome koji prediktivni modeli imaju veću tačnost klasifikacije, njima bi smo pridružili veću težinu, ili bi smo koristili samo njih. Međutim, mi ovu informaciju nemamo. Jedan od mogućih pristupa jeste da se kao merilo tačnosti svakog prediktivnog modela iz ansambla koristi njegova tačnost estimirana na posebno, validacionom skupu instanci, ili njegova tačnost na trening skupu. Ovaj drugi pristup koristi se u *AdaBoost* algoritmu. Detaljna diskusija o težinskom većinskom odlučivanju može se pronaći u [42].

Behavior Knowledge Space (BKS)

BKS koristi tabelu, formiranu na osnovu klasifikacija trening skupa, u kojoj se vodi evidencija koliko često se pojavljuje svaki od klasifikacionih vektora, [43-44]. Pod klasifikacionim vektorom podrazumeva se vektor formiran od predikcija svakog od članova ansambla, za datu ulaznu instancu. Zatim se svakom od klasifikacionih vektora pridružuje ona klasa za koju se posmatrani vektor najviše puta pojavljuje tokom obučavanja. Prilikom korišćenja, svaki put kada se odgovarajući klasifikacioni vektor pojavi na izlazima ansambla, njemu pridružena klasa se bira kao klasa u koju ansambl klasifikuje tekuću instancu. Opisana procedura se najbolje razume pomoću primera, prikazanog na slici 2.2.



Slika 2.2 Ilustracija BKS *algoritma*

Pretpostavimo da se ansambl sastoji od tri prediktivna modela, C_1, C_2, C_3 i da se koristi za klasifikaciju instanci u jednu od tri klase, c_1, c_2, c_3. U ovom slučaju postoji ukupno 27 različitih klasifikacionih vektora koji se mogu pojaviti na izlazima tri klasifikatora. Tokom obučavanja, potrebno je voditi evidenciju koliko često se svaki klasifikacioni vektor pojavljuje zajedno sa odgovarajućom tačnom klasom za svako od pojavljivanja. Na slici 2.2 prikazana je jedna hipotetička situacija. Na primer, pretpostavimo da se klasifikacioni vektor $\{c_1, c_2, c_1\}$ pojavljuje ukupno 28 puta, od toga u 10 slučaja kada se ovaj vektor pojavio ispravna klasa je bila c_1, u 15 slučaja tačna klasa je bila c_2, dok je u 3 slučaja tačna klasa bila c_3. Obzirom da se ovaj vektor najčešće pojavljivao kada su u pitanju bile instance koje pripadaju klasi c_2, njemu će biti asocirana klasa c_2. Tokom testiranja, svaki put kada se na izlazima tri klasifikatora pojavi klasifikacioni vektor $\{c_1, c_2, c_1\}$ ansambl će tekuću instancu klasifikovati u klasu c_2. Potrebno je primetiti da je izbor klase c_2 drugačiji od izbora klase c_1 koja bi bila izabrana u slučaju većinskog odlučivanja, što jasno ukazuje da *BKS* algoritam koristi potpuno drugačiji pristup od svih prethodno razmatranih algoritama za kombinovanje klasifikatora.

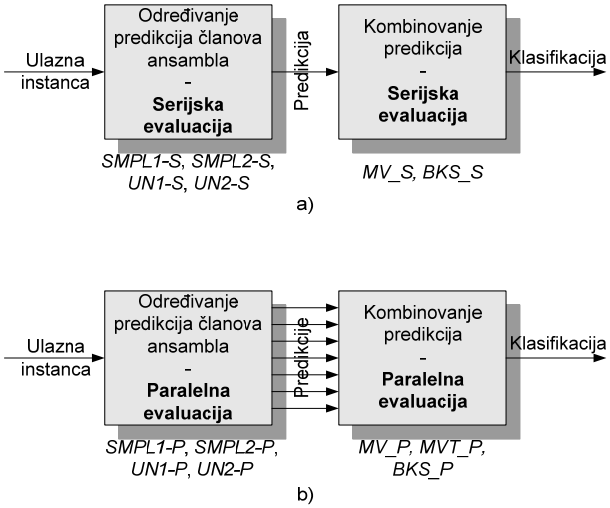
Koliko je autorima ovog tehničkog rešenja poznato, u dostupnoj literaturi postoji samo jedan rad koji tretira problem hardverske implementacije ansambala stabala odluka [45]. Međutim u ovom radu predlaže se samo rešenje koje svako stablo odluke iz ansambla implementira kao poseban modul, unutar kojega se svaki čvor implementira takođe kao poseban modul. Ovaj način implementacije rezultuje u izuzetno velikoj potrošnji hardverskih resursa, što u slučaju rada sa velikim ansamblima sa složenim stablima odluka može predstavljati ograničavajući faktor primenljivosti ovoga rešenja. Pored toga, u predloženo rešenju implementiran je samo najjednostavniji model kombinacije predikcija individualnih klasifikatora iz ansambla, prosto većinsko odlučivanje.

3. IP jezgra za hardversku realizaciju stabala odluka

Ansambli prediktivnih modela bazirani su na ideji kombinovanja predikcija većeg broja individualno obučenih prediktivnih modela (tipično stabala odluke ili veštačkih neuronskih mreža) prilikom klasifikovanja novih instanci. Svaki ansambl prediktivnih modela sastoji se iz dve ključne komponente. Prvu komponentu čine individualni prediktivni modeli od kojih je sastavljen ansambl. Svaka instanca se prilikom klasifikacije pomoću ansambla prvo klasifikuje pomoću svakog člana ansambla. Druga komponenta svakog ansambla predstavlja algoritam pomoću kojega se predikcije individualnih prediktivnih modela kombinuju u jednu, zajedničku predikciju čitavog ansambla.

U ovoj glavi biće izložene arhitekture za efikasnu hardversku implementaciju ansambala formiranih od ortogonalnih, neortogonalnih i nelinearnih stabala odluka. Pretpostavka je da je ansambl već formiran, odnosno da je poznata struktura svakog od stabala odluka koji čine ansambl, kao i da je odabran algoritam za kombinovanje predikcija članova ansambla koji se želi koristiti.

Svaka arhitektura za implementaciju ansambla sastoji se iz dva modula: modula koji implementira individualne članove ansambla i modula koji kombinuje predikcije individualnih članova. Obzirom da se ansambl uvek ima više od jednog člana, svaki od ovih modula može se realizovati na dva načina: serijski i paralelno. Na slici 3.1 prikazana je konceptualna organizacija serijske odnosno paralelne arhitekture za hardversku implementaciju ansambla prediktivnih modela.



*Slika **Error! No text of specified style in document.**.1 Arhitekture za hardversku implementaciju ansambla prediktivnih modela: a) serijska, b) paralelna*

U slučaju serijske implementacije, individualni članovi evaluiraju se serijski, jedan nakon drugog, a i njihove predikcije se kombinuju serijski. Kao što se sa slike 3.1a može videti, modul za određivanje predikcija članova ansambla serijski evaluira članove ansambla i njihove predikcije prosleđuje ka modulu za kombinovanje predikcije serijski, jednu nakon druge. Modul za kombinovanje predikcija u ovom slučaju mora da prihvati predikcije članova ansambla, i da ih zatim kombinuje u jednu, zajedničku klasifikaciju čitavog ansambla, pomoću odabranog algoritma za kombinovanje.

U slučaju paralelne realizacije, svaki od individualnih prediktivnih modela evaluira se u paraleli, a predikcije individualnih članova ansambla se takođe kombinuju u paraleli u cilju određivanja predikcije čitavog ansambla. U ovom slučaju modul za određivanje predikcija članova ansambla istovremeno evaluira sve članove i njihove predikcije prosleđuje ka modulu za kombinovanje predikcija istovremeno, u paraleli, kao što se može videti sa slike 3.1b. Modul za kombinovanje predikcija u istom trenutku prihvata predikcije svih članova ansambla i u paraleli ih kombinuje pomoću odabranog algoritma za kombinovanje da bi odredio zajedničku predikciju čitavog ansambla.

Kao i obično, paralelna arhitektura obezbeđuje veću brzinu klasifikacije od serijske, reda N puta veću, pri čemu je sa N označen broj članova ansambla, ali i zahteva znatno veći broj resursa za implementaciju ansambla od serijske.

Arhitekture za implementaciju ansambala prediktivnih modela prikazane na slici 3.1 mogu se koristiti za implementaciju ansambla koji je sastavljen od proizvoljnih prediktivnih modela kao što su stabla odluka, veštačke neuronske mreže ili *support vector* mašine. Jedina razlika u ovim slučajevima bila bi u strukturi modula za određivanje predikcija članova ansambla. U slučaju da je ansambl sastavljen od stabala odluka za realizaciju

modula za određivanje predikcija mogu se iskoristiti arhitekture za hardversku implementaciju individualnih stabala odluka, predstavljene u [46], sa većim ili manjim modifikacijama.

Ukoliko posmatramo serijsku arhitekturu, modul za određivanje predikcija se zapravo sastoji od samo jednog modula za hardversku implementaciju stabla odluke, baziranog na nekoj od modifikovanih *SMPL* ili *UN* arhitektura, koji se koristi da se serijski evaluira individualna stabla iz ansambla. Modifikacija postojećih *SMPL* i *UN* arhitektura je neophodna jer sada moraju čuvati strukturne podatke ne samo za jedno stablo, već za čitav ansambl. U glavi 3.1 detaljno će biti prikazane modifikacije koje je neophodno izvršiti na *SMPL* i *UN* arhitekturama da bi se one mogle efikasno koristiti za serijsku evaluaciju članova ansambla. Korišćenjem ovih modifikovanih arhitektura moguće je definisati četiri različite arhitekture za paralelnu evaluaciju članova ansambla: *SMPL1-S, SMPL2-S, UN1-S* i *UN2-S*.

Kod paralelne arhitekture modul za određivanje predikcija se sastoji iz *N* nezavisnih modula baziranih na arhitekturama predloženim u [46] koji u paraleli evaluiraju članove ansambla. U ovom slučaju nije potrebna nikakva modifikacija postojećih *SMPL* i *UN* arhitektura, već se one mogu direktno koristiti. Kao i u slučaju serijske arhitekture, i u slučaju paralelne arhitekture mogu se definisati ukupno četiri različite arhitekture, bazirane na postojećim *SMPL* i *UN* arhitekturama: *SMPL1-P, SMPL2-P, UN1-P* i *UN2-P*.

Arhitekture za kombinovanje predikcija individualnih klasifikatora biće predstavljene u glavi 3.2. Treba napomenuti da su predstavljene arhitekture za kombinovanje predikcija univerzalne, odnosno da se mogu koristiti za kombinovanje predikcija različitih prediktivnih modela, a ne samo stabala odluka.

Na kraju, u glavi 3.3, biće izvršena analiza potrebnih hardverskih resursa i performansi predloženih arhitektura za hardversku implementaciju ansambala stabala odluka i izvršeno poređenje sa performansama softverskih implementacija.

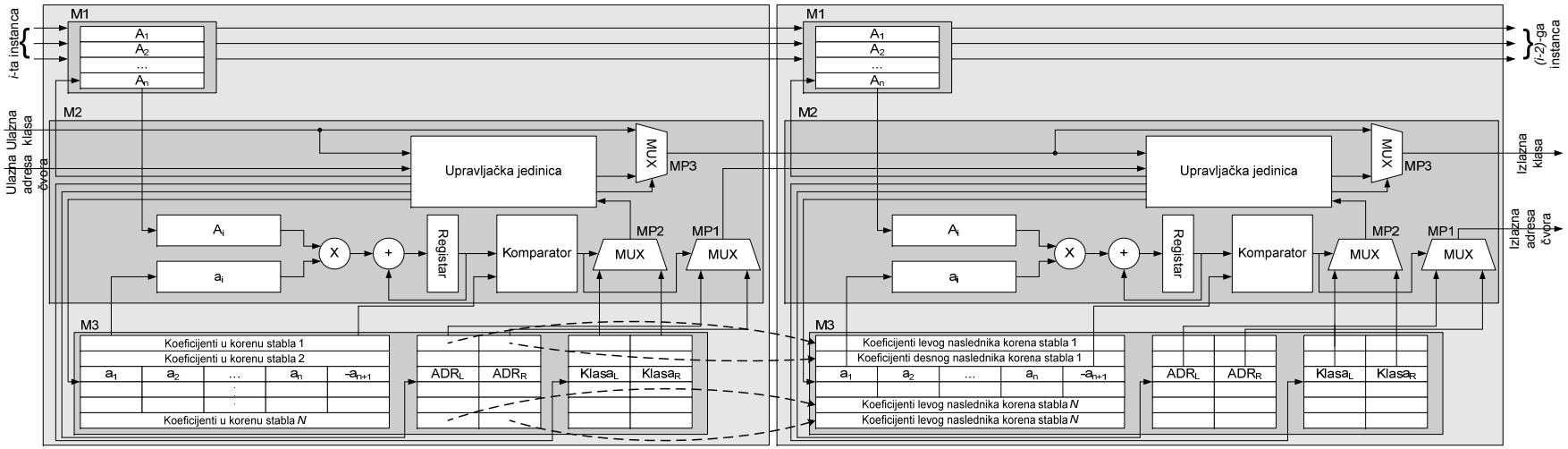
3.1 Modifikovane SMPL i UN arhitekture za serijsku evaluaciju članova ansambla

Kao što je već rečeno, ukoliko se modul za određivanje predikcija članova ansambla realizuje serijski, u slučaju da su prediktivni modeli stabla odluka, ovaj modul sastoji se samo iz jednog modula za hardversku implementaciju individualnog stabla odluke, baziranog na modifikovanoj *SMPL* ili *UN* arhitekturi. Modifikacije postojećih *SMPL* i *UN* arhitektura su neophodne, jer je u ovom slučaju potrebno čuvati strukturne informacije o svim stablima koja čine ansambl.

3.1.1 Modifikovana SMPL arhitektura za serijsku evaluaciju članova ansambla

Predložena *SMPL* arhitektura za hardversku implementaciju individualnog stabla odluke mora se u slučaju korišćenja za serijsku evaluaciju članova ansambla stabala odluka modifikovati tako da omogući smeštanje strukturnih podataka o svim stablima iz ansambla. Ovo je moguće ukoliko se memorije u M3 modulu unutar svakog univerzalnog čvora prošire tako da mogu da sadrže relevantne strukturne podatke o svakom stablu iz ansambla. Pored ovoga, potrebno je modifikovati način kodiranja ovih strukturnih podataka, pogotovo u slučaju listova stabla. Takođe je potrebno modifikovati trenutke dovođenja instanci koje se žele klasifikovati pomoću ansambla i način generisanja kontrolnog signala „*Ulazna adresa čvora*“ za univerzalni čvor koji se nalazi u prvoj fazi protočne obrade.

Modifikovana *SMPL* arhitektura koja se može koristiti za serijsku evaluaciju članova ansambla stabala odluka prikazana je na slici 3.2. Na slici 3.2 prikazana je struktura dva modifikovana univerzalna čvora radi lakšeg objašnjenja potrebnih modifikacija.



*Slika **Error! No text of specified style in document.**.2 Modifikovana SMPL arhitektura za serijsku evaluaciju članova ansambla stabala odluka*

Kao što je već rečeno, najveće modifikacije koje je potrebno izvršiti odnose se na način popunjavanja memorija unutar M3 modula i njihovu veličinu. Veličina memorija za smeštanje koeficijenata hiperravni, adresa čvorova naslednika i klasa asociranih čvorovima naslednicima mora se povećati tako da memorije u svakom od nivoa budu dovoljno velike da se u njih mogu smestiti strukturni podaci o svim čvorovima koji se nalaze na odgovarajućem nivou u svakom stablu iz ansambla. Na primer, memorije unutar M3 modula iz prvog univerzalnog čvora moraju biti dovoljno velike da se u njih mogu smestiti strukturni podaci o korenima svakog stabla iz ansambla, odnosno strukturni podaci o ukupno *N* čvorova, ukoliko sa *N* označimo broj stabala koja čine ansambl. Memorije iz M3 modula unutar drugog univerzalnog čvora moraju biti dovoljno velike da se u njih mogu smestiti strukturni podaci o svim čvorovima koji se nalaze na drugom nivou u svakom od stabala iz ansambla. U najgorem slučaju broj ovih čvorova iznosi *2N*, mada u praksi ovaj broj može biti i manji. Prilikom implementacije konkretnog ansambla, veličine ovih memorija direktno zavise od broja čvorova koji se nalaze na odgovarajućem nivou unutar svakog od stabala koja čine ansambl. Ovaj broj se lako može odrediti prebrojavanjem čvorova lociranih na odgovarajućem nivou u svakom od stabala iz ansambla. Način popunjavanja memorija je takav da se na sukcesivnim adresama nalaze podaci o čvorovima iz istog stabla počevši od prvog a zaključno sa poslednjim stablom iz ansambla. Na ovaj način se svaka memorija može posmatrati kao da je izdeljena na *N* blokova različite veličine, formiranih od sukcesivnih memorijskih lokacija. Prvi blok uvek čine podaci o čvorovima koji pripadaju prvom stablu, sledeći blok sadrži podatke o čvorovima koji pripadaju drugom stablu iz ansambla, sve do poslednjeg bloka koji sadrži podatke o čvorovima koji pripadaju poslednjem, *N*-tom stablu iz ansambla.

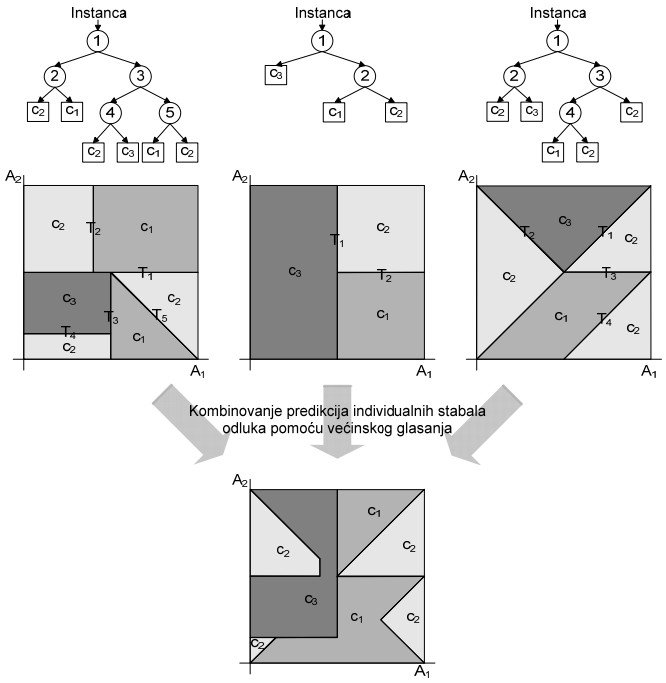
Nakon što je određena veličina memorija unutar M3 modula za svaki od univerzalnih čvorova, neophodno je odrediti sadržaj memorija za smeštanje adresa čvorova naslednika. Obzirom da se u sledećem univerzalnom čvoru nalaze strukturni podaci o čvorovima iz ukupno *N* stabala odluka, potrebno je vrlo pažljivo odrediti adrese na kojima su smešteni strukturni podaci o dva čvora naslednika za posmatrani čvor. Ovo nije veliki problem jer su podaci o čvorovima koji pripadaju istom stablu grupisani u blokove, tako da se adresa čvora naslednika dobija tako što se na relativnu adresu (koja zapravo predstavlja redni broj čvora u stablu) doda odgovarajući ofset koji određuje koliko lokacija treba „preskočiti“ od početka memorije da bi se stiglo do lokacije prvog čvora iz posmatranog stabla.

Sam princip rada modifikovane *SMPL* arhitekture je nepromenjen. Svaka instanca se obrađuje pomoću protočne obrade kroz ukupno *M* faza, gde *M* predstavlja dubinu najdubljeg stabla odluke iz ansambla. Razlika u odnosu na *SMPL* arhitekturu za implementaciju individualnog stabla odluke je u vremenskim trenucima dovođenja novih instanci na ulaz čitavog sistema. Obzirom da je instancu potrebno klasifikovati pomoću *N* individualnih stabala, istu instancu potrebno je dovesti na ulaz sistema *N* puta uzastopno. Prilikom svakog novog dovođenja potrebno je uvećati vrednost ulaznog signala „*Ulazna adresa čvora*“ da bi univerzalni čvor u prvoj fazi protočne obrade koristio čvor sledećeg stabla iz ansambla. Kada se nova instanca prvi put dovede na ulaz sistema ulazni signal „*Ulazna adresa čvora*“ treba da ima vrednost nula, i u tom slučaju će početi evaluacija instance korišćenjem korena prvog stabla iz ansambla. Prilikom narednog dovođenja instance ulazni signal „*Ulazna adresa čvora*“ treba da ima vrednost jedan da bi se izvršila evaluacija instance korišćenjem korena drugog stabla iz ansambla. Prilikom poslednjeg dovođenja instance ulazni signal „*Ulazna adresa čvora*“ treba da ima vrednost *N-1* da bi se instanca evaluirala pomoću korena *N*-tog stabla iz ansambla. Tek nakon *N* dovođenja iste instance može se preći na narednu instancu za koju se mora primeniti isti postupak.

Postupak inicijalizacije memorija unutar M3 modula i sam način rada modifikovane *SMPL* arhitekture najlakše je ilustrovati pomoću konkretnog primera.

3.1.2 Primer korišćenja modifikovane SMPL arhitekture

Kao primer pomoću kojega će biti ilustrovan način rada modifikovane *SMPL* arhitekture posmatrajmo ansambl sastavljen od ukupno tri neortogonalna stabla odluke prikazan na slici 3.3. Kao što se sa slike 3.3 može videti, ansambl se koristi za rešavanje klasifikacionog problema opisanog pomoću dva atributa, A_1 i A_2, pri čemu se instanca može klasifikovati u jednu od ukupno tri klase, c_1, c_2 i c_3. Na samoj slici prikazana je struktura svakog stabla odluke kao i klasifikacija prostora atributa pomoću njega. Na dnu slike prikazana je i klasifikacija prostora atributa pomoću ansambla u slučaju da se predikcije pojedinačnih stabala kombinuju korišćenjem većinskog odlučivanja.



*Slika **Error! No text of specified style in document.**.3 Ansambl od tri neortogonalna stabla odluke i klasifikacija prostora atributa*

Slika 3.3 ilustruje i način na koji se vrši particija prostora atributa pomoću ansambla. U slučaju većinskog odlučivanja instanca se klasifikuje u klasu za koju glasa najveći broj članova. Kao što se na slici 3.3 može videti, particija prostora pomoću ansambla može biti izrazito složena iako su particije pojedinačnih članova relativno jednostavne. Ovo i jeste glavna prednost prilikom korišćenja ansambla.

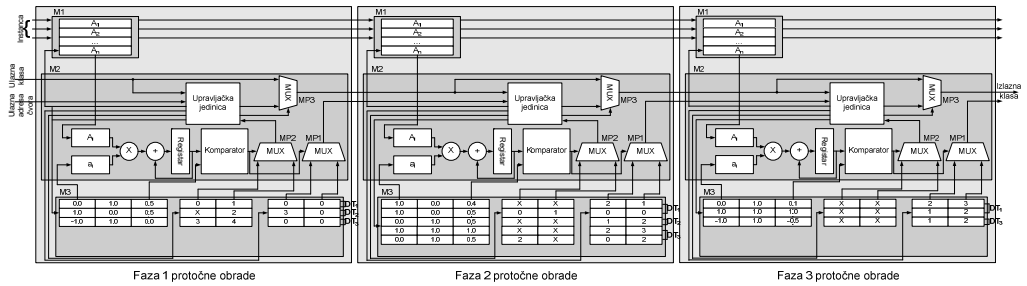
Tri stabla odluke koja čine posmatrani ansambl imaju 5, 2 i 4 čvora respektivno i dubine od 3, 2 i 3 nivoa. Maksimalna dubina stabla u ansamblu iznosi 3 nivoa, tako da će prilikom hardverske implementacije biti neophodno koristiti tri univerzalna čvora organizovana u tri faze protočne obrade. Ukupan broj čvorova lociranih u različitim nivoima iznosi: 3 čvora u prvom nivou (3 korena), 4 čvora u drugom nivou i 3 čvora u trećem nivou. Testovi asocirani svakom od čvorova definisani su na sledeći način.

$$Stablo\,1:\begin{cases} T_1: 0\cdot A_1+1\cdot A_2-0.5=0 \\ T_2: 1\cdot A_1+0\cdot A_2-0.4=0 \\ T_3: 1\cdot A_1+0\cdot A_2-0.5=0 \\ T_4: 0\cdot A_1+1\cdot A_2-0.1=0 \\ T_5: 1\cdot A_1+1\cdot A_2-1=0 \end{cases}$$

$$Stablo\,2:\begin{cases} T_1: 1\cdot A_1+0\cdot A_2-0.5=0 \\ T_2: 0\cdot A_1+0\cdot A_2-0.5=0 \end{cases} \qquad (3.1)$$

$$Stablo\,3:\begin{cases} T_1: -1\cdot A_1+1\cdot A_2+0=0 \\ T_2: 1\cdot A_1+1\cdot A_2-1=0 \\ T_3: 0\cdot A_1+1\cdot A_2-0.5=0 \\ T_4: -1\cdot A_1+1\cdot A_2+0.5=0 \end{cases}$$

Izgled sistema za serijsku evaluaciju članova ansambla baziranog na modifikovanoj *SMPL* arhitekturi prikazan je na slici 3.4. Na ovoj slici takođe su prikazane veličina i sadržaj memorija unutar M3 modula za svaki od ukupno tri univerzalna čvora.

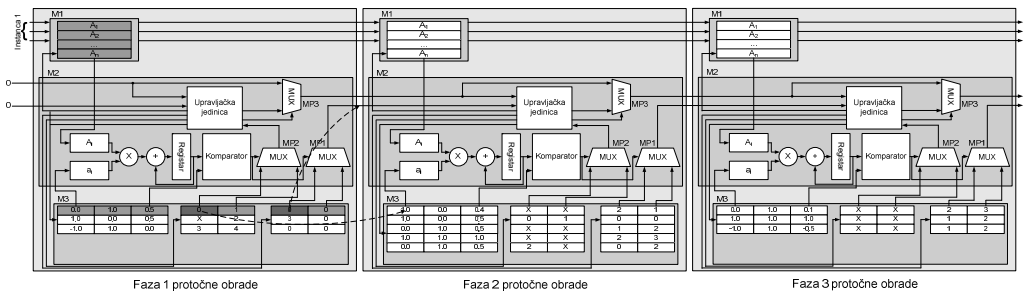


*Slika **Error! No text of specified style in document.**.4 Detaljna struktura sistema za hardversku implementaciju ansambla sa slike 3.3 baziranog na modifikovanoj* SMPL *arhitekturi*

Veličina memorija iz M3 modula unutar prvog univerzalnog čvora dovoljna je za smeštanje informacija o tri korena. Kao što je ranije rečeno, svaka od memorija organizovana je u ukupno tri bloka pri čemu je u ovom slučaju svaki od njih iste veličine i čuva podatke o jednom korenu. Slično, veličina memorija iz M3 modula unutar drugog univerzalnog čvora odabrana je tako da se u njih mogu smestiti informacije o ukupno 5 čvorova koji se nalaze u drugom nivou svakog od tri stabla koja čine ansambl. I ove memorije organizovane su u tri bloka, ovaj put različitih veličina. Prvi blok sadrži informacije o ukupno dva čvora, drugi sadrži informacije o jednom čvoru dok treći blok sadrži informacije o dva čvora. Veličina blokova odgovara broju čvorova lociranih u drugom nivou za svako od tri stabla. Veličinu memorija iz M3 modula unutar trećeg univerzalnog čvora dovoljna je za smeštanje podataka o ukupno 3 čvora. Ova memorija organizovana je u dva bloka, prvi koji čuva podatke o dva čvora iz prvog stabla i drugi koji čuva podatke o jednom čvoru iz trećeg stabla. U ovom slučaju ne postoji blok asociran drugom stablu, jer u ovom stablu nema ni jednog čvora na trećem nivou. Ovi blokovi se takođe mogu videti na slici 3.4.

Glavna razlika u odnosu na osnovnu *SMPL* arhitekturu leži u načinu formiranja adresa čvorova naslednika. Na primer, posmatrajmo memoriju adresa čvorova naslednika unutar prvog univerzalnog čvora. Kao adrese čvorova naslednika korena prvog stabla navedene su vrednosti 0 i 1 jer se upravo u prvoj i drugoj vrsti nalaze podaci o čvorovima naslednicima u M3 modulu iz drugog univerzalnog čvora. Adrese kao i uvek počinju od vrednosti nula. Te dve vrste čine prvi blok asociran prvom stablu odluke. Ako pogledamo adrese čvorova naslednika za koren drugog stabla (druga vrsta u M3 modulu unutar prvog univerzalnog čvora) vidimo da one imaju vrednosti X i 2. Obzirom da je levi naslednik korena drugog stabla zapravo list, njegovu adresu možemo postaviti na proizvoljnu vrednost. Adresa desnog naslednika je zapravo 2, jer se u trećoj vrsti unutar M3 modula iz drugog univerzalnog čvora nalaze podaci o tom čvoru. Od te vrste počinje drugi blok koji je pridružen drugom stablu odluke. Na kraju, treći blok, asociran trećem stablu odluke počinje od četvrte i završava sa petom vrstom unutar M3 modula iz drugog univerzalnog čvora. Upravo zbog toga su kao adrese čvorova naslednika korena trećeg stabla odluke unutar prvog univerzalnog čvora navedene vrednosti 3 i 4. Po ovom principu određene su i sve ostale adrese čvorova naslednika u preostala dva univerzalna čvora.

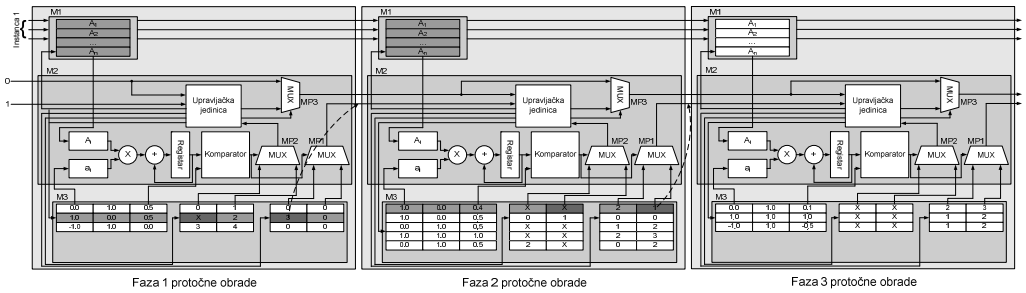
Razmotrimo sada način klasifikacije instanci pomoću sistema sa slike 3.4. U prvom ciklusu na ulaz M1 modula iz prvog univerzalnog čvora dovodimo instancu koju želimo klasifikovati pomoću ansambla. Označimo ovu instancu sa instanca 1. Istovremeno na ulaze „*Ulazna klasa*“ i „*Ulazna adresa čvora*“ potrebno je dovesti vrednosti 0 i 0. Ukoliko je vrednost ulaza „*Ulazna klasa*“ jednaka nuli, univerzalni čvor u prvoj fazi protočne obrade će raditi u režimu evaluacije i odrediće položaj instance u odnosu na hiperravan definisanu u čvoru čija se adresa nalazi na ulazu „*Ulazna adresa čvora*“. Obzirom da je vrednost ovog ulaznog signala takođe postavljena na nulu, evaluiraće se hiperravan asocirana korenu prvog stabla iz ansambla, upravo ono što i želimo. Nakon odgovarajućeg broja taktova, koji zavisi od toga na koji način se vrši određivanje položaja instance u odnosu na hiperravan, univerzalni čvor iz prve faze protočne obrade u stanju je da izračuna vrednosti koje je potrebno proslediti narednom čvoru. Ova situacija je prikazana na slici 3.5. Tamno sivom bojom označena je vrsta unutar M3 modula koja se trenutno koristi.



*Slika **Error! No text of specified style in document.**.5 Situacija u klasifikacionom sistemu nakon završetka prvog ciklusa*

Neka je pozicija instance 1 u odnosu na hiperravan definisanu u korenu prvog stabla odluke takva da je u narednom ciklusu potrebno posetiti levog naslednika korena, čvor broj 2. Podaci o ovom čvoru smešteni su u prvoj vrsti unutar M3 modula iz drugog univerzalnog čvora. Zbog toga izabrana adresa čvora naslednika ima vrednost 0 (tamno sivo polje unutar M3 modula iz prvog univerzalnog čvora na slici 3.5). Ova vrednost prosleđuje se kroz izlaz „*Izlazna adresa čvora*“ ka narednom univerzalnom čvoru. Istovremeno se preko izlaza „*Izlazna klasa*“ prosleđuje vrednost 0, jer levi naslednik korena prvog stabla nije list, pa instanca 1 još uvek nije klasifikovana.

U sledećem ciklusu, instanca 1 ponovo se dovodi na ulaz prvog univerzalnog čvora, ali se ovaj put menja vrednost ulaznog signala „*Ulazna adresa čvora*“ na vrednost 1. Ovo znači da će se u tekućem ciklusu instanca 1 evaluirati korišćenjem korena drugog stabla. Istovremeno će se instanca 1 u drugoj fazi protočne obrade evaluirati u čvoru broj 2 prvog stabla odluke. Aktivne memorijske lokacije kao i vrednosti izlaznih signala koje će biti prosleđene ka narednim fazama protočne obrade prikazane su različitim nijansama sive boje na slici 3.6.

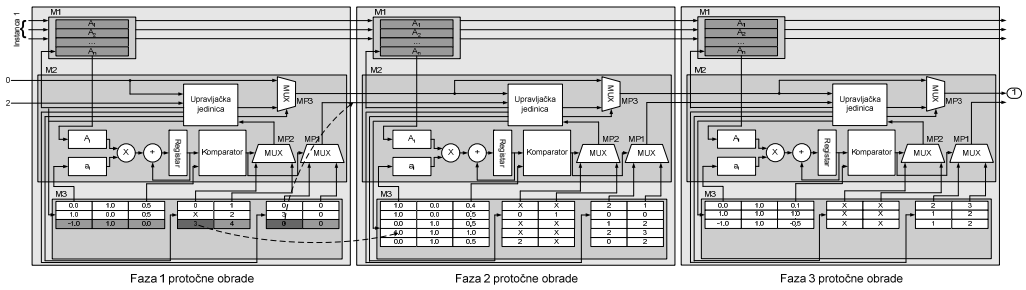


*Slika **Error! No text of specified style in document.**.6 Situacija u klasifikacionom sistemu nakon završetka drugog ciklusa*

Nakon završetka drugog ciklusa imamo sledeću situaciju. Prvi univerzalni čvor završio je određivanje pozicije instance 1 u odnosu na hiperravan definisanu u korenu drugog stabla odluke. Neka je pozicija takva da je potrebno posetiti levog naslednika. Obzirom da je levi naslednik korena drugog stabla zapravo list, instanca 1 se već može klasifikovati pomoću drugog stabla u klasu c_3. Na izlaz „*Izlazna adresa čvora*“ prosleđuje se proizvoljna vrednost, jer dalja evaluacija instance 1 pomoću drugog stabla nije potrebna. Na izlaz „*Izlazna klasa*“ postavlja se vrednost 3, koja označava klasu u koju je instancu 1 klasifikovalo drugo stablo.

Istovremeno, univerzalni čvor u drugoj fazi protočne obrade nastavlja klasifikaciju instance 1 pomoću prvog stabla, određujući njen položaj u odnosu na hiperravan definisanu u čvoru 2. Neka je pozicija instance 1 takva da je u sledećoj fazi potrebno posetiti desnog naslednika čvora 2 prvog stabla. Pošto je ovaj naslednik zapravo list, instanca 1 može se klasifikovati pomoću prvog stabla u klasu c_1. Na izlaze „*Izlazna klasa*“ i „*Izlazna adresa čvora*“ prosleđuju se vrednosti 1 i X respektivno.

U trećem ciklusu, instanca 1 se ponovo dovodi na ulaz prvog univerzalnog čvora, ali se ovaj put ulazni signal „*Ulazna adresa čvora*“ ima vrednost 2. Ovo znači da će se u tekućem ciklusu instanca 1 evaluirati korišćenjem korena trećeg stabla. Obzirom da je instanca 1 već klasifikovana pomoću prvog i drugog stabla odluke, univerzalni čvorovi u drugoj i trećoj fazi protočne obrade rade u režimu propuštanja i samo prosleđuju ulazne signale na odgovarajuće izlaze. Aktivne memorijske lokacije kao i vrednosti izlaznih signala koje će biti prosleđene ka narednim fazama protočne obrade prikazane su različitim nijansama sive boje na slici 3.7.



*Slika **Error! No text of specified style in document.**.7 Situacija u klasifikacionom sistemu nakon završetka trećeg ciklusa*

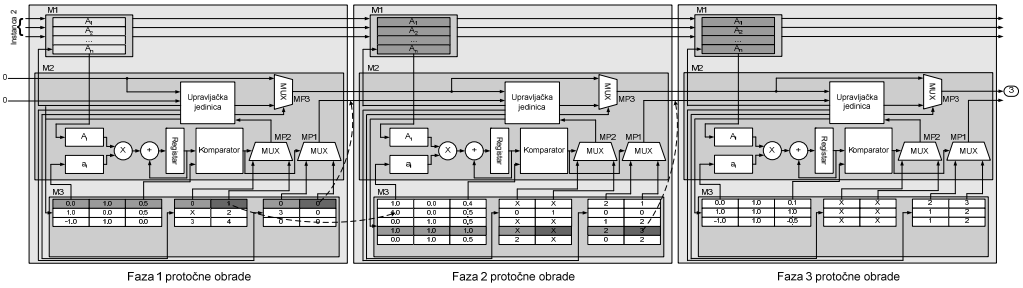
Nakon završetka trećeg ciklusa imamo sledeću situaciju. Prvi univerzalni čvor završio je određivanje pozicije instance 1 u odnosu na hiperravan definisanu u korenu trećeg stabla odluke. Neka je pozicija takva da je potrebno posetiti levog naslednika, čvor 2. Obzirom da levi naslednik korena trećeg stabla u ovom slučaju nije list na izlaz „*Izlazna klasa*“ postavlja se vrednost 0, da bi se univerzalnom čvoru u drugoj fazi protočne obrade signaliziralo da

tekuća instanca još nije klasifikovana. Na izlaz „*Izlazna adresa čvora*“ postavlja se vrednost 3, jer se je to adresa na kojoj se čuvaju informacije o čvoru 2 iz trećeg stabla u univerzalnom čvoru iz druge faze protočne obrade.

Istovremeno, univerzalni čvor u drugoj fazi radi u režimu prosleđivanja pošto je instanca 1 već klasifikovana pomoću drugog stabla. U ovom slučaju univerzalni čvor u drugoj fazi samo prosleđuje vrednosti ulaznih signala na odgovarajuće izlaze.

U istom ciklusu i univerzalni čvor iz treće faze radi u režimu propuštanja pošto je instanca 1 već klasifikovana i pomoću prvog stabla odluke. Usled toga, na kraju trećeg ciklusa, na izlazu čitavog sistema pojaviće se klasa u koje je prvo stablo odluke iz ansambla klasifikovalo instancu 1, klasa c_1.

U četvrtom ciklusu, na ulaz prvog univerzalnog čvora dovodi se naredna instanca, instanca 2. Obzirom da se u ovom primeru ansambl sastoji od tri stabla odluke, svaka instanca se mora dovoditi na ulaz sistema za klasifikaciju ukupno tokom tri uzastopna ciklusa. Pošto su tri ciklusa u kojima je na ulaz sistema dovođena instanca 1 protekla, u ovom ciklusu moguće je na ulaz dovesti sledeću instancu koju je potrebno klasifikovati. Ulazni signal „*Ulazna adresa čvora*“ treba da ima vrednost 0 kako bi se u tekućem ciklusu instanca 2 evaluirala korišćenjem korena prvog stabla. Aktivne memorijske lokacije kao i vrednosti izlaznih signala koje će biti prosleđene ka narednim fazama protočne obrade prikazane su različitim nijansama sive boje na slici 3.8.



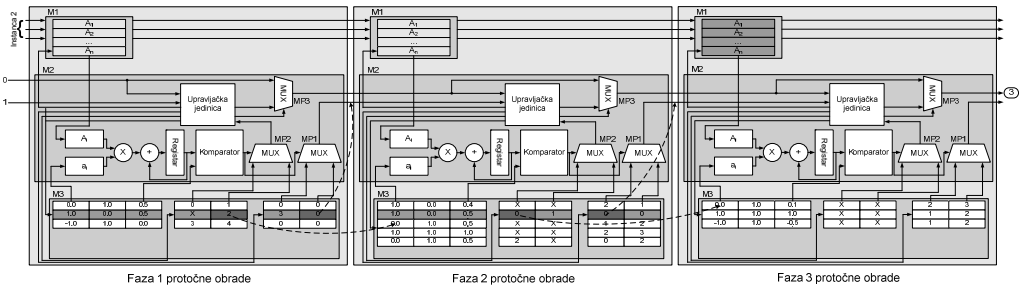
*Slika **Error! No text of specified style in document.**.8 Situacija u klasifikacionom sistemu nakon završetka četvrtog ciklusa*

Nakon završetka četvrtog ciklusa prvi univerzalni čvor je završio određivanje pozicije instance 2 u odnosu na hiperravan definisanu u korenu prvog stabla odluke. Neka je pozicija takva da je potrebno posetiti desnog naslednika, čvor 3. Obzirom da desni naslednik korena prvog stabla u ovom slučaju nije list na izlaz „*Izlazna klasa*“ postavlja se na vrednost 0, da bi se univerzalnom čvoru u drugoj fazi protočne obrade signaliziralo da tekuća instanca još nije klasifikovana. Na izlaz „*Izlazna adresa čvora*“ postavlja se vrednost 1, jer se je to adresa na kojoj se čuvaju informacije o čvoru 3 iz prvog stabla u univerzalnom čvoru iz druge faze.

Istovremeno, univerzalni čvor u drugoj fazi nastavlja sa klasifikacijom instance 1 pomoću trećeg stabla. Univerzalni čvor u drugoj fazi određuje poziciju instance 1 u odnosu na hiperravan definisanu u čvoru 2 iz trećeg stabla. Neka je pozicija instance 1 takva da je u narednom ciklusu potrebno posetiti desnog naslednika čvora 2. Obzirom da je ovaj naslednik list, instanca 1 može se klasifikovati pomoću trećeg stabla u klasu c_3. Na izlaze „*Izlazna klasa*“ i „*Izlazna adresa čvora*“ univerzalnog čvora prosleđuju se vrednosti 3 i X respektivno.

U istom ciklusu univerzalni čvor iz treće faze radi u režimu propuštanja pošto je instanca 1 u prethodnom ciklusu već klasifikovana pomoću drugog stabla odluke. Usled toga, na kraju četvrtog ciklusa, na izlazu čitavog sistema pojaviće se klasa u koje je drugo stablo odluke iz ansambla klasifikovalo instancu 1, a to je u ovom slučaju klasa c_3.

U petom ciklusu, na ulaz prvog univerzalnog čvora ponovo se dovodi instanca 2, ali ovaj put ulazni signal „*Ulazna adresa čvora*“ ima vrednost 1 kako bi se u tekućem ciklusu instanca 2 evaluirala korišćenjem korena drugog stabla. Aktivne memorijske lokacije kao i vrednosti izlaznih signala koje će biti prosleđene ka narednim fazama protočne obrade prikazane su različitim nijansama sive boje na slici 3.9.



*Slika **Error! No text of specified style in document.**.9 Situacija u klasifikacionom sistemu nakon završetka petog ciklusa*

Nakon završetka petog ciklusa imamo sledeću situaciju. Prvi univerzalni čvor završio je određivanje pozicije instance 2 u odnosu na hiperravan definisanu u korenu drugog stabla odluke. Neka je pozicija takva da je potrebno posetiti desnog naslednika, čvor 2. Obzirom da desni naslednik korena prvog stabla u ovom slučaju nije list na izlaz „*Izlazna klasa*“ postavlja se vrednost 0, da bi se univerzalnom čvoru u drugoj fazi protočne obrade signaliziralo da tekuća instanca još nije klasifikovana. Na izlaz „*Izlazna adresa čvora*“ postavlja se na vrednost 2, jer se je to adresa na kojoj se čuvaju informacije o čvoru 2 iz drugog stabla u univerzalnom čvoru iz druge faze.

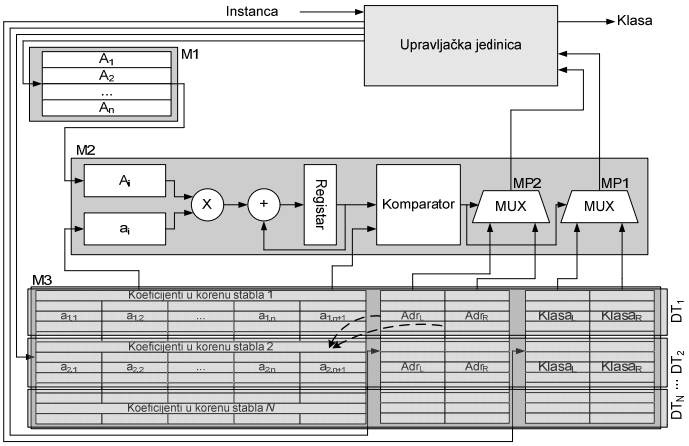
Istovremeno, univerzalni čvor u drugoj fazi nastavlja sa klasifikacijom instance 2 pomoću prvog stabla. Univerzalni čvor u drugoj fazi određuje poziciju instance 2 u odnosu na hiperravan definisanu u čvoru 3 iz prvog stabla. Neka je pozicija instance 1 takva da je u narednom ciklusu potrebno posetiti levog naslednika čvora 2, čvor 4. Na izlaze „*Izlazna klasa*“ i „*Izlazna adresa čvora*“ univerzalnog čvora prosleđuju se vrednosti 0 i 0 respektivno.

U istom ciklusu univerzalni čvor iz treće faze radi u režimu propuštanja pošto je instanca 1 prethodnom ciklusu već klasifikovana pomoću trećeg stabla odluke. Usled toga, na kraju petog ciklusa, na izlazu čitavog sistema pojaviće se klasa u koje je treće stablo odluke iz ansambla klasifikovalo instancu 1, klasa c_3.

Nakon pet ciklusa, svako stablo iz ansambla je klasifikovalo instancu 1. Prvo stablo klasifikovalo je instancu 1 u klasu c_1, dok su drugo i treće stablo klasifikovali instancu 1 u klasu c_3. Ove predikcije bi bile prosleđene ka modulu za kombinovanje predikcija serijski, tokom poslednja tri ciklusa (klasa c_1 na kraju trećeg ciklusa, i klase c_3 na kraju četvrtog i petog ciklusa). Modul za kombinovanje predikcija bi na osnovu ovih podataka mogao da donese odluku o konačnoj klasifikaciji instance 1 pomoću ansambla. Ukoliko bi se koristio algoritma većinskog odlučivanja instanca 1 bila bi klasifikovana u klasu c_3 od strane ansambla.

3.1.3 Modifikovana *UN* arhitektura za serijsku evaluaciju članova ansambla

Kao i u slučaju *SMPL* arhitekture i ranije predložena *UN* arhitektura za hardversku implementaciju individualnih stabala odluka [46], mora se modifikovati da bi se mogla iskoristiti za serijsku evaluaciju članova ansambla. Modifikacije koje je neophodno izvršiti opet se odnose na veličinu memorija iz M3 modula, kao i na način njihove inicijalizacije. Modifikovana *UN* arhitektura prikazana je na slici 3.10.



*Slika **Error! No text of specified style in document.**.10 Modifikovana* UN *arhitektura za serijsku evaluaciju članova ansambla stabala odluka*

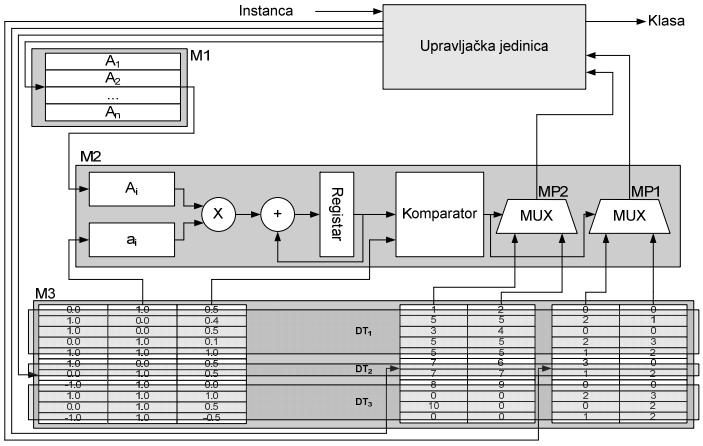
Memorije unutar M3 modula moraju biti dovoljno velike da se u njih mogu smestiti strukturne informacije o svakom stablu iz ansambla, a ne samo o jednom stablu kao ranije. Pored toga način dodeljivanja adresa čvorovima naslednicima, a pogotovo listovima mora da se promeni da bi čitav sistem pravilno funkcionisao. Slično modifikovanoj *SMPL* arhitekturi, i u slučaju modifikovane *UN* arhitekture memorije unutar M3 modula organizovane su blokovski, što je i prikazano na slici 3.10. Prvi blok čine memorijske lokacije u kojima su smeštene strukturne informacije o svim čvorovima iz prvog stabla. Drugi blok čine memorijske lokacije u kojima su smeštene strukturne informacije o svim čvorovim iz drugog stabla, i tako dalje sve do poslednjeg bloka u kojem se nalaze strukturne informacije o svim čvorovima iz poslednjeg, *N*-tog stabla iz ansambla. Veličina blokova ne mora biti ista, jer su po pravilu stabla koja čine ansambl različite veličine.

Obzirom da je svakom stablu iz ansambla pridružen jedan blok, sve adrese čvorova naslednika za posmatrano stablo moraju biti iz opsega koji je pridružen datom bloku. Sa druge strane, adrese koje su pridružene listovima moraju odgovarati početnoj adresi narednog bloka, jer se na toj adresi nalaze podaci o korenu sledećeg stabla iz ansambla. Ova situacija je prikazana na slici 3.10 isprekidanim strelicama. Kao adresu koja je pridružena listovima poslednjeg stabla iz ansambla potrebno je koristiti adresu 0, jer je tekuća instanca evaluirana od strane svakog stabla iz ansambla i potrebno je vratiti se na koren prvog stabla. Ovo je ujedno i indikacija za upravljačku jedinicu da je postupak evaluacije tekuće instance od strane ansambla završen i da se može preći na sledeću instancu.

Za razliku od originale *UN* arhitekture, u slučaju modifikovane *UN* arhitekture sa slike 3.10, klase koje su pridružene čvorovima stabla više se ne mogu birati na proizvoljan način. Kod originalne *UN* arhitekture upravljačka jedinica je informaciju da je instanca klasifikovana dobijala analizirajući adresu narednog čvora koga treba posetiti. U slučaju kada je ona bila nula, to je bila indikacija da je tekuća instanca klasifikovana i da se trenutna vrednost klase, pročitana iz memorije za smeštanje asociranih klasa, može proslediti na izlaz čitavog sistema. U slučaju modifikovane *UN* arhitekture ovo više nije moguće, jer adresa narednog čvora više nije jednaka nuli kada se dostigne neki od listova, već ona ima vrednost korena narednog stabla iz ansambla. Zbog toga se informacija da je tekuće stablo klasifikovalo tekuću instancu mora proslediti upravljačkoj jedinici na neki drugi način. Srećom, ovo je moguće izvesti izborom drugačije vrednosti za klasu koja je asocirana čvorovima. Umesto da se ova vrednost postavlja proizvoljno, ona uvek treba da bude jednaka nuli, slično kao kod originalne *SMPL* arhitekture. Obzirom da se za svaku od postojećih klasa koriste brojevi različiti od nule, vrednost nula ostaje neiskorišćena pa se može iskoristiti da označi da je reč o čvoru a ne o listu tekućeg stabla. Na ovaj način upravljačka jedinica može prepoznati da li je tekuća instanca stigla do lista, prosto ispitujući tekuću vrednost klase.

3.1.4 Primer korišćenja modifikovane *UN* arhitekture

Način inicijalizacije memorija unutar M3 modula kao i sam način rada modifikovane *UN* arhitekture najbolje se može razumeti na konkretnom primeru. Kao i u slučaju ilustracije načina rada modifikovane *SMPL* koristićemo primer prikazan na slici 3.3. U tom primeru, potrebno je realizovati ansambl od ukupno tri neortogonalna stabla odluke, koji se koristi za klasifikaciju instanci opisanih pomoću dva atributa u jednu od tri moguće klase. Slika 3.11 prikazuju izgled sistema baziranog na modifikovanoj *UN* arhitekturi koji realizuje posmatrani ansambl.

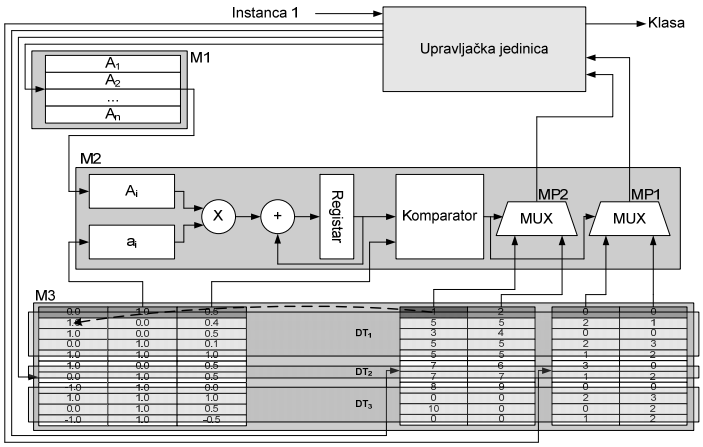


*Slika **Error! No text of specified style in document.**.11 Detaljna struktura sistema za hardversku implementaciju ansambla sa slike 3.3 baziranog na modifikovanoj* UN *arhitekturi*

Za ovaj konkretni primer, veličina memorija unutar M3 modula mora biti dovoljna da bi se unutar njih mogli smestiti podaci o ukupno 11 čvorova; 5 čvorova koji čine prvo stablo odluke, 2 čvora koja čine drugo stablo odluke i 4 čvora koja čine treće stablo odluke. Ovo su ujedno i veličine blokova na koje su podeljene memorije. Prvi blok počinje od adrese 0, drugi od adrese 5, a treći od adrese 7.

Adrese čvorova naslednika unutar istog stabla uvek imaju vrednosti iz opsega adresa koji je asociran posmatranom bloku, odnosno stablu. Na primer, adrese čvorova naslednika za prvo stablo imaju vrednosti od 1 do 4, što se može videti na slici 3.11. Adrese u slučaju listova tekućeg stabla trebaju imati vrednosti koje upućuju na koren sledećeg stabla odluke. U našem primeru, prvo stablo odluke ima ukupno 6 listova i za svaki od njih adresa narednog čvora koji treba posetiti iznosi 5, jer je to prva adresa asocirana drugog bloku, na kojoj se nalaze podaci o korenu drugog stabla. Vrednosti za klasu svakog od čvorova bilo kog stabla moraju biti nula, da bi bilo moguće raspoznati kada je tekuća instanca klasifikovana.

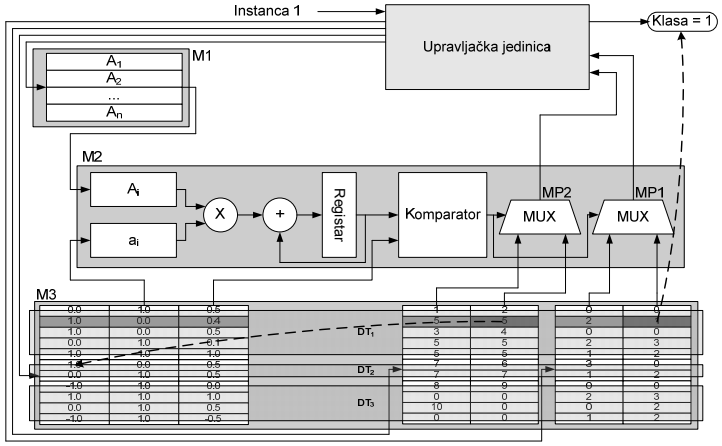
Razmotrimo sada način klasifikacije instanci pomoću sistema sa slike 3.11. U prvom ciklusu na ulaz sistema sa slike 3.11 dovodimo instancu 1. Tokom prvog ciklusa sistem će odrediti poziciju instance 1 u odnosu na hiperravan koja je definisana u korenu prvog stabla, koristeći podatke iz prve vrste unutar memorija iz M3 modula. Ova situacija prikazana je na slici 3.12. Tamno sivom bojom označena je vrsta unutar M3 modula koja je trenutno adresirana.



*Slika **Error! No text of specified style in document.**.12 Situacija u klasifikacionom sistemu nakon završetka prvog ciklusa*

Neka je pozicija instance 1 u odnosu na hiperravan definisanu u korenu prvog stabla odluke takva da je u narednom ciklusu potrebno posetiti levog naslednika korena, čvor broj 2. Podaci o ovom čvoru smešteni su u drugoj vrsti unutar M3 modula. Zbog toga izabrana adresa čvora naslednika ima vrednost 1 (tamno sivo polje unutar M3 modula na slici 3.12). Ova vrednost nalazi se na izlazu multipleksera MP2 i u narednom ciklusu biće korišćena za adresiranje memorija unutar M3 modula. Istovremeno se na izlazu MP1 multipleksera nalazi vrednost 0, što upravljačka jedinica tumači kao znak da instanca 1 još uvek nije klasifikovana pomoću prvog stabla odluke.

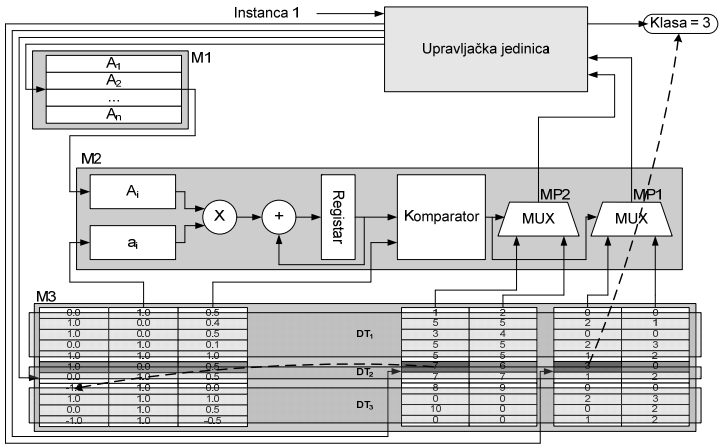
U sledećem ciklusu, računa se pozicija instance 1 u odnosu na hiperravan definisanu u čvoru 2 prvog stabla. Aktivne memorijske lokacije kao i vrednosti od interesa prikazane su različitim nijansama sive boje na slici 3.13.



*Slika **Error! No text of specified style in document.**.13 Situacija u klasifikacionom sistemu nakon završetka drugog ciklusa*

Nakon završetka drugog ciklusa imamo sledeću situaciju. Neka je pozicija instance 1 u odnosu na hiperravan definisanu u čvoru 2 prvog stabla odluke takva da je u narednom ciklusu potrebno posetiti desnog naslednika, koji je u ovom slučaju list. Obzirom da je reč o listu, izlaz multipleksera MP1 imaće vrednost 1, vrednost klase koja je pridružena ovom listu, a to je klasa c_1. Obzirom da je ova vrednost različita od nule, upravljačka jedinica zaključuje da je instanca 1 klasifikovana pomoću prvog stabla i ovu vrednost prosleđuje na izlaz „*Klasa*“. Istovremeno se na izlazu MP2 multipleksera nalazi vrednost 5, koja predstavlja adresu korena drugog stabla odluke, pošto je instanca 1 klasifikovana pomoću prvog stabla, pa se u narednom ciklusu može započeti proces klasifikacije pomoću drugog stabla iz ansambla.

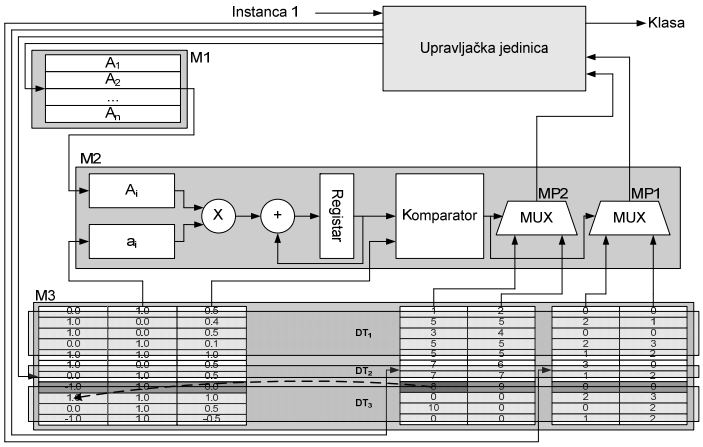
U sledećem ciklusu, računa se pozicija instance 1 u odnosu na hiperravan definisanu u korenu drugog stabla. Aktivne memorijske lokacije kao i vrednosti od interesa prikazane su različitim nijansama sive boje na slici 3.14.



*Slika **Error! No text of specified style in document.**.14 Situacija u klasifikacionom sistemu nakon završetka trećeg ciklusa*

Nakon završetka trećeg ciklusa imamo sledeću situaciju. Neka je pozicija instance 1 u odnosu na hiperravan definisanu u korenu drugog stabla odluke takva da je u narednom ciklusu potrebno posetiti levog naslednika, koji je u ovom slučaju list. Obzirom da je reč o listu, izlaz multipleksera MP1 opet će imati vrednost različitu od nule, u ovom slučaju to će biti vrednost 3, vrednost klase koja je pridružena ovom listu, klasa c_3. Obzirom da je ova vrednost različita od nule, upravljačka jedinica opet može zaključiti da je instanca 1 klasifikovana i pomoću drugog stabla i ovu vrednost prosleđuje na izlaz „*Klasa*“. Istovremeno se na izlazu MP2 multipleksera nalazi se vrednost 7, koja predstavlja adresu korena trećeg stabla odluke, pošto je instanca 1 klasifikovana i pomoću drugog stabla, pa se u narednom ciklusu može započeti proces klasifikacije pomoću trećeg stabla.

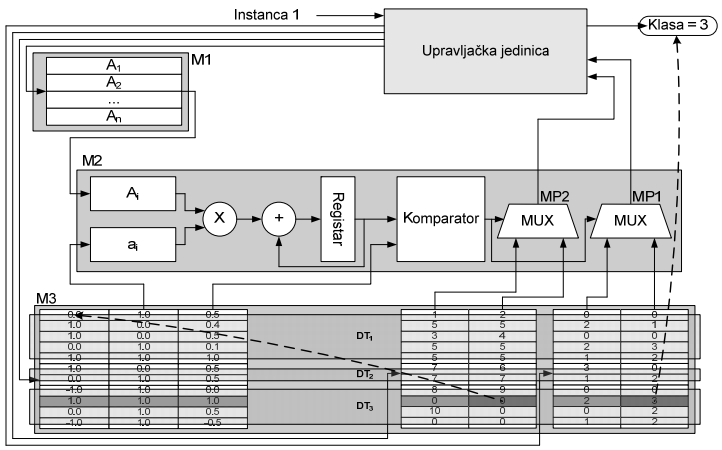
U sledećem ciklusu, računa se pozicija instance 1 u odnosu na hiperravan definisanu u korenu trećeg stabla. Aktivne memorijske lokacije kao i vrednosti od interesa prikazane su različitim nijansama sive boje na slici 3.15.



*Slika **Error! No text of specified style in document.**.15 Situacija u klasifikacionom sistemu nakon završetka četvrtog ciklusa*

Nakon završetka četvrtog ciklusa imamo sledeću situaciju. Neka je pozicija instance 1 u odnosu na hiperravan definisanu u korenu trećeg stabla odluke takva da je u narednom ciklusu potrebno posetiti levog naslednika, a to je u ovom slučaju čvor 2. Podaci o ovom čvoru smešteni su u devetoj vrsti unutar M3 modula. Zbog toga izabrana adresa čvora naslednika ima vrednost 8 (tamno sivo polje unutar M3 modula na slici 3.15). Ova vrednost nalazi se na izlazu multipleksera MP2 i u narednom ciklusu biće korišćena za adresiranje memorija unutar M3 modula. Istovremeno se na izlazu MP1 multipleksera nalazi vrednost 0, što upravljačka jedinica tumači kao znak da instanca 1 još uvek nije klasifikovana pomoću trećeg stabla odluke.

U sledećem ciklusu, računa se pozicija instance 1 u odnosu na hiperravan definisanu u čvoru 2 trećeg stabla. Aktivne memorijske lokacije kao i vrednosti od interesa prikazane su različitim nijansama sive boje na slici 3.16.



*Slika **Error! No text of specified style in document.**.16 Situacija u klasifikacionom sistemu nakon završetka petog ciklusa*

Nakon završetka petog ciklusa imamo sledeću situaciju. Neka je pozicija instance 1 u odnosu na hiperravan definisanu u čvoru 2 trećeg stabla odluke takva da je u narednom ciklusu potrebno posetiti desnog naslednika, koji je u ovom slučaju list. Obzirom da je reč o listu, izlaz multipleksera MP1 imaće vrednost 3, vrednost klase koja je pridružena ovom listu, a to je klasa c_3. Kako je ova vrednost različita od nule, upravljačka jedinica zaključuje da je instanca 1 klasifikovana pomoću trećeg stabla i ovu vrednost prosleđuje na izlaz „*Klasa*“. Istovremeno se na izlazu MP2 multipleksera nalazi se vrednost 0, koja predstavlja adresu korena prvog stabla odluke, jer, pošto je instanca 1 klasifikovana pomoću sva tri stabla odluke, u narednom ciklusu se može započeti proces klasifikacije nove instance pomoću prvog stabla.

Nakon pet ciklusa, svako stablo iz ansambla je klasifikovalo instancu 1. Prvo stablo klasifikovalo je instancu 1 u klasu c_1, dok su drugo i treće stablo klasifikovali instancu 1 u klasu c_3. Ove predikcije bi bile prosleđene ka modulu za kombinovanje predikcija serijski, tokom ciklusa 2, 3 i 5, (klasa c_1 na kraju drugog ciklusa, i klase c_3 na kraju trećeg i petog ciklusa). Ovde se može uočiti i razlika u brzini klasifikacije instanci pomoću *SMPL* i *UN* arhitektura. U slučaju *SMPL* arhitekture klasifikacije instanci se prosleđuju nakon svakog ciklusa, nakon nekog inicijalnog kašnjenja koje odgovara dubini sistema za protočnu obradu. Za ovaj isti primer, klasifikacija instance 1 od strane tri stabla iz ansambla pojavila se na izlazu sistema sa slike 3.4 u ciklusima 3, 4 i 5. Za razliku od *SMPL* arhitekture, *UN* arhitektura nije u stanju da na kraju svakog ciklusa izračuna novu klasifikaciju instance. U ovom konkretnom primeru informacije o klasifikaciji instance 1 od strane tri stabla iz ansambla prosleđene su u ciklusima 2, 3 i 5. Kod *UN* arhitekture ne može se garantovati da će se nakon nekog početnog kašnjenja klasifikacije instanci pojavljivati nakon svakog ciklusa.

Modul za kombinovanje predikcija bi na osnovu ovih podataka mogao da donese odluku o konačnoj klasifikaciji instance 1 pomoću ansambla. Ukoliko bi se koristio algoritam većinskog odlučivanja instanca 1 bila bi klasifikovana u klasu c_3 od strane ansambla.

3.2 Arhitekture za kombinovanje predikcija individualnih prediktivnih modela

Kao što je rečeno u uvodu, da bi se implementirao ansambl prediktivnih modela, bilo softverski ili hardverski, moraju se realizovati dva modula. Prvi modul zadužen je za evaluaciju prediktivnih modela koji čine ansambl i određivanje njihovih predikcija klase u koju je potrebno klasifikovati tekuću instancu. Drugi modul zadužen je za donošenje kolektivne odluke na osnovu predikcija individualnih članova ansambla. Do sada su bile predstavljene arhitekture za paralelnu i serijsku evaluaciju stabala odluka koja čine ansambl (glava 3.1). U ovoj glavi biće predstavljene arhitekture za efikasnu realizaciju najčešće korišćenih pravila kombinovanja predikcija individualnih članova ansambla u cilju donošenja zajedničke odluke o klasi u koju treba klasifikovati tekuću instancu. Bitno je napomenuti da se predložene arhitekture, za razliku od predloženih arhitektura za evaluaciju članova ansambla koje su se mogle primeniti samo za realizaciju stabala odluka, mogu koristiti za kombinovanje predikcija individualnih prediktivnih modela bez obzira na njihovu vrstu. U tom smislu su predložene arhitekture univerzalne i mogu se koristiti za realizaciju modula za kombinovanje prilikom hardverske implementacije proizvoljnog ansambla.

U ovoj glavi biće predložene arhitekture za serijsku i paralelnu implementaciju sledećih algoritama za kombinovanje predikcija:

- većinsko odlučivanje u tri varijante (jednoglasno odlučivanje, odlučivanje prostom većinom i većinsko odlučivanje),

- težinsko većinsko odlučivanje,
- *behavior knowledge space* (BKS) algoritam.

Serijske arhitekture mogu se kombinovati sa ranije predloženim *SMPL* i *UN* arhitekturama za serijsku evaluaciju članova ansambla u slučaju serijske implementacije ansambla stabala odluka. Slično, paralelne arhitekture mogu se kombinovati sa ranije predloženim *SMPL* i *UN* arhitekturama za paralelnu evaluaciju članova ansambla u cilju paralelne implementacije ansambla stabala odluka.

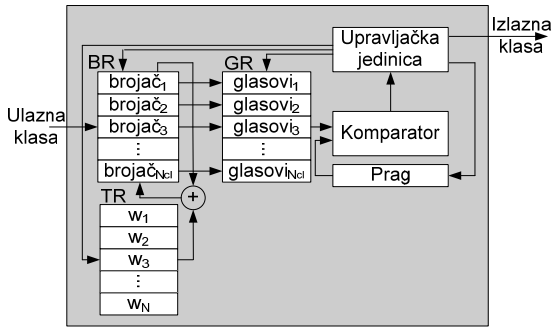
3.2.1 Serijske arhitekture

Implementacija različitih varijanti većinskog odlučivanja

Na slici 3.17 prikazana je *MV_S* arhitektura pomoću koje je moguće realizovati sledeće algoritme za kombinovanje predikcija individualnih članova ansambla:

- jednoglasno odlučivanje,
- odlučivanje prostom većinom,
- većinsko odlučivanje,
- težinsko većinsko odlučivanje.

MV_S arhitektura bazira se na serijskoj evaluaciji predikcija individualni prediktivnih modela radi donošenja kolektivne odluke. Predikcije individualnim prediktivnih modela učitavaju se jedna po jedna tokom *N* taktova, a nakon toga se donosi kolektivna odluka.



Slika ***Error! No text of specified style in document.****.17* MV_S *arhitektura za serijsku implementaciju algoritama većinskog i težinskog većinskog odlučivanja*

Princip rada *MV_S* arhitekture je sledeći. Predikcije individualnih prediktivnih modela učitavaju se preko ulaznog signala „*Ulazna klasa*“ tokom *N* taktova. N_{cl} brojačkih registara (modul BR na slici 3.17) čuva podatke o tome koliko je prediktivnih modela „glasalo“ za svaku od mogućih klasa. Svakom od *N* prediktivnih modela iz ansambla pridružen je po jedan težinski registar u kome se čuva težina pridružena posmatranom modelu (modul TR na slici 3.17). Različiti algoritmi zahtevaće različit sadržaj ovih težinskih registara. Vrednost ulaznog signala „*Ulazna klasa*“ koristi se kao adresa za selekciju brojačkog registra, čija se vrednost zatim uvećava za vrednost koja se nalazi u težinskom registru prediktivnog modela čija se predikcija trenutno nalazi na ulazu u sistem. Nakon procesiranja predikcija svih članova ansambla, tekuće vrednosti u brojačkim registrima prebacuju se u registre koji čuvaju glasove koje je svaka od klasa dobila prilikom klasifikacije tekuće instance (modul GR na slici 3.17). Sadržaj brojačkih registara se zatim postavlja na nulu kako bi bili spremni za obradu sledeće instance. Tokom narednih *N* taktova vrednosti u registrima glasova se sekvencijalno porede se unapred definisanim pragom pomoću komparatora. Kada se naiđe na klasu čiji je broj glasova veći od praga upravljačka jedinica prosleđuje broj te klase na izlaz „*Izlazna klasa*“. Ova vrednost zapravo predstavlja klasifikaciju tekuće instance od strane celokupnog ansambla. U slučaju da ne postoji ni jedna klasa čiji je broj glasova veći od praga, vrednost izlaza „*Izlazna klasa*“ ostaje nula, što znači da ansambl ne može doneti odluku o klasi kojoj tekuća instanca pripada.

MV_S arhitektura zapravo ima dve faze obrade koje se mogu odvijati u paraleli, odnosno *MV_S* arhitektura koristi protočnu obradu. U prvoj fazi, koja traje *N* taktova, određuje se broj glasova koja je svaka od mogućih klasa dobila od strane članova ansambla. U drugoj fazi, koja traje N_{cl} taktova, određuje se klasa koja je dobila najveći broj glasova. U slučaju protočne obrade, radi postizanja sinhronizma obe faze moraju imati jednako vreme izvršavanja. Da bi se ispunio ovaj uslov, trajanje druge faze mora se produžiti sa N_{cl} na *N* taktova, što ne predstavlja veliki problem. Nakon ove modifikacije, *MV_S* arhitektura zahteva *N* taktova za klasifikaciju jedne instance, sa inicijalnim kašnjenjem od *2N* taktova.

Pravilnim izborom težina i praga *MV_S* arhitektura se može konfigurisati tako da može da realizuje jednoglasno odlučivanje, odlučivanje prostom većinom, većinsko odlučivanje i težinsko većinsko odlučivanje.

U slučaju jednoglasnog odlučivanja, svi prediktivni modeli iz ansambla moraju se složiti po pitanju klase u koju se klasifikuje tekuća instanca. Drugim rečima, svaki individualni prediktivni model iz ansambla mora tekuću instancu klasifikovati u istu klasu. Da bi *MV_S* arhitektura mogla realizovati ovaj algoritam kombinovanja, sve težinske faktore treba postaviti na jedinicu (TR modul), a vrednost praga treba postaviti na *N-1*.

Kod odlučivanja prostom većinom bira se ona klasa za koju se odlučilo 50% + 1 prediktivni model iz ansambla. *MV_S* arhitektura realizuje ovaj algoritma kombinovanja kada se svi težinski faktori postave na jedinicu, a prag ima vrednost $\lfloor N/2 \rfloor$.

Prilikom većinskog odlučivanja bira se ona klasa za koju je glasao najveći broj članova ansambla, bez obzira da li je taj broj veći ili manji od 50% svih glasova. Da bi *MV_S* arhitektura mogla realizovati ovaj algoritam kombinovanja sve težinske faktore treba postaviti na jedinicu, ali ovaj put je potreban varijabilan prag. Obzirom da upravljačka jedinica ima kontrolu nad upisom podataka u registar sa pragom ovo nije problem. Na početku se vrednost praga postavlja na nulu. Tokom N_{cl} taktova, upravljačka jedinica poredi vrednost tekućeg glasačkog registra sa tekućom vrednošću praga. Ukoliko je broj glasova veći od trenutne vrednosti praga, upravljačka jedinica čuva indeks tekuće klase i prag postavlja na vrednost koja je jednaka broju glasova koje je ta klasa dobila. Nakon N_{cl} taktova upravljačka jedinica šalje na izlaz „*Izlazna klasa*“ poslednju vrednost indeksa klase koja je sačuvana. Opisani postupak zapravo predstavlja hardversku implementaciju algoritma za određivanja pozicije elementa sa maksimalnom vrednošću u nizu brojeva.

U slučaju težinskog većinskog odlučivanja svakom prediktivnom modelu u ansamblu dodeljena je specifična težina. U ovom slučaju glasački registri (GR modul) ne čuvaju ukupan broj glasova već ukupan zbir težina prediktivnih modela koji su instancu klasifikovali u istu klasu. Kao klasifikacija instance od strane čitavog ansambla uzima se ona klasa čiji je zbir težina prediktivnih modela najveći. Da bi realizovala ovaj algoritam kombinovanja, *MV_S* arhitekturu treba konfigurisati na isti način kao i u slučaju većinskog odlučivanja, sa jednom razlikom. Vrednosti težinskih faktora više ne treba postaviti na jedinicu, već na onu vrednost koja odgovara težini koja je pridružena svakom individualnom prediktivnom modelu.

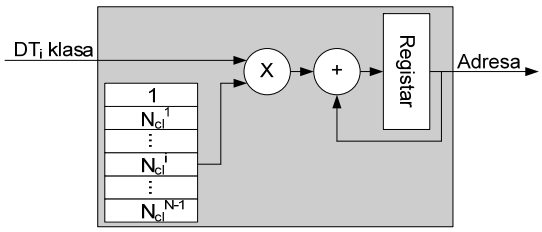
Implementacija BKS algoritma

BKS algoritam koristi tabelu koja zapravo predstavlja mapiranje klasifikacionih vektora u odgovarajuće izlazne klase. Svaki put kada se neki klasifikacioni vektor pojavi na izlazima ansambla, njemu pridružena klasa se bira kao klasa u koju ansambl klasifikuje tekuću instancu. Veličina ove tabele iznosi $(N_{cl})^N$ lokacija. Sama tabela se realizuje kao RAM ili ROM memorija, a glavni problem predstavlja hardverska implementacija postupka za izračunavanje adrese lokacije u *BKS* tabeli na osnovu trenutnog klasifikacionog vektora opisanog pomoću sledećeg izraza.

$$adresa = \sum_{i=1}^{N} \left(DTC_i - 1\right) \cdot \left(N_{cl}\right)^{i-1} \tag{3.2}$$

Simbol DTC_i u izrazu (3.2) predstavlja klasu, iz skupa mogućih klasa $\{1, 2, ..., N_{cl}\}$, u koju je *i*-ti prediktivni model klasifikovao tekuću instancu.

Modul za serijsko izračunavanje adrese lokacije u *BKS* tabeli prikazan je na slici 3.18.



Slika ***Error! No text of specified style in document.****..18 Arhitektura modula za serijsko izračunavanje adrese lokacije u* BKS *tabeli*

Predloženi modul zahteva jedna množač i jedan sabirač pomoću kojih se izračunava vrednost adrese lokacije u *BKS* tabeli korišćenjem izraza (3.2). Takođe je potrebno i *N* registara u kojima se čuvaju vrednosti faktora $(N_{cl})^i$, *i=0, ..., N-1*, potrebnih za izračunavanje vrednosti izraza (3.2). Tokom *N* taktova na ulaz „DT_i *klasa*“ dovodi se predikcija *i*-tog prediktivnog modela koja se zatim množi sa odgovarajućim faktorom i dodaje na trenutnu vrednost adrese koja se čuva u pomoćnom registru. Nakon *N* taktova na izlazu „*Adresa*“ pojavljuje se adresa lokacije u *BKS* tabeli u kojoj se nalazi klasa u koju će ansambl klasifikovati tekuću instancu.

Arhitektura za serijsko kombinovanje predikcija pomoću *BKS* algoritma sastojala bi se iz dva modula. Prvi modul bi serijski izračunavao adresu u *BKS* tabeli na osnovu trenutne vrednosti klasifikacionog vektora, i on je

prikazan na slici 3.18. Drugi modul bio bi RAM ili ROM memorija u kojoj se čuva *BKS* tabela. U daljem tekstu ova arhitektura označena je kao *BKS_S* arhitektura.

3.2.2 Paralelne arhitekture

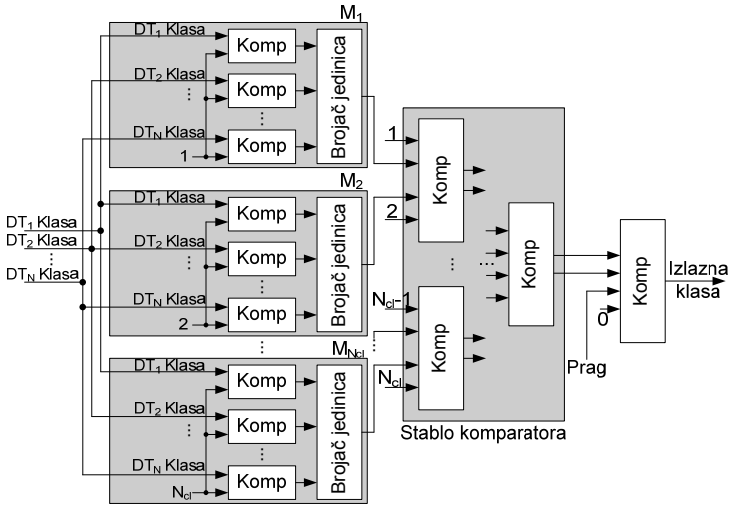
U slučaju korišćenja neke od arhitektura za paralelnu evaluaciju članova ansambla, radi očuvanja brzine klasifikacije, neophodno je koristiti arhitekture koje su u stanju da donesu kolektivnu odluku o klasifikaciji tekuće instance obrađujući istovremeno predikcije svih članova ansambla. Predstavljene serijske arhitekture za kombinovanje predikcija nisu u stanju da odgovore ovim zahtevima, pa je za svaki od razmatranih algoritama kombinovanja predikcija potrebno razviti nove, paralelne arhitekture.

Implementacija različitih varijanti većinskog odlučivanja

Na slici 3.18 prikazana je *MV_P* arhitektura pomoću koje je moguće realizovati sledeće algoritme za kombinovanje predikcija individualnih članova ansambla:

- jednoglasno odlučivanje,
- odlučivanje prostom većinom,
- većinsko odlučivanje,
- težinsko većinsko odlučivanje.

MV_P arhitektura bazira se na paralelnoj evaluaciji predikcija individualnih prediktivnih modela radi donošenja kolektivne odluke. Predikcije individualnim prediktivnih modela učitavaju istovremeno, u toku jednog takta, a nakon toga se donosi kolektivna odluka.



Slika ***Error! No text of specified style in document.****.19* MV_P *arhitektura za paralelnu implementaciju algoritama većinskog odlučivanja*

Na slici 3.19, signali DT_i *Klasa* predstavljaju predikcije individualnih prediktivnih modela koji čine ansambl. *MV_P* arhitektura koristi protočnu obradu prilikom donošenja kolektivne odluke o klasi kojoj tekuća instanca pripada. Imajući ovo u vidu, *MV_P* arhitektura ima odgovarajući broj faza u protočnoj obradi. Moduli $M_1, ..., M_{N_{cl}}$, čine prvu fazu protočne obrade i koriste se za određivanje broja glasova datih svakoj od mogućih N_{cl} klasa. Na primer, modul M_1 određuje koliko je prediktivnih modela iz ansambla klasifikovalo tekuću instancu u klasu 1, odnosno koliko glasova je dobila klasa 1. Treba voditi računa da je ovaj broj glasova potrebno izračunati u samo jednom taktu, istovremeno procesirajući izlaze svih članova ansambla. Ovo je moguće uraditi pomoću N komparatora, gde je sa N označen broj članova ansambla. Svaki od komparatora poredi predikciju jednog od članova sa željenom klasom. Ukoliko se one poklapaju izlaz komparatora se postavlja na jedinicu, u obrnutom slučaju izlaz komparatora je nula. Izlazi N komparatora čine jedan N-bitni binarni vektor. Broj jedinica u ovom vektoru jednak je broju glasova koji je dotična klasa dobila od strane ansambla. Sledeći modul, „*Brojač jedinica*“ upravo određuje koliko jedinica je prisutno u posmatranom binarnom vektoru. i ova operacija mora se izvesti u paraleli. Ovaj modul je zapravo jedna kombinaciona mreža sa N jednobitnih ulaza i $\lceil ld(N) \rceil$ izlaza.

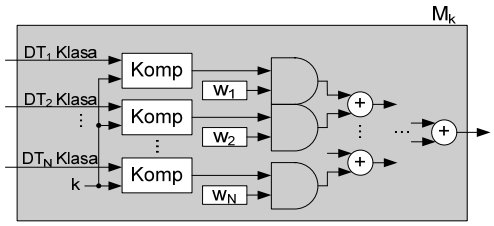
Na osnovu do sada rečenog, jasno je da se na izlazima svih modula iz prve faze protočne obrade nalazi podaci o broju glasova koje je svaka klasa dobila od strane ansambla prilikom klasifikovanja tekuće instance. U sledećim faza protočne obrade neophodno je odrediti klasu sa maksimalnim brojem glasova. I ova operacija se mora izvesti u paraleli, što je moguće ostvariti ukoliko se koristi stablo posebnih komparatora. Svaki nivo u stablu komparatora

zapravo čini jednu fazu u protočnoj obradi. Kao što se sa slike 3.19 može primetiti svaki od komparatora ima četiri ulaza i dva izlaza. Dva od četiri ulaza predstavljaju broj glasova koji su odgovarajuće klase dobile, a druga dva ulaza predstavljaju kodove pridružene tim klasama (zapravo su to binarni brojevi od 1 do N_{cl} što se može i videti na slici 3.19). Komparator poredi broj glasova koje su dobile klase i na svoja dva izlaza prosleđuje relevantne podatke o klasi koja je dobila veći broj glasova (kod klase i broj glasova koje je ta klasa dobila). Ove informacije se koriste u komparatorima u sledećeg nivou stabla komparatora, sve dok se ne dođe do poslednjeg nivoa. Nakon prolaska kroz sve nivoe u stablu komparatora, na izlazu komparatora iz poslednjeg nivoa nalaze se podaci o klasi koja je dobila najveći broj glasova. Ovi podaci porede se pomoću još jednog komparatora sa unapred definisanim pragom kome je kao kod klase pridružena vrednost nula. U slučaju da je klasa sa maksimalnim brojem glasova dobila broj glasova koji je veći od praga, na izlazu komparatora pojaviće se kod te klase, što će zapravo predstavljati klasu u koju je ansambl klasifikovao tekuću instancu. U slučaju da je maksimalni broj glasova manji od praga, na izlazu komparatora se pojavljuje nula, koja označava situaciju da ansambl ne može da dođe do odluke u koju klasu treba klasifikovati tekuću instancu.

Kao i u slučaju *MV_S* arhitekture pravilnim odabirom vrednosti praga *MV_P* arhitektura može realizovati različite varijante algoritma većinskog odlučivanja.

Ukoliko prag postavimo na vrednost *N-1*, *MV_P* arhitektura će realizovati algoritam jednoglasnog odlučivanja. Odlučivanje prostom većinom ostvaruje se ukoliko vrednost praga postavimo na $\lfloor N/2 \rfloor$, a većinsko odlučivanje realizuje se ukoliko vrednost praga postavimo na 0.

Da bi *MV_P* arhitektura mogla da realizuje težinsko većinsko odlučivanje, potrebno je modifikovati module $M_1, ..., M_{N_{cl}}$. Kod težinskog većinskog odlučivanja svakom članu ansambla pridružena je težina, tako da u ovom slučaju nije dovoljno samo prebrojati koliko članova je glasalo za koju klasu, već se umesto toga moraju sabrati težine pridružene članovima. Modifikovani moduli $M_1, ..., M_{N_{cl}}$ koji realizuju ovu operaciju prikazani su na slici 3.20.



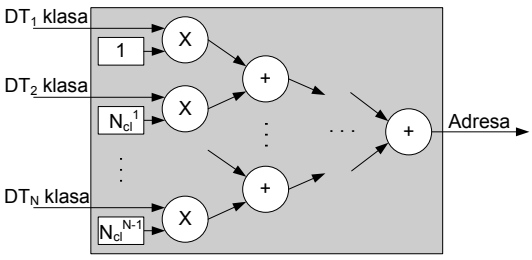
*Slika **Error! No text of specified style in document.**.20 Modifikovani modul za određivanje broja glasova datih klasi* k *u slučaju težinskog većinskog odlučivanja*

N komparatora, pomoću niza I kola, kontrolišu koje će težine biti sabrane da bi se formirao podatak u ukupnoj težini koja je asocirana klasi *k*. Odabrane težine se sumiraju pomoću stabla sabirača sa protočnom obradom podataka.

Ovo je jedina modifikacija koja je neophodna da bi *MV_P* arhitektura realizovala težinsko većinsko odlučivanje. Kao i u slučaju većinskog odlučivanja, vrednost praga treba postaviti na nulu.

Implementacija BKS algoritma

Kao što je ranije rečeno, prilikom implementacije BKS algoritma za kombinovanje predikcija individualnih članova ansambla, kritična operacija za hardversku implementaciju jeste izračunavanje adrese lokacije u BKS tabeli koju treba posetiti. Ova adresa računa se pomoću izraza (3.2). Međutim, u ovom slučaju neophodno je osmisliti arhitekturu koja će biti u stanju da izraz (3.2) izračuna u paraleli. Modul za paralelno izračunavanje adrese lokacije u BKS tabeli prikazan je na slici 3.21.



*Slika **Error! No text of specified style in document.**.21 Arhitektura modula za paralelno izračunavanje adrese lokacije u* BKS *tabeli*

Obzirom da je sve članove iz sume u izrazu (3.2) potrebno sumirati istovremeno, neophodno je koristiti stablo sabirača. Članovi koje je potrebno sumirati su zapravo proizvodi koji se formiraju pomoću N množača. Svaki od nivoa u stablu sabirača kao i nivo sa množačima čini po jednu fazu u protočnoj obradi podataka. Na ovaj način je moguće postići maksimalnu propusnu moć sistema.

3.3 Potrebni hardverski resursi i performanse arhitektura za implementaciju ansambala stabala odluka

U prethodnim odeljcima predstavljene su arhitekture za hardversku implementaciju modula za određivanje predikcija članova ansambla kao i modula za kombinovanje predikcija članova ansambla. Kombinovanjem različitih arhitektura za realizaciju ova dva modula mogu se formirati arhitekture za hardversku implementaciju ansambala sa različitim karakteristikama. Kao što je ranije već rečeno, sve arhitekture se mogu podeliti u dve grupe: arhitekture sa paralelnim procesiranjem i arhitekture sa serijskim procesiranjem. Na slici 3.1 prikazana je generalna struktura ove dve grupe arhitektura. Međutim, nakon detaljnog prikaza arhitektura za realizaciju pomenuta dva modula, u mogućnosti smo da prikažemo detaljniju strukturu obe grupe arhitektura za hardversku implementaciju ansambala stabala odluka.

U nastavku ćemo razmatrati sledeće arhitekture za implementaciju dva modula potrebna prilikom hardverske realizacije ansambala stabala odluka:

Arhitekture za određivanje predikcije članova ansambla sa serijskim procesiranjem:

1. *SMPL1-S* – arhitektura bazirana na modifikovanoj *SMPL* arhitekturi kod koje se određivanje položaja instance u odnosu na hiperravan vrši serijskom evaluacijom izraza (A.2)
2. *SMPL2-S* – arhitektura bazirana na modifikovanoj *SMPL* arhitekturi kod koje se određivanje položaja instance u odnosu na hiperravan vrši paralelnom evaluacijom izraza (A.2)
3. *UN1-S* – arhitektura bazirana na modifikovanoj *UN* arhitekturi kod koje se određivanje položaja instance u odnosu na hiperravan vrši serijskom evaluacijom izraza (A.2)
4. *UN2-S* – arhitektura bazirana na modifikovanoj *UN* arhitekturi kod koje se određivanje položaja instance u odnosu na hiperravan vrši paralelnom evaluacijom izraza (A.2)

Arhitekture za određivanje predikcije članova ansambla sa paralelnim procesiranjem:

5. *SMPL1-P* – arhitektura bazirana na originalnoj *SMPL* arhitekturi kod koje se određivanje položaja instance u odnosu na hiperravan vrši serijskom evaluacijom izraza (A.2)
6. *SMPL2-P* – arhitektura bazirana na originalnoj *SMPL* arhitekturi kod koje se određivanje položaja instance u odnosu na hiperravan vrši paralelnom evaluacijom izraza (A.2)
7. *UN1-P* – arhitektura bazirana na originalnoj *UN* arhitekturi kod koje se određivanje položaja instance u odnosu na hiperravan vrši serijskom evaluacijom izraza (A.2)
8. *UN2-P* – arhitektura bazirana na originalnoj *UN* arhitekturi kod koje se određivanje položaja instance u odnosu na hiperravan vrši paralelnom evaluacijom izraza (A.2)

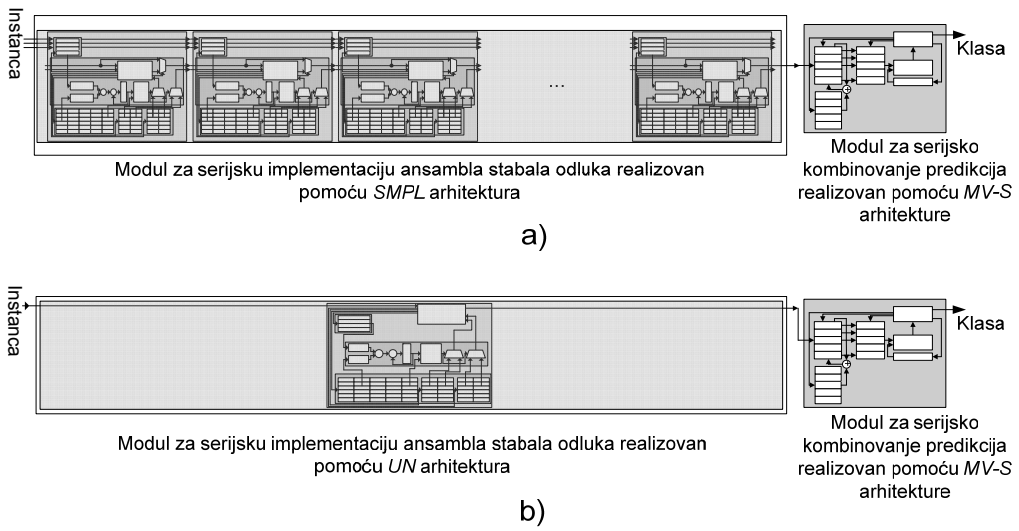
Arhitekture za kombinovanje predikcija članova ansambla sa serijskim procesiranjem:

9. *MV-S* – serijska arhitektura za implementaciju različitih varijanti algoritma većinskog odlučivanja
10. *BKS-S* – serijska arhitektura za implementaciju *BKS* algoritma kombinovanja

Arhitekture za kombinovanje predikcija članova ansambla sa paralelnim procesiranjem:

11. *MV-P* – paralelna arhitektura za implementaciju različitih varijanti algoritma većinskog odlučivanja
12. *MVT-P* – paralelna arhitektura za implementaciju različitih algoritma težinskog većinskog odlučivanja
13. *BKS-P* – serijska arhitektura za implementaciju *BKS* algoritma kombinovanja

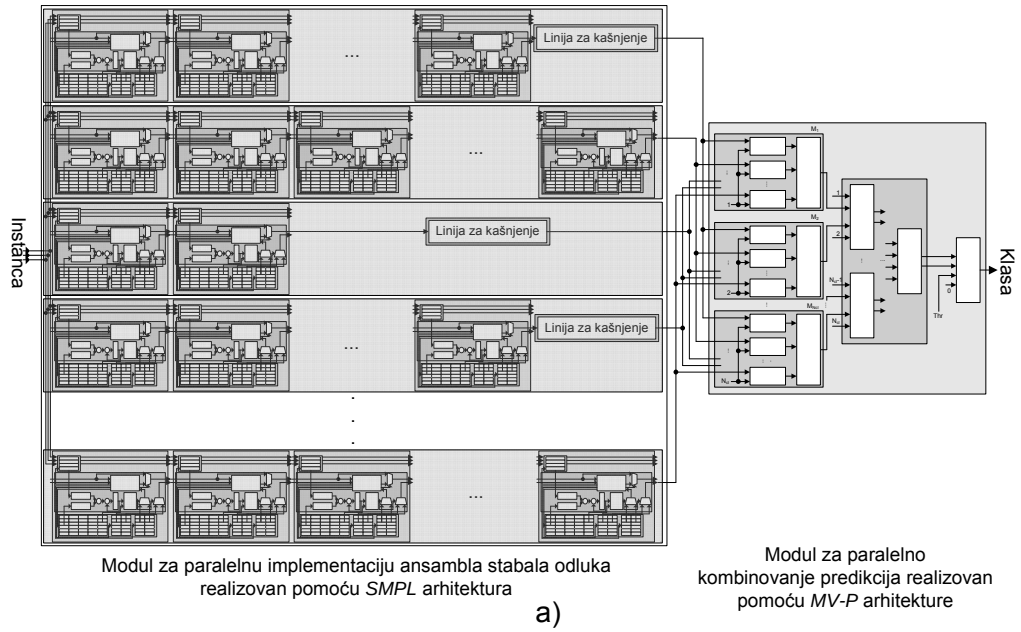
Kao ilustracija načina na koje se mogu kombinovati arhitekture za realizaciju pomenuta dva modula, na slici 3.22 prikazan je izgled arhitektura za hardversku implementaciju ansambla stabala odluka kod kojih je modul za određivanje predikcija članova ansambla realizovan pomoću *SMPL1-S* ili *SMPL2-S* arhitektura (slika 3.22a) ili pomoću *UN1-S* ili *UN2-S* arhitektura (slika 3.22b). U oba slučaja modul za kombinovanje predikcija članova ansambla realizovan je pomoću *MV-S* arhitekture.

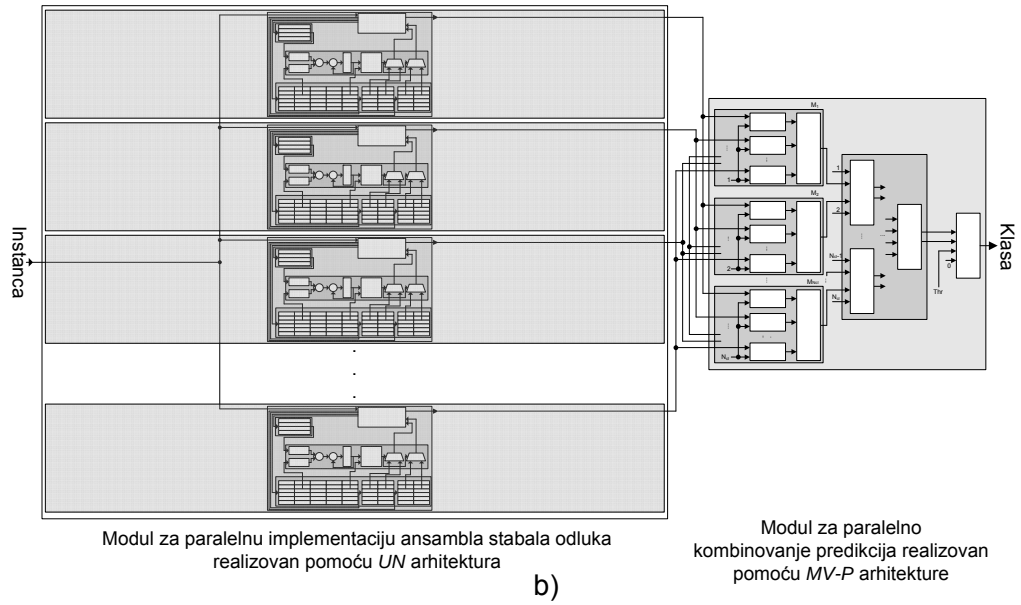


Slika ***Error! No text of specified style in document.****..22 Arhitekture za hardversku implementaciju ansambala stabala odluka sa serijskim procesiranjem a) bazirane na* SMPL *arhitekturama, b)bazirane sa* UN *arhitekturama*

Arhitekture sa serijskim procesiranjem odlikuju se relativno skromnih zahtevima u pogledu hardverskih resursa, jer one u jednom trenutku procesiraju samo jedno stablo iz ansambla. Ova činjenica rezultuje u relativno maloj brzini klasifikacije instanci ukoliko se ansambl realizuje pomoću ovih arhitektura.

U slučaju arhitektura za određivanje predikcija članova ansambla baziranih na originalnim *SMPL* arhitekturama predstavljenim u glavi 2, potreban broj univerzalnih čvorova bira se tako da omogući implementaciju najdubljeg stabla odluke u ansamblu. Na slici 3.23 prikazan je izgled arhitektura za hardversku implementaciju ansambala stabala odluka kod kojih je modul za određivanje predikcija članova ansambla realizovan pomoću *SMPL1-P* ili *SMPL2-P* arhitektura (slika 3.23a) ili pomoću *UN1-P* ili *UN2-P* arhitektura (slika 3.23b). I ovde je u oba slučaja modul za kombinovanje predikcija članova ansambla realizovan pomoću *MV-P* arhitekture.





*Slika **Error! No text of specified style in document.**.23 Arhitekture za hardversku implementaciju ansambala stabala odluka sa paralelnim procesiranjem a) bazirane na* SMPL *arhitekturama, b) bazirane sa* UN *arhitekturama*

Kao što je ranije rečeno, u slučaju arhitektura sa paralelnim procesiranjem, svako stablo odluke iz ansambla evaluira se u paraleli. Kod modula za određivanje predikcija članova ansambla baziranih na modifikovanim *SMPL* arhitekturama neophodno je uvesti linije za kašnjenje kod onih stabala odluka iz ansambla čija je dubina manja od dubine najdubljeg stabla iz ansambla. Ovo je neophodno da bi se očuvao sinhronizam prilikom protočne obrade instanci u sistemu. Slična stvar mora se izvesti i u slučaju modula baziranih na modifikovanim *UN* arhitekturama, samo što je u ovom slučaju potrebno koristiti linije sa varijabilnim kašnjenjem.

Kao što se sa slike 3.23 može zaključiti, arhitekture sa paralelnim procesiranjem zahtevaju znatno više hardverskih resursa u odnosu na arhitekture sa serijskim procesiranjem. Ovo je naravno posledica paralelnog određivanja predikcija članova ansambla. Kao rezultat ovog paralelizma, brzina klasifikacije arhitektura sa paralelnim procesiranjem imaju znatno veću brzinu klasifikacije od arhitektura sa serijskim procesiranjem.

3.3.1 Teorijska analiza potrebnih resursa i performansi

Za svaku od prethodno navedenih arhitektura za implementaciju modula za određivanje predikcija članova ansambla i modula za kombinovanje predikcija u tabelama 3.1 i 3.2 respektivno, prikazani su neophodni hardverski resursi kao i performanse u terminima:

- broja stabala odluka koja čine ansambl, N
- broja čvorova za svako od N stabala odluka, N_{dt_i}
- dubini svakog stabla odluke, M_i
- broja posećenih čvorova u svakom stablu prilikom klasifikacije tekuće instance, N_{nv_i}
- broja atributa posmatranog klasifikacionog problema, n
- broja klasa posmatranog klasifikacionog problema, N_{cl}
- broja bitova koji se koriste za reprezentaciju vrednosti atributa klasifikacionog problema, N_a
- broja bitova koji se koriste za reprezentaciju vrednosti koeficijenata hiperravni u svakom od čvorova stabla odluke, N_c
- broja bitova koji se koriste za reprezentaciju vrednosti težinskih faktora, N_w
- trajanja periode globalnog sinhronizacionog signala (takta), CP

Kao i ranije, neophodni hardverski resursi za implementaciju arhitektura izraženi su preko veličine memorije potrebne za smeštanje relevantnih podataka, i broju sabirača i množača. U slučaju arhitektura za hardversku implementaciju algoritama za kombinovanje predikcija pored ovih podataka hardverski resursi izraženi su i brojem neophodnih komparatora.

U tabeli 3.1 u koloni koja sadrži podatke o potrebnim memorijskim resursima za svaku od arhitektura navedena su po četiri podatka. Prvi se odnosi na ukupnu veličinu potrebnih memorija za smeštanje instanci, koje se nalaze unutar modula M1. Druga tri odnose se na ukupnu veličinu memorija potrebnih za smeštanje koeficijenata, adresa čvorova naslednika i klasa asociranih čvorovima naslednicima, modul M3. Treća i četvrta kolona prikazuju broj neophodnih sabirača i množača koji se nalaze unutar modula M2. Poslednja, peta kolona, predstavlja propusnu moć sistema izraženu u broj instanci koje se mogu klasifikovati u jedinici vremena.

U tabeli 3.2, u koloni koja sadrži podatke o potrebnim memorijskim resursima za svaku od arhitektura navedeni su relevantni podaci. Ovaj put broj podataka nije isti za sve arhitekture, jer se ovoga puta arhitekture međusobno značajno razlikuju pa nije moguće na jednoobrazan način predstaviti podatke. Sve ostale kolone imaju isto značenje kao i u tabeli 3.2.

Tabela **Error! No text of specified style in document.**.1 Potrebni hardverski resursi i performanse arhitektura za realizaciju modula za određivanje predikcija članova ansambla

Arhitektura	Memorija	Sabirači	Množači	Propusna moć
SMPL1-S	$\max_{i=1}^{N}\{M_i\}\cdot n \; x \; N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \max_{i=1}^{N}\left\{\left\lceil ld\left(N_{dt_i}\right)\right\rceil\right\}$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	$\max_{i=1}^{N}\{M_i\}$	$\max_{i=1}^{N}\{M_i\}$	$\frac{1}{\left(\max_{i=1}^{N}\{M_i\}+N-1\right)\cdot(n+1)\cdot CP}$
SMPL2-S	$\max_{i=1}^{N}\{M_i\}\cdot n \; x \; N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \max_{i=1}^{N}\left\{\left\lceil ld\left(N_{dt_i}\right)\right\rceil\right\}$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	$\max_{i=1}^{N}\{M_i\}\cdot(n-1)$	$\max_{i=1}^{N}\{M_i\}\cdot n$	$\frac{1}{\left(\max_{i=1}^{N}\{M_i\}+N-1\right)\cdot CP}$
UN1-S	$n \; x \; N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \max_{i=1}^{N}\left\{\left\lceil ld\left(N_{dt_i}\right)\right\rceil\right\}$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	1	1	$\frac{1}{\sum_{i=1}^{N} N_{nv_i}\cdot(n+1)\cdot CP}$

UN2-S	$n \; x \; N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \max_{i=1}^{N}\left\{\left\lceil ld\left(N_{dt_i}\right)\right\rceil\right\}$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	$n-1$	n	$\frac{1}{\sum_{i=1}^{N} N_{nv_i}\cdot CP}$
SMPL1-P	$\left(\sum_{i=1}^{N} M_i \cdot n\right) x \, N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\sum_{i=1}^{N}\left(2\cdot N_{dt_i} x \left\lceil ld\left(N_{dt_i}\right)\right\rceil\right)$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	$\sum_{i=1}^{N} M_i$	$\sum_{i=1}^{N} M_i$	$\frac{1}{(n+1)\cdot CP}$
SMPL2-P	$\left(\sum_{i=1}^{N} M_i \cdot n\right) x \, N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\sum_{i=1}^{N}\left(2\cdot N_{dt_i} x \left\lceil ld\left(N_{dt_i}\right)\right\rceil\right)$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	$\sum_{i=1}^{N} M_i \cdot (n-1)$	$\sum_{i=1}^{N} M_i \cdot n$	$\frac{1}{CP}$
UN1-P	$N{\cdot}n \; x \; N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\sum_{i=1}^{N}\left(2\cdot N_{dt_i} x \left\lceil ld\left(N_{dt_i}\right)\right\rceil\right)$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	N	N	$\frac{1}{\max_{i=1}^{N}\left\{N_{nv_i}\right\}\cdot(n+1)\cdot CP}$
UN2-P	$N{\cdot}n \; x \; N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\sum_{i=1}^{N}\left(2\cdot N_{dt_i} x \left\lceil ld\left(N_{dt_i}\right)\right\rceil\right)$, $\left(2\cdot\sum_{i=1}^{N} N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	$N{\cdot}(n{-}1)$	$N{\cdot}n$	$\frac{1}{\max_{i=1}^{N}\left\{N_{nv_i}\right\}\cdot CP}$

Većina memorijskih resursa ima linearan porast po svakom od relevantnih parametara. Što se tiče stope rasta neophodnog broja sabirača i množača, analizom podataka može se zaključiti da je ona u najgorem slučaju linearna po svakom od relevantnih parametara. Može se zaključiti da se sve predložene arhitekture izuzetno dobro skaliraju sa porastom složenosti problema koji se rešava, što predstavlja izuzetno dobru osobinu pogotovo u slučaju hardverske implementacije.

Analizom memorijskih zahteva razmatranih arhitektura možemo videti da sve arhitekture imaju gotovo identične memorijske zahteve. Paralelne arhitekture zahtevaju nešto veću memoriju za čuvanje instanci koje se trenutno klasifikuju, ali u većini praktičnih slučajeva ovaj član nije dominantan u ukupnom iznosu za potrebnu memoriju. Međutim, ukoliko se analizira broj neophodnih sabirača i množača možemo videti da se on drastično razlikuje od arhitekture do arhitekture. Ubedljivo najbolja je *UN1-S* arhitektura koja zahteva samo jedan sabirač i jedan množač. Ovo je praktično minimalni broj sabirača i množača koji je potreban da bi se mogao implementirati bilo koji ansambl stabala odluka. Kao što se moglo i očekivati, arhitekture kod kojih se članovi ansambla evaluiraju serijski zahtevaju značajno manji broj sabirača i množača u poređenju sa arhitekturama kod kojih se članovi ansambla evaluiraju u paraleli. Arhitekture sa paralelnom evaluacijom zahtevaju okolo *N* puta veći broj sabirača i množača u odnosu na arhitekture sa serijskom evaluacijom. U slučaju implementacije ansambala sa velikim brojem članova, broj zahtevanih sabirača i množača može postati neprihvatljivo veliki.

Potpuno suprotna situacija prisutna je u slučaju kada poredimo propusne moći različitih arhitektura. Arhitekture sa paralelnom evaluacijom članova ansambla imaju značajno veću propusnu moć od arhitektura sa serijskom evaluacijom. Ova propusna moć je oko *N* puta veća kada se porede odgovarajuće arhitekture sa paralelnom i serijskom evaluacijom. Naravno, veća propusna moć „plaćena“ je većim zahtevima za hardverskim resursima.

Glavni zaključak koji se može izvući na osnovu tabele 3.1 jeste da predložene arhitekture nude različite odnose performanse/resursi te se za svaku konkretnu aplikaciju može odabrati arhitektura koja najbliže odgovara konkretnim zahtevima.

Tabela **Error! No text of specified style in document.**.2 Potrebni hardverski resursi i performanse arhitektura za realizaciju modula za kombinovanje predikcija članova ansambla

Arhitektura	Memorija	Komparatori	Sabirači	Množači	Propusna moć
MV-S	$2 \cdot N_{cl} \cdot \lceil ld(N_{cl}) \rceil$, $N \cdot N_w$	1	0	1	$\frac{1}{N \cdot CP}$
BKS-S	$N \cdot \left\lceil ld\left((N_{cl})^{N-1}\right) \right\rceil$, $(N_{cl})^N \, x \lceil ld(N_{cl}) \rceil$	*0*	*1*	*1*	$\frac{1}{N \cdot CP}$
MV-P	0	$(N_{cl}+1) \cdot N$	0	0	$\frac{1}{CP}$
MVT-P	$N \cdot N_w$	$(N_{cl}+1) \cdot N$	$N_{cl} \cdot (N-1)$	*0*	$\frac{1}{CP}$
BKS-P	$N \cdot \left\lceil ld\left((N_{cl})^{N-1}\right) \right\rceil$, $(N_{cl})^N \, x \lceil ld(N_{cl}) \rceil$	*0*	$N-1$	*N*	$\frac{1}{CP}$

Slični zaključci se mogu izvesti i u slučaju arhitektura za realizaciju modula za kombinovanje predikcija članova ansambla. Paralelne arhitekture imaju *N* puta veću propusnu moć od odgovarajućih serijskih arhitektura. Što se tiče zahteva za hardverskim resursima, serijske arhitekture imaju izrazito skromne zahteve u poređenju sa paralelnim arhitekturama. *BKS* arhitekture mogu imati velike memorijske zahteve u slučaju implementacije ansambala sa velikim brojem članova. Ovo je generalno problem *BKS* algoritma, jer veličina tabele u slučaju velikih ansambala može postati neprihvatljiva. Sve arhitekture za implementaciju različitih varijanti algoritma većinskog odlučivanja imaju vrlo skromne hardverske zahteve što ih čini vrlo atraktivnim za korišćenje prilikom hardverske implementacije ansambala stabala odluka. Upravo iz ovog razloga, prilikom procene potrebnih hardverskih resursa i performansi sistema za realizaciju ansambala stabala odluka za konkretne UCI test probleme, razmatrani su sistemi kod kojih je modul za kombinovanje predikcija članova ansambla realizovan korišćenjem neke od arhitektura za implementaciju algoritma većinskog odlučivanja (*MV-S, MV-P* ili *MVT-P* arhitekture).

4. Primena

Nakon teorijske analize potrebnih hardverskih resursa i brzine klasifikacije različitih arhitektura za hardversku implementaciju ansambala stabala odluka, interesantno je odrediti njihove performanse i na realnim problemima iz prakse. Za ove poterbe korišćen je skup od ukupno 29 standardnih test problema odabranih iz standardne baze podatake za testiranje algoritama mašinskog učenja, „*UCI Machine Learning Repository*“, [47]. Glavne karakteristike odabranih problema prikazane su u tabeli 4.1.

Tabela 4.1 Karakteristike realnih klasifikacionih problema preuzetih iz UCI baze podataka

Problem	Oznaka	Broj atributa	Broj klasa	Broj instanci
Australian Credit Approval	AUSC	14	2	690
Balance Scale	BC	4	3	625
Breast Cancer Wisconsin	BCW	9	2	699
Breast Cancer	BSC	9	2	264
Car Evaluation	CAR	6	4	1728
Contraceptive Method Choice	CMC	9	3	1333
German Credit	GER	24	2	1000
Glass Identification	GLS	9	6	214
Hepatitis	HEP	19	2	155
Cleveland Heart Disease	HRTC	13	5	297
Statlog Heart Disease	HRTS	13	2	270
Ionosphere	ION	34	2	351
Iris	IRS	4	3	150
Liver Disorders	LIV	6	2	345
Lymphography	LYM	18	4	148
Page Blocks	PAGE	10	5	5427
Pima Indians Diabetes	PID	8	2	768
Sonar	SON	60	2	208
Thyroid Disease	THY	5	3	215
Tic-Tac-Toe Endgame	TTT	9	2	958
Statlog Vehicle Silhouettes	VEH	18	4	846
Congressional Voting Records	VOTE	16	2	232
Vowel Recognition	VOW	13	11	990
Waveform21	W21	21	3	5000
Waveform40	W40	40	3	5000
Wisconsin Diagnostic Breast Cancer	WDBC	30	2	569
Wine Recognition	WINE	13	3	178
Wisconsin Prognostic Breast Cancer	WPBC	33	2	194
Zoo	ZOO	17	7	101

Za svaku od arhitektura razvijen je RTL model pomoću VHDL jezika za opis hardvera koji je zatim sintetizovan pomoću *Xilinx ISE Foundation* 13.1 programskog paketa kompanije *Xilinx*. Razvijeni *RTL* modeli projektovani su sa mogućnošću parametrizacije, što omogućava njihovo jednostavno prilagođavanje karakteristikama problema koji se trenutno rešava. Na primer, *entity* deklaracija za *SMPL1-S* arhitekturu ima sledeći izgled.

```
entity SMPL1s_dte is
   generic (
      attribute_res_g          : integer := 8; -- number of bits used to represent attributes
      coef_res_g               : integer := 8; -- number of bits used to represent coefficients of the hyperplanes
      class_res_g              : integer := 4; -- number of bits used to represent class
      num_attributes_g         : integer := 60; -- number of attributes
      att_mem_bus_size_g   : integer := 18; -- size of the address bus for the attribute memory
      num_nodes_g              : integer := 2; -- maximum number of nodes in the DT
      coef_mem_bus_size_g: integer := 15; -- size of the address bus for the coefficient memory
      rln_mem_bus_size_g    : integer := 8; -- size of the address bus for the rigth-left node memory
      num_pipeline_stages_g: integer := 3 -- number of required pipeline stages
            );
```

```
  port (
    clk            : in  std_logic; -- clock input

    instance_i  : in std_logic_vector ((num_attributes_g*attribute_res_g) - 1 downto 0); -- new instance input
    class_i     : in std_logic_vector (class_res_g - 1 downto 0); -- input class
    node_adr_i : in std_logic_vector (att_mem_bus_size_g - 1 downto 0); -- root node address

    class_o     : out std_logic_vector(class_res_g - 1 downto 0) -- output class
      );
end entity SMPL1s_dte;
```

Izgled *entity* deklaracije u slučaju *SMPL1-S* arhitekture

Kao ciljna familija na kojoj su implementirane predložene arhitekture odabrana je *Xilinx Virtex5* familija.

4.1. Neophodni hardverski resursi

Tabele 3.3-3.10 sadrže podatke o neophodnim hardverskim resursima za implementaciju *SMPL1-S, SMPL2-S, UN1-S, UN2-S, SMPL1-P, SMPL2-P, UN2-P* i *UN* arhitektura za svaki od 29 odabranih UCI problema. Kao i ranije, hardverski resursi iskazani su veličinom potrebne memorije, pri čemu su odvojeno prikazane veličine memorija potrebnih za implementaciju M1 i M3 modula, i brojem potrebnih sabirača odnosno množača za realizaciju M2 modula. Svaka od ukupno osam tabela sadrži informacije o neophodnim hardverskim resursima za po jednu *SMPL* odnosno *UN* arhitekturu.

Tabela **Error! No text of specified style in document.**.3 Neophodni hardverski resursi u slučaju FPGA implementacije *SMPL1-S* arhitekture za 29 odabranih UCI test problema

Problem	M1 (kbits)	M3 (kbits)	Sabirači	Množači
ausc	0.68	4.34	5.00	5.00
bc	0.27	3.88	7.00	7.00
bsc	0.35	2.33	4.00	4.00
bcw	0.35	1.19	4.00	4.00
car	0.41	13.03	7.00	7.00
cmc	1.05	39.83	12.00	12.00
ger	1.41	13.06	6.00	6.00
gls	0.53	11.05	6.00	6.00
hep	0.37	0.13	2.00	2.00
hrtc	0.63	11.75	5.00	5.00
hrts	0.51	2.03	4.00	4.00
Ion	1.00	4.18	3.00	3.00
irs	0.12	1.31	3.00	3.00
liv	0.41	3.45	7.00	7.00
lym	0.53	3.49	3.00	3.00
page	0.78	30.47	8.00	8.00
pid	0.55	6.98	7.00	7.00
son	1.17	4.84	2.00	2.00
thy	0.15	1.57	3.00	3.00
ttt	0.53	5.14	6.00	6.00
veh	1.58	27.41	9.00	9.00
vote	0.31	1.22	2.00	2.00
vow	1.02	137.01	8.00	8.00
w21	2.05	59.68	10.00	10.00
w40	3.91	86.06	10.00	10.00
wdbc	0.88	2.43	3.00	3.00
wine	0.25	2.76	2.00	2.00
wpbc	0.97	4.41	3.00	3.00
zoo	0.83	15.63	5.00	5.00

Tabela **Error! No text of specified style in document.**.4 Neophodni hardverski resursi u slučaju FPGA implementacije *SMPL2-S* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	0.68	4.34	65.00	70.00
bc	0.27	3.88	21.00	28.00
bsc	0.35	2.33	32.00	36.00
bcw	0.35	1.19	32.00	36.00
car	0.41	13.03	35.00	42.00
cmc	1.05	39.83	96.00	108.00
ger	1.41	13.06	138.00	144.00
gls	0.53	11.05	48.00	54.00
hep	0.37	0.13	36.00	38.00
hrtc	0.63	11.75	60.00	65.00
hrts	0.51	2.03	48.00	52.00
Ion	1.00	4.18	99.00	102.00
irs	0.12	1.31	9.00	12.00
liv	0.41	3.45	35.00	42.00
lym	0.53	3.49	51.00	54.00
page	0.78	30.47	72.00	80.00
pid	0.55	6.98	49.00	56.00
son	1.17	4.84	118.00	120.00
thy	0.15	1.57	12.00	15.00
ttt	0.53	5.14	48.00	54.00
veh	1.58	27.41	153.00	162.00
vote	0.31	1.22	30.00	32.00
vow	1.02	137.01	96.00	104.00
w21	2.05	59.68	200.00	210.00
w40	3.91	86.06	390.00	400.00
wdbc	0.88	2.43	87.00	90.00
wine	0.25	2.76	24.00	26.00
wpbc	0.97	4.41	96.00	99.00
zoo	0.83	15.63	80.00	85.00

Tabela **Error! No text of specified style in document.**.5 Neophodni hardverski resursi u slučaju FPGA implementacije *UN1-S* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	0.14	4.34	1.00	1.00
bc	0.04	3.88	1.00	1.00
bsc	0.09	2.33	1.00	1.00
bcw	0.09	1.19	1.00	1.00
car	0.06	13.03	1.00	1.00
cmc	0.09	39.83	1.00	1.00
ger	0.23	13.06	1.00	1.00
gls	0.09	11.05	1.00	1.00
hep	0.19	0.13	1.00	1.00
hrtc	0.13	11.75	1.00	1.00
hrts	0.13	2.03	1.00	1.00
Ion	0.33	4.18	1.00	1.00
irs	0.04	1.31	1.00	1.00
liv	0.06	3.45	1.00	1.00
lym	0.18	3.49	1.00	1.00
page	0.10	30.47	1.00	1.00
pid	0.08	6.98	1.00	1.00
son	0.59	4.84	1.00	1.00
thy	0.05	1.57	1.00	1.00
ttt	0.09	5.14	1.00	1.00
veh	0.18	27.41	1.00	1.00
vote	0.16	1.22	1.00	1.00
vow	0.13	137.01	1.00	1.00
w21	0.21	59.68	1.00	1.00
w40	0.39	86.06	1.00	1.00

wdbc	0.29	2.43	1.00	1.00
wine	0.13	2.76	1.00	1.00
wpbc	0.32	4.41	1.00	1.00
zoo	0.17	15.63	1.00	1.00

Tabela **Error! No text of specified style in document.**.6 Neophodni hardverski resursi u slučaju FPGA implementacije *UN2-S* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	0.14	4.34	13.00	14.00
bc	0.04	3.88	3.00	4.00
bsc	0.09	2.33	8.00	9.00
bcw	0.09	1.19	8.00	9.00
car	0.06	13.03	5.00	6.00
cmc	0.09	39.83	8.00	9.00
ger	0.23	13.06	23.00	24.00
gls	0.09	11.05	8.00	9.00
hep	0.19	0.13	18.00	19.00
hrtc	0.13	11.75	12.00	13.00
hrts	0.13	2.03	12.00	13.00
Ion	0.33	4.18	33.00	34.00
irs	0.04	1.31	3.00	4.00
liv	0.06	3.45	5.00	6.00
lym	0.18	3.49	17.00	18.00
page	0.10	30.47	9.00	10.00
pid	0.08	6.98	7.00	8.00
son	0.59	4.84	59.00	60.00
thy	0.05	1.57	4.00	5.00
ttt	0.09	5.14	8.00	9.00
veh	0.18	27.41	17.00	18.00
vote	0.16	1.22	15.00	16.00
vow	0.13	137.01	12.00	13.00
w21	0.21	59.68	20.00	21.00
w40	0.39	86.06	39.00	40.00
wdbc	0.29	2.43	29.00	30.00
wine	0.13	2.76	12.00	13.00
wpbc	0.32	4.41	32.00	33.00
zoo	0.17	15.63	16.00	17.00

Tabela **Error! No text of specified style in document.**.7 Neophodni hardverski resursi u slučaju FPGA implementacije *SMPL1-P* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	5.27	4.22	92.00	92.00
bc	7.03	3.75	135.00	135.00
bsc	4.13	2.26	80.00	80.00
bcw	9.79	1.15	47.00	47.00
car	23.73	12.73	167.00	167.00
cmc	33.52	39.28	270.00	270.00
ger	10.11	12.97	143.00	143.00
gls	0.56	10.96	115.00	115.00
hep	13.58	0.13	3.00	3.00
hrtc	7.49	11.57	107.00	107.00
hrts	18.59	1.96	59.00	59.00
Ion	2.42	4.19	56.00	56.00
irs	7.91	1.32	62.00	62.00
liv	7.38	3.31	135.00	135.00
lym	18.65	3.48	42.00	42.00
page	13.13	30.17	191.00	191.00
pid	22.27	6.68	168.00	168.00
son	3.03	4.82	38.00	38.00
thy	9.93	1.58	62.00	62.00
ttt	32.34	4.98	113.00	113.00

veh	5.16	27.16	184.00	184.00
vote	26.66	1.23	33.00	33.00
vow	49.83	136.60	210.00	210.00
w21	86.72	59.24	243.00	243.00
w40	10.84	85.70	222.00	222.00
wdbc	7.62	2.40	37.00	37.00
wine	19.34	2.83	60.00	60.00
wpbc	16.10	4.37	60.00	60.00
zoo	5.27	15.57	97.00	97.00

Tabela **Error! No text of specified style in document.**.8 Neophodni hardverski resursi u slučaju FPGA implementacije *SMPL2-P* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	12.58	4.22	1196.00	1288.00
bc	5.27	3.75	405.00	540.00
bsc	7.03	2.26	640.00	720.00
bcw	4.13	1.15	376.00	423.00
car	9.79	12.73	835.00	1002.00
cmc	23.73	39.28	2160.00	2430.00
ger	33.52	12.97	3289.00	3432.00
gls	10.11	10.96	920.00	1035.00
hep	0.56	0.13	54.00	57.00
hrtc	13.58	11.57	1284.00	1391.00
hrts	7.49	1.96	708.00	767.00
Ion	18.59	4.19	1848.00	1904.00
irs	2.42	1.32	186.00	248.00
liv	7.91	3.31	675.00	810.00
lym	7.38	3.48	714.00	756.00
page	18.65	30.17	1719.00	1910.00
pid	13.13	6.68	1176.00	1344.00
son	22.27	4.82	2242.00	2280.00
thy	3.03	1.58	248.00	310.00
ttt	9.93	4.98	904.00	1017.00
veh	32.34	27.16	3128.00	3312.00
vote	5.16	1.23	495.00	528.00
vow	26.66	136.60	2520.00	2730.00
w21	49.83	59.24	4860.00	5103.00
w40	86.72	85.70	8658.00	8880.00
wdbc	10.84	2.40	1073.00	1110.00
wine	7.62	2.83	720.00	780.00
wpbc	19.34	4.37	1920.00	1980.00
zoo	16.10	15.57	1552.00	1649.00

Tabela **Error! No text of specified style in document.**.9 Neophodni hardverski resursi u slučaju FPGA implementacije *UN1-P* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	4.10	4.22	30.00	30.00
bc	1.17	3.75	30.00	30.00
bsc	2.64	2.26	30.00	30.00
bcw	2.64	1.15	30.00	30.00
car	1.76	12.73	30.00	30.00
cmc	2.64	39.28	30.00	30.00
ger	7.03	12.97	30.00	30.00
gls	2.64	10.96	30.00	30.00
hep	5.57	0.13	30.00	30.00
hrtc	3.81	11.57	30.00	30.00
hrts	3.81	1.96	30.00	30.00
Ion	9.96	4.19	30.00	30.00
irs	1.17	1.32	30.00	30.00
liv	1.76	3.31	30.00	30.00
lym	5.27	3.48	30.00	30.00

page	2.93	30.17	30.00	30.00
pid	2.34	6.68	30.00	30.00
son	17.58	4.82	30.00	30.00
thy	1.46	1.58	30.00	30.00
ttt	2.64	4.98	30.00	30.00
veh	5.27	27.16	30.00	30.00
vote	4.69	1.23	30.00	30.00
vow	3.81	136.60	30.00	30.00
w21	6.15	59.24	30.00	30.00
w40	11.72	85.70	30.00	30.00
wdbc	8.79	2.40	30.00	30.00
wine	3.81	2.83	30.00	30.00
wpbc	9.67	4.37	30.00	30.00
zoo	4.98	15.57	30.00	30.00

Tabela **Error! No text of specified style in document.**.10 Neophodni hardverski resursi u slučaju FPGA implementacije *UN2-P* arhitekture za 29 odabranih UCI test problema

Problem	***M1 (kbits)***	***M3 (kbits)***	***Sabirači***	***Množači***
ausc	4.10	4.22	390.00	420.00
bc	1.17	3.75	90.00	120.00
bsc	2.64	2.26	240.00	270.00
bcw	2.64	1.15	240.00	270.00
car	1.76	12.73	150.00	180.00
cmc	2.64	39.28	240.00	270.00
ger	7.03	12.97	690.00	720.00
gls	2.64	10.96	240.00	270.00
hep	5.57	0.13	540.00	570.00
hrtc	3.81	11.57	360.00	390.00
hrts	3.81	1.96	360.00	390.00
Ion	9.96	4.19	990.00	1020.00
irs	1.17	1.32	90.00	120.00
liv	1.76	3.31	150.00	180.00
lym	5.27	3.48	510.00	540.00
page	2.93	30.17	270.00	300.00
pid	2.34	6.68	210.00	240.00
son	17.58	4.82	1770.00	1800.00
thy	1.46	1.58	120.00	150.00
ttt	2.64	4.98	240.00	270.00
veh	5.27	27.16	510.00	540.00
vote	4.69	1.23	450.00	480.00
vow	3.81	136.60	360.00	390.00
w21	6.15	59.24	600.00	630.00
w40	11.72	85.70	1170.00	1200.00
wdbc	8.79	2.40	870.00	900.00
wine	3.81	2.83	360.00	390.00
wpbc	9.67	4.37	960.00	990.00
zoo	4.98	15.57	480.00	510.00

Rezultati eksperimenata izvedenih sa 29 odabranih UCI test problema prikazani u tabelama 3.3-3.10 u saglasnosti su sa opštim izrazima iz tabele 3.1. Kao i ranije, ovi rezultati mogu da posluže za bolju procenu neophodnih hardverskih resursa u slučaju rešavanja konkretnih, realnih problema. Na osnovu podataka iz tabela 3.3-3.10 može se zaključiti da zahtevani resursi arhitektura sa serijskom evaluacijom, ne prevazilaze resurse koje obezbeđuju savremene FPGA komponente. U slučaju arhitektura sa paralelnom evaluacijom to već nije slučaj. Na primer, najveća FPGA komponenta iz familije Virtex5, *XC5VSX240T*, sadrži ukupno 18576 kilobita memorije i 1056 posvećenih hardverskih sabirača i množača. Pomoću ove komponente se bez većih problema mogu implementirati četiri arhitekture sa serijskom evaluacijom (*SMPL1-S, SMPL2-S, UN1-S* i *UN2-S*) za svaki od 29 razmatranih UCI test problema. Međutim, u slučaju arhitektura sa paralelnom evaluacijom ansambla za 11 UCI test problema, zahtevani hardverski resursi prevazilaze resurse koje nudi i trenutno najveća FPGA komponenta. Naravno, ovo nije nepremostiva prepreka, jer bi se bez većih problema zahtevani ansambl mogao realizovati korišćenjem nekoliko FPGA komponenti.

Ukoliko analiziramo količinu neophodne veličine memorijskih resursa najzahtevnije su *SMPL1-P* i *SMPL2-P* arhitekture u slučaju *w40* test problema, koje zahtevaju po 172.42 kilobita. Ovaj količina je daleko manja od ukupno 18576 kilobita memorije koje nam stoje na raspolaganju u *XC5VSX240T* komponenti. Ono što se takođe može primetiti je da sve arhitekture imaju približno iste memorijske zahteve.

Naviše sabirača zahteva *SMPL2-P* arhitektura takođe u slučaju *w40* test problema, 8658. *XC5VSX240T* komponenta na raspolaganju ima ukupno 1056 hardverskih sabiračkih modula. Očigledno da ovaj broj ne bi bio dovoljan za implementaciju *SMPL2-P* arhitekture. Međutim ne treba gubiti iz vida činjenicu da nam pored hardverskih sabiračkih modula na raspolaganju stoji i programabilna logika unutar FPGA komponente organizovana u konfigurabilne logičke blokove, *CLB*. Ovi blokovi se mogu konfigurisati i povezati tako da realizuju proizvoljni digitalni sistem, pa i sabirače koji su nama neophodni. *XC5VSX240T* komponenta nudi ukupno 18720 CLB blokova. Potreban broj *CLB* blokova da bi se realizovao jedan 10-bitni sabirač iznosi ne više od dva. U ovom konkretnom slučaju potrebno je oko 15204 *CLB* blokova da bi se implementiralo zahtevanih 8658 sabirača, što iznosi oko 80% ukupnih programabilnih resursa koje ova komponenta nudi. Obzirom da je potrebno realizovati i približno isti broj množača, jasno je da čak i korišćenjem programabilnih resursa nije moguće realizovati čitav sistem samo pomoću jedne FPGA komponente. Međutim, kao što je ranije već rečeno, ukoliko se upotrebi veći broj FPGA komponenti i ovaj sistem bi se mogao realizovati.

Najveći broj množača potreban je za implementaciju *SMPL2-P* arhitekture u slučaju *w40* test problema, u proseku 8880. Situacija je slična kao i u slučaju sabirača, obzirom da *XC5VSX240T* komponenta na raspolaganju ima ukupno 1056 hardverskih množača.

Već prilikom teorijske analize nagovešteno je da arhitekture sa paralelnom evaluacijom ansambla, *SMPL2* i *UN2*, zahtevaju daleko više sabirača i množača od arhitektura sa serijskom evaluacijom. Takođe je pokazano da što se tiče potrebnih memorijskih resursa, svih osam arhitektura imaju podjednake zahteve. Analizom eksperimentalnih rezultata prikazanih u tabelama 3.3-3.10, zaključci teorijske analize mogu se potvrditi. Ono što rezultati eksperimenata još bolje pokazuju jeste konkretan količina resursa koji su neophodni da bi se bilo koja od predloženih arhitektura realizovala u slučaju realnih problema. Međutim, ono što se u ovom slučaju može izvesti kao zaključak na osnovu svih rezultata, jeste da se samo arhitekture za serijsku evaluaciju članova ansambla mogu koristiti bez većih problema, čak i u slučaju FPGA komponenti sa skromnijim mogućnostima. U slučaju arhitektura sa paralelnom evaluacijom to nije uvek slučaj.

4.2. Performanse

Pored analize hardverskih resursa koji su neophodni za realizaciju predloženih arhitektura za odabrane UCI test probleme, od interesa je takođe proceniti vremena potrebna za klasifikaciju instanci za svaku od osam predloženih arhitektura. Takođe je interesantno uporediti dobijene rezultate sa rezultatima postignutim u slučaju softverske implementacije ansambala stabala odluka u dva slučaja: kada se softverska implementacija izvršava na savremenom desktop računaru i kada se izvršava na savremenom *embedded* mikroprocesoru. Kao i u slučaju softverske implementacije stabala odluka i sistemi za evaluaciju ansambala stabala odluka implementirani su korišćenjem C programskog jezika. Implementacija u C jeziku je zatim kompajlirana korišćenjem *Microsoft Visual C++* kompajlera i izvršena pod *Microsoft Windows XP* operativnim sistemom na PC računaru sa *Intel Pentium* D 820 procesorom koji je radio na 2.8 GHz i ukupno 3 GB operativne memorije.

Predložene arhitekture implementirane su i pomoću FPGA komponente familije *Virtex 5*. Tabela 3.11 sadrži podatke o minimalnoj periodi globalnog sinhronizacionog signala, različitih sistema za hardversku implementaciju ansambala stabla odluka za 29 UCI test problema. Pomoću ovih vrednosti može se dobiti bolja predstava o učestanosti na kojima rade predložene arhitekture u konkretnim slučajevima.

Sa druge strane, rezultati prikazani u tabeli 3.12 predstavljaju prosečna ubrzanja u vremenu klasifikacije instance, za svaku od osam predloženih arhitekture za hardversku realizaciju stabla odluke u odnosu na softversku implementaciju koja se izvršavala na PC računaru. Za svaki od 29 UCI test problema, izvršeno je 50 eksperimenata. U svakom eksperimentu na slučajan način formiran je trening skup koji je činilo 70% raspoloživih instanci. Preostalih 30% instanci činilo je skup korišćen za kresanje stabla odluke. Korišćenjem ovih skupova formiran je ansambl sačinjen od 30 stabala odluka, a zatim je izmereno prosečno vreme klasifikacije instance za svaku od razmatranih implementacija. Vrednosti prikazane u tabeli 3.12 izračunate su kao količnik prosečnog vremena klasifikacije instance pomoću odgovarajuće softverske implementacije i vremena klasifikacije instance pomoću hardverske implementacije.

Tabela **Error! No text of specified style in document.**.11 Minimalne periode takta sistema za hardversku realizaciju ansambala stabala odluka baziranih na *SMPL* i *UN* arhitekturama za odabrane UCI probleme

Test skup	Minimalna perioda globalnog sinhronizacionog signala [ns]							
	SMPL1-S	*SMPL2-S*	*UN1-S*	*UN2-S*	*SMPL1-P*	*SMPL2-P*	*UN1-P*	*UN2-P*
ausc	6.888	10.000	6.757	12.412	6.531	9.993	5.063	12.740
bc	6.728	9.119	6.420	9.398	4.972	6.552	4.853	9.524
bcw	6.446	9.836	4.851	12.491	4.998	5.970	3.981	12.040
bsc	6.643	9.975	5.834	12.487	4.989	6.495	4.006	14.016
car	7.006	8.760	6.572	11.377	6.365	7.360	4.812	10.120
cmc	9.541	9.990	6.520	11.588	7.066	9.998	5.548	14.277
ger	7.274	9.846	6.527	12.487	6.774	9.913	4.389	14.005
gls	6.700	9.993	6.457	12.487	6.461	7.991	4.115	14.016
hep	6.334	9.932	5.770	14.256	4.870	5.863	3.926	14.256
hrtc	7.663	9.985	6.555	12.450	6.833	9.997	4.248	13.773
hrts	6.918	9.383	6.320	12.996	4.989	7.113	4.323	13.773
ion	6.530	9.999	6.618	12.497	6.459	9.975	4.379	14.116
irs	6.893	8.395	6.354	9.621	4.945	6.561	3.941	7.019
liv	7.006	8.760	6.616	11.377	6.365	7.360	4.812	10.120
lym	6.853	9.993	6.664	12.489	4.995	9.641	4.278	14.269
page	7.671	9.489	6.655	11.021	6.928	9.979	5.372	11.464
pid	7.816	9.982	6.654	10.658	6.719	9.995	4.930	10.620
son	6.958	9.983	6.651	13.299	6.548	9.941	5.238	14.283
thy	6.765	8.605	5.687	12.082	4.976	5.946	3.976	9.618
ttt	6.879	9.836	5.974	12.491	4.974	6.526	4.006	14.016
veh	6.530	9.992	6.628	12.391	6.595	9.980	4.135	14.280
vote	6.475	9.309	6.281	12.486	4.978	5.744	3.940	11.607
vow	7.518	9.992	6.653	12.458	6.432	9.995	5.223	14.260
w21	8.022	12.662	6.517	13.234	7.348	9.995	5.439	14.250
w40	8.427	13.353	6.635	14.062	7.250	11.693	5.487	14.268
wdbc	6.731	9.982	6.650	12.485	6.211	9.997	4.145	11.812
wine	6.564	9.986	5.991	12.494	4.983	6.476	4.015	12.377
wpbc	6.797	9.985	6.570	15.069	6.465	9.993	3.969	14.786
zoo	6.973	9.997	5.910	13.284	4.990	9.983	4.063	14.235
AVG [ns]	7.088	9.901	6.355	12.342	5.966	8.518	4.504	12.757
AVG [MHz]	141.09	101.00	157.36	81.02	167.62	117.40	222.03	78.39

Tabela **Error! No text of specified style in document.**.12 Ubrzanje hardverskih u odnosu na softversku PC implementaciju

Test skup	PC							
	SMPL1-S	*SMPL2-S*	*UN1-S*	*UN2-S*	*SMPL1-P*	*SMPL2-P*	*UN1-P*	*UN2-P*
ausc	3.98	41.08	0.68	5.52	125.81	1233.33	27.05	161.23
bc	7.07	26.08	1.07	3.65	286.96	1088.78	42.42	108.08
bcw	2.49	16.31	0.82	3.19	96.32	806.37	30.01	99.21
bsc	3.80	25.30	0.78	3.64	151.74	1165.57	34.05	97.32
car	5.86	32.83	0.86	3.49	193.65	1172.26	35.38	117.76
cmc	5.02	47.96	0.57	3.20	203.44	1437.77	20.02	77.81
ger	3.44	63.61	0.57	7.51	110.94	1895.29	25.63	200.83
gls	5.93	39.79	0.98	5.06	184.62	1492.73	46.09	135.30
hep	1.43	18.21	0.98	7.93	55.70	925.29	43.18	237.84
hrtc	4.09	43.95	0.77	5.69	137.62	1316.91	35.76	154.42
hrts	3.37	34.79	0.92	6.25	140.20	1376.72	40.25	176.87
ion	1.87	42.70	0.56	10.42	56.65	1283.95	25.48	276.62
irs	5.76	23.66	2.11	6.98	241.05	908.40	102.18	286.87
liv	8.70	48.71	1.19	4.85	287.33	1739.39	49.10	163.44
lym	2.43	31.61	0.73	7.40	99.85	982.87	34.09	194.18
page	6.69	59.47	0.81	5.37	222.15	1696.53	30.06	154.96
pid	6.63	46.73	0.91	5.14	231.39	1399.96	37.01	154.64
son	1.12	47.54	0.48	14.75	35.64	1432.21	18.41	411.91
thy	4.86	22.92	2.65	7.49	198.17	995.05	113.77	282.18

ttt	4.16	29.07	1.05	5.01	172.48	1314.61	46.86	133.94
veh	4.88	60.61	0.51	5.23	144.99	1820.39	24.76	136.21
vote	1.90	22.41	1.02	8.75	73.95	1089.54	48.92	282.30
vow	6.83	71.93	0.91	6.82	239.45	2157.24	34.85	178.73
w21	21.82	304.18	2.17	23.47	714.76	11560.28	77.87	653.90
w40	15.47	400.40	1.92	37.20	539.60	13717.18	69.76	1099.96
wdbc	1.43	29.85	0.53	8.81	46.42	894.12	25.67	279.24
wine	2.09	19.21	1.14	7.68	82.50	888.75	51.20	232.51
wpbc	1.85	42.71	0.58	8.58	58.20	1280.18	28.73	262.18
zoo	3.03	38.04	0.95	7.61	127.02	1142.85	41.49	213.16
AVG	5.10	59.71	1.01	8.16	181.33	2076.36	42.76	240.12

Rezultati prikazani u tabeli 3.12:

- osam arhitektura za implementaciju ansambala stabala odluka realizovano je pomoću FPGA komponente familije *Virtex 5* koja je u proseku radila na sledećim učestanostima:
    - *SMPL1-S* arhitektura, 141.09 MHz
    - *SMPL2-S* arhitektura, 101.00 MHz
    - *UN1-S* arhitektura, 157.36 MHz
    - *UN2-S* arhitektura, 81.02 MHz
    - *SMPL1-P* arhitektura, 167.62 MHz
    - *SMPL2-P* arhitektura, 117.40 MHz
    - *UN1-P* arhitektura, 222.03 MHz
    - *UN2-P* arhitektura, 78.39 MHz
- učestanost *Pentium D 820* mikroprocesora na kome je izvršavana softverska implementacija iznosila je 2.8 GHz,

Kao što se može videti iz tabele 3.12, svih osam arhitektura pruža znatna ubrzanja u klasifikaciji instanci pomoću ansambala stabala odluka u odnosu na PC.

U slučaju poređenja sa PC implementacijom, prosečno skraćenje vremena potrebnog za klasifikovanje instance pomoću ansambla stabala odluka iznosi:

- 5.10 puta u slučaju korišćenja *SMPL1-S* arhitekture,
- 59.71 puta u slučaju korišćenja *SMPL2-S* arhitekture,
- 1.01 puta u slučaju korišćenja *UN1-S* arhitekture,
- 8.16 puta u slučaju korišćenja *UN2-S* arhitekture,
- 181.33 puta u slučaju korišćenja *SMPL1-P* arhitekture,
- 2076.36 puta u slučaju korišćenja *SMPL2-P* arhitekture,
- 42.76 puta u slučaju korišćenja *UN1-P* arhitekture,
- 240.12 puta u slučaju korišćenja *UN2-P* arhitekture.

Čak i u poređenju sa jednim od trenutno najjačih desktop računara, arhitekture sa paralelnom evaluacijom članova ansambla, implementirane pomoću FPGA komponenti, pružaju značajna skraćenja u vremenu klasifikacije instanci. Arhitekture sa serijskom evaluacijom nude manja ubrzanja u odnosu na PC implementaciju, međutim treba imati na umu da one zahtevaju znatno manje hardverskih resursa.

Reference

[1] Quinlan J. R., Induction of decision trees, Machine Learning, 1, 1986., pp. 81-106.

[2] Quinlan J. R., C4.5: Programs for Machine Learning, San Mateo, CA: Morgan Kaufmann, 1993

[3] Heath D., Kasif S., Salzberg S., "Induction of oblique decision trees", *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, 1993, pp. 1002-1007.

[4] B.V. Dasarathy and B.V. Sheela, "Composite classifier system design: Concepts and methodology," Proceedings of the IEEE, vol. 67, no. 5, pp. 708–713, 1979.

[5] L.K. Hansen and P. Salamon, "Neural network ensembles," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993–1001, 1990.

[6] R.E. Schapire, "The strength of weak learnability," Machine Learning, vol. 5, no. 2, pp. 197–227, 1990.

[7] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive mixtures of local experts," Neural Computation, vol. 3, pp. 79–87, 1991.

[8] M.J. Jordan and R.A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," Neural Computation, vol. 6, no. 2, pp. 181–214, 1994.

[9] J.A. Benediktsson and P.H. Swain, "Consensus theoretic classification methods," IEEE Trans. on Systems, Man and Cybernetics, vol. 22, no. 4, pp. 688–704, 1992.

[10] L. Xu, A. Krzyzak, and C.Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," IEEE Transactions on Systems, Man and Cybernetics, vol. 22, no. 3, pp. 418–435, 1992.

[11] T.K. Ho, J.J. Hull, and S.N. Srihari, "Decision combination in multiple classifier systems," IEEE Trans. on Pattern Analy. Machine Intel., vol. 16, no. 1, pp. 66–75, 1994.

[12] G. Rogova, "Combining the results of several neural network classifiers," Neural Networks, vol. 7, no. 5, pp. 777–781, 1994.

[13] L. Lam and C.Y. Suen, "Optimal combinations of pattern classifiers," Pattern Recognition Letters, vol. 16, no. 9, pp. 945–954, 1995.

[14] K. Woods, W.P.J. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 4, pp. 405–410, 1997.

[15] I. Bloch, "Information combination operators for data fusion: A comparative review with classification," IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans., vol. 26, no. 1, pp. 52–67, 1996.

[16] S.B. Cho and J.H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," IEEE Transactions on Systems, Man and Cybernetics, vol. 25, no. 2, pp. 380–384, 1995.

[17] L.I. Kuncheva, J.C. Bezdek, and R. Duin, "Decision templates for multiple classifier fusion: An experimental comparison," Pattern Recognition, vol. 34, no. 2, pp. 299–314, 2001.

[18] H. Drucker, C. Cortes, L.D. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," Neural Computation, vol. 6, no. 6, pp. 1289–1301, 1994.

[19] R. Battiti and A.M. Colla, "Democracy in neural nets: Voting schemes for classification," Neural Networks, vol. 7, no. 4, pp. 691–707, 1994.

[20] L.I. Kuncheva, "Classifier ensembles for changing environments," 5th Int. Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, F. Roli, J. Kittler, and T. Windeatt, Eds., vol. 3077, pp. 1–15, 2004.

[21] E. Alpaydin and M.I. Jordan, "Local linear perceptrons for classification," IEEE Transactions on Neural Networks, vol. 7, no. 3, pp. 788–792, 1996.

[22] F. Roli, G. Giacinto, and G. Vernazza, "Methods for designing multiple classifier systems," 2nd Int. Workshop on Multiple Classifier Systems, in Lecture Notes in Computer Science, J. Kittler and F. Roli, Eds., vol. 2096, pp. 78–87, 2001.

[23] G. Giacinto and F. Roli, "Approach to the automatic design of multiple classifier systems," Pattern Recognition Letters, vol. 22, no. 1, pp. 25–33, 2001.

[24] L. Breiman, "Bagging predictors," Machine Learning, vol. 24, no. 2, pp. 123–140, 1996.

[25] Y. Freund and R.E. Schapire, "Decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119–139, 1997.

[26] F.M. Alkoot and J. Kittler, "Experimental evaluation of expert fusion strategies," Pattern Recognition Letters, vol. 20, no. 11–13, pp. 1361–1369, Nov. 1999.

[27] J. Kittler, M. Hatef, R.P.W. Duin, and J. Mates, “On combining classifiers,” IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 20, no. 3, pp. 226–239, 1998.
[28] M. Muhlbaier, A. Topalis, and R. Polikar, “Ensemble confidence estimates posterior probability,” 6th Int. Workshop on Multiple Classifier Sys., in Lecture Notes in Comp. Science, N. Oza, R. Polikar, J. Kittler, and F. Roli, Eds., vol. 3541, pp. 326–335, 2005.
[29] J. Grim, J. Kittler, P. Pudil, and P. Somol, “Information analysis of multiple classifier fusion,” 2nd Int. Workshop on Multiple Classifier Systems, in Lecture Notes in Computer Science, J. Kittler and F. Roli, Eds., vol. 2096, pp. 168–177, 2001.
[30] L.I. Kuncheva, “A theoretical study on six classifier fusion strategies,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 2, pp. 281–286, 2002.
[31] S.B. Cho and J.H. Kim, “Multiple network fusion using fuzzy logic,” IEEE Transactions on Neural Networks, vol. 6, no. 2, pp. 497–501, 1995.
[32] M. Grabisch and J.M. Nicolas, “Classification by fuzzy integral: performance and tests,” Fuzzy Sets and Systems, vol. 65, no. 2–3, pp. 255–271, 1994.
[33] Y. Lu, “Knowledge integration in a multiple classifier system,” Applied Intelligence, vol. 6, no. 2, pp. 75–86, 1996.
[34] T.K. Ho, “Random subspace method for constructing decision forests,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp. 832–844, 1998.
[35] D.H. Wolpert, “Stacked generalization“, Neural Networks, vol. 5, pp. 241-259, 1992.
[36] R. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “ Adaptive mixtures of local experts“, Neural Computation, vol. 3, pp. 79-87, 1991.
[37] G. Brown, “Diversity in neural network ensembles,” Ph.D. dissertation, University of Birmingham, 2004.
[38] B. Rosen , “Ensemble learning using decorrelated neural networks,” Connection Science, vol. 8, no. 3, pp. 373–384, 1996.
[39] P.J. Boland, “Majority system and the Condorcet jury theorem,” Statistician, vol. 38, no. 3, pp. 181–189, 1989.
[40] L. Shapley and B. Grofman, “Optimizing group judgmental accuracy in the presence of interdependencies,” Public Choice (Historical Archive), vol. 43, no. 3, pp. 329–343, 1984.
[41] L.I. Kuncheva, Combining Pattern Classifiers, Methods and Algorithms. New York, NY: Wiley Interscience, 2005.
[42] N. Littlestone and M.Warmuth, “Weighted majority algorithm,” Information and Computation, vol. 108, pp. 212–261, 1994.
[43] Y.S. Huang and C.Y. Suen, “Behavior-knowledge space method for combination of multiple classifiers,” Proc. of IEEE Computer Vision and Pattern Recog., pp. 347–352, 1993.
[44] Y.S. Huang and C.Y. Suen, “A method of combining multiple experts for the recognition of unconstrained handwritten numerals,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 1, pp. 90–94, 1995.
[45] Bermak, A., Martinez, D., “A Compact 3D VLSI Classifier using Bagging Threshold Network Ensembles”, IEEE Trans. on Neural Networks, vol. 14, no. 5, September 2003, pp. 1097-1109.
[46] R. Struharik, “Digitalna elektronska kola za realizaciju stabala odluka”, doktorska disertacija, FTN, 2009.
[47] A. Asuncion, D.J. Newman, UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science, 2007 http://www.ics.uci.edu/~mlearn/MLRepository.html

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

Трг Доситеја Обрадовића 6, 21000 Нови Сад, Република Србија
Деканат: 021 6350-413; 021 450-810; Централа: 021 485 2000
Рачуноводство: 021 458-220; Студентска служба: 021 6350-763
Телефакс: 021 458-133; e-mail: ftndean@uns.ac.rs

Наш број:
Ваш број:
Датум: 2012-12-07

ИЗВОД ИЗ ЗАПИСНИКА

Наставно-научног већа Факултета техничких наука у Новом Саду, на 2. редовној седници одржаној дана 28.11.2012. године, донело је следећу одлуку:

-непотребно изостављено-

Тачка 14.2.2. Питања научноистраживачког рада и међународне сарадње / верификација нових техничких решења

У циљу доношења одлуке о прихватању ***техничког решења –под називом:***

ИП ЈЕЗГРА ЗА ХАРДВЕРСКУ РЕАЛИЗАЦИЈУ АНАСАМБАЛА СТАБАЛА ОДЛУКА

именују се рецензенти:

- Проф. др Драган Васиљевић, Електротехнички факултет у Београду
- Проф. др Теуфик Токић. Електронски факултет у Нишу

Аутори техничког решења: доцент др Растислав Струхарик, проф. др Ладислав Новак.

-непотребно изостављено-

Записник водила:
Јасмина Димић, дипл. правник

Тачност података оверава:
Секретар
Иван Нешковић, дипл. правник

Декан
Проф. др Раде Дорословачки

Softver:
IP jezgra za hardversku realizaciju ansambala stabala odluka

Rukovodilac projekta: prof. dr Ljiljana Živanov
Odgovorno lice: dr Rastislav Struharik
Autori: Rastislav Struharik, Ladislav Novak
Fakultet tehničkih nauka (FTN), Novi Sad

Razvijeno: u okviru projekta tehnološkog razvoja TR-32016
Godina: 2011 - 2012.
Primena: jun 2012.

Kratak opis

Hardverske implementacije ansambala stabala odluka mogu predstavljati jedino rešenje u slučajevima kada je potrebno izvršiti klasifikaciju instance u vrlo kratkom vremenu, ili kada je potrebno adaptivno formiranje prediktivnog modela, u toku rada sistema. Ovo su tipični zahtevi koji se sreću prilikom projektovanja savremenih a pogotovo budućih *embedded* sistema. Imajući u vidu ove činjenice, razvijena su IP jezga za hardversku realizaciju ansambala stabala odluka koja u mnogome olakšavaju integraciju i rad sa stablima odluka prilikom projektovanja *embedded* sistema.

Tehničke karakteristike:
IP jezgra za hardversku realizaciju ansambala stabala odluka modelovana su pomoću standardnog jezika za specifikaciju hardvera, VHDL. Razvijeni modeli su parametrizovani što omogućava jednostavno prilagođavanje arhitekture trenutnim potrebama korisnika.

Tehničke mogućnosti:
Korišćenjem konfiguracionih parametara definisanih unutar VHDL modela arhitektura za realizaciju ansambala stabala odluka (broj bitova koji se koristi za predstavu vednosti atributa problema, koeficijenata razdvajajućih hiperpovrši, ciljnih klasa; broja atributa preko kojih je opisan klasifikacioni problem; broj stabala u ansamblu; maksimalnog broja čvorova u realizovanom stablu odluke, itd.) moguće je prilagoditi VHDL model potrebama tekuće aplikacije. Pilikom sinteze hardvera ovi parametri se koriste kako bi se automatski generisala optimalna hardverska implementacija za tekuću aplikaciju.

Realizator:
Fakultet tehničkih nauka – FTN

Korisnik:
Fakultet tehničkih nauka – FTN, Novi Sad

Podtip rešenja:
Softver – M85

Mišljenje

Tehničko rešenje "IP jezgra za hardversku realizaciju ansambala stabala odluka" autora doc. dr Rastislava Struharika, i prof. dr Ladislava Novaka, realizovano 2011.-2012. godine, prikazano je na 41 stranice A4 formata, grupisano je u ukupno pet poglavlja:

1. Opis problema koji se rešava tehničkirn rešenjem,
2. Stanje rešenosti problema u svetu - prikaz i analiza postojećih rešenja,
3. Suština tehničkog rešenja (uključujući i prateće ilustracije i tehničke crteze),
4. Detaljan opis primene tehničkog rešenja i
5. Literatura.

Tehničko rešenje pripada polju tehničko-tehnoloških nauka i oblasti elektrotehničkog inženjerstva. Naručilac tehničkog rešenja je Fakultet tehnilkih nauka u Novom Sadu, Republika Srbija, koji je i korisnik tehničkog rešenja.

Tehničko rešenje je realizovano u okviru projekta "Nove generacije ugrađenih elektronskih komponenti i sistema u neorganskim i organskim tehnologijama za uređaje široke potrošnje" (Broj projekta TR 32016, Program istraživanja u oblasti tehnološkog razvoja za period 2011-2014., Tehnološka oblast - Elektronika, telekomunikacije i informacione tehnologije, Rukovodilac projekta: dr Ljiljana Živanov, redovni profesor).

Na osnovu analize tehničkog rešenja "IP jezgra za hardversku realizaciju ansambala stabala odluka" autora doc. dr Rastislava Struharika, i prof. dr Ladislava Novaka, mogu se izvesti sledeći zaključci:
1. Dokumentacija tehničkog rešenja jasno prikazuje kompletnu strukturu tehničkog rešenja – opis problema, daje dataljniji osvrt na stanje u svetu, sadrži odgovarajući prikaz teorijskih osnova na kojima je zasnovano tahničko rešenje i posebno detaljno prikazuje strukturu i primenu realizovanog tehničkog rešenja.
2. Predloženo tehničko rešenje, "IP jezgra za hardversku realizaciju ansambala stabala odluka", predstavlja efikasan alat za rešavanje problema u oblasti hardverske implementacije jedne grupe algoritama mašinskog učenja, ansambala stabala odluka.
3. Tehničko rešenje karakteriše originalan naučni doprinos koji ima izraženu praktičnu dimenziju budući da se kroz korišćenje niza konfiguracionih parametara omogućava njegova fleksibilna i univerzalnija primena.

Na osnovu prethodnog, predlažem da se "IP jezgra za hardversku realizaciju ansambala stabala odluka", autora doc. Dr Rastislava Struharika, i prof. Dr Ladislava Novaka, prihvati kao novo tehničko rešenje i u skladu sa Pravilnikom o postupku i načinu vrednovanja, i kvantitativnom iskazivanju naučnoistraživačkih rezultata istraživača ("Službeni glasnik RS", broj 38/2008) klasifikuje kao rezultat "M85 Prototip, nova metoda, softver, standardizovan ili atestiran instrument, nova genska proba, mikroorganizmi".

U Nišu, 17.12.2012. god.

Prof. Dr Teufik Tokić,
Univerzitet u Nišu
Elektronski fakultet

Softver:
IP jezgra za hardversku realizaciju ansambala stabala odluka

Rukovodilac projekta: prof. dr Ljiljana Živanov
Odgovorno lice: dr Rastislav Struharik
Autori: Rastislav Struharik, Ladislav Novak
Fakultet tehničkih nauka (FTN), Novi Sad

Razvijeno: u okviru projekta tehnološkog razvoja TR-32016
Godina: 2011 - 2012.
Primena: jun 2012.

Kratak opis

Hardverske implementacije ansambala stabala odluka mogu predstavljati jedino rešenje u slučajevima kada je potrebno izvršiti klasifikaciju instance u vrlo kratkom vremenu, ili kada je potrebno adaptivno formiranje prediktivnog modela, u toku rada sistema. Ovo su tipični zahtevi koji se sreću prilikom projektovanja savremenih a pogotovo budućih *embedded* sistema. Imajući u vidu ove činjenice, razvijena su IP jezga za hardversku realizaciju ansambala stabala odluka koja u mnogome olakšavaju integraciju i rad sa stablima odluka prilikom projektovanja *embedded* sistema.

Tehničke karakteristike:
IP jezgra za hardversku realizaciju ansambala stabala odluka modelovana su pomoću standardnog jezika za specifikaciju hardvera, VHDL. Razvijeni modeli su parametrizovani što omogućava jednostavno prilagođavanje arhitekture trenutnim potrebama korisnika.

Tehničke mogućnosti:
Korišćenjem konfiguracionih parametara definisanih unutar VHDL modela arhitektura za realizaciju ansambala stabala odluka (broj bitova koji se koristi za predstavu vednosti atributa problema, koeficijenata razdvajajućih hiperpovrši, ciljnih klasa; broja atributa preko kojih je opisan klasifikacioni problem; broj stabala u ansamblu; maksimalnog broja čvorova u realizovanom stablu odluke, itd.) moguće je prilagoditi VHDL model potrebama tekuće aplikacije. Pilikom sinteze hardvera ovi parametri se koriste kako bi se automatski generisala optimalna hardverska implementacija za tekuću aplikaciju.

Realizator:
Fakultet tehničkih nauka – FTN

Korisnik:
Fakultet tehničkih nauka – FTN, Novi Sad

Podtip rešenja:
Softver – M85

Mišljenje

Fakultet tehničkih nauka je razvio IP jezgra za hardversku implementaciju ansambala stabala odluka. IP jezgra su opisana korišćenjem VHDL jezika za modelovanje hardvera. Razvijeni modeli jezgara se vrlo lako mogu prilagoditi trenutnim potrebama aplikacije korišćenjem konfiguracionih parametara. Na ovaj način omogućena je široka upotreba ovih jezgara u velikom broju različitih aplikacija.

U predloženom tehničkom rešenju razmatran je problem hardverske implementacije ansambala ortogonalnih, neortogonalnih i nelinearnih stabala odluka. Analizom postojećih rešenja utvrđeno je da u dostupnoj literaturi postoje samo jedan rad koji se bavi problematikom hardverske implementacije ansambala stabala odluka, ali se predloženo rešenje može primeniti samo za implementaciju ansambala ortogonalnih stabala odluka korišćenjem prostog većinskog odlučivanja. Postojeća arhitektura zasnovana na vrlo neefikasnom pristupu za hardversku implementaciju ansambala stabala odluka koji se bazira na implementiranju svih čvorova svakog stabla odluke kao individualnih modula koji su zatim međusobno povezani na način koji odražava topologiju posmatranog stabla odluke. Ovakav pristup implementaciji bio bi opravdan u slučaju da je stablo odluke sistem sa visokim stepenom paralelizma, kao što je to slučaj sa veštačkim neuronskim mrežama kod kojih su u proces klasifikacije svake instance uključeni svi neuroni iz mreže. Međutim, prilikom klasifikacije instance pomoću stabla odluke nije neophodno izračunati izlaz svakog od čvorova stabla. Naprotiv, da bi se klasifikovala bilo koja instanca korišćenjem stabla odluke dovoljno je evaluirati samo one čvorove u stablu koji se nalaze na tačno jednoj putanji od korena do jednog lista stabla odluke.

Tehničko rešenje predlaže čitav niz različitih arhitektura za harversku implementaciju ansambala stabala odluka. Pretpostavka je da je ansambl već formiran, odnosno da je poznata stuktura svakog od stabala odluka koji čine ansambl, kao i da je odabran algoritam za kombinovanje predikcija članova ansambla koji se želi koristiti. Svaka arhitektura za implementaciju ansambla sastoji se iz dva modula: modula koji implementira individualne članove ansambla i modula koji kombinuje predikcije individualnih članova. Obzirom da se ansambl uvek ima više od jednog člana, svaki od ovih modula može se realizovati na dva načina: serijski i paralelno.

U slučaju serijske implementacije, individualni članovi evaluiraju se serijski, jedan nakon drugog, a i njihove predikcije se kombinuju serijski. Modul za određivanje predikcija članova ansambla serijski evaluira članove ansambla i njihove predikcije prosleđuje ka modulu za kombinovanje predikcije serijski, jednu nakon druge. Modul za kombinovanje predikcija u ovom slučaju mora da prihvati predikcije članova ansambla, i da ih zatim kombinuje u jednu, zajedničku klasifikaciju čitavog ansambla, pomoću odabranog algoritma za kombinovanje.

U slučaju paralelne realizacije, svaki od individualnih prediktivnih modela evaluira se u paraleli, a predikcije individualnih članova ansambla se takođe kombinuju u paraleli u cilju određivanja predikcije čitavog ansambla. U ovom slučaju modul za određivanje predikcija članova ansambla istovremeno evaluira sve članove i njihove predikcije prosleđuje ka modulu za kombinovanje predikcija istovremeno, u paraleli. Modul za kombinovanje predikcija u istom trenutku prihvata predikcije svih članova ansambla i u paraleli ih kombinuje pomoću odabranog algoritma za kombinovanje da bi odredio zajedničku predikciju čitavog ansambla.

U slučaju serijske arhitekture, modul za određivanje predikcija se zapravo sastoji od samo jednog modula za hardversku implementaciju stabla odluke, baziranog na nekoj od modifikovanih SMPL ili UN arhitektura koji se koristi da se serijski evaluira individualna stabla iz ansambla. Modifikacija postojećih SMPL i UN arhitektura je neophodna jer sada moraju čuvati strukturne podatke ne samo za jedno stablo, već za čitav ansambl. Korišćenjem ovih modifikovanih arhitektura predložene su četiri različite arhitekture za paralelnu evaluaciju članova ansambla: SMPL1-S, SMPL2-S, UN1-S i UN2-S.

Kod paralelne arhitekture modul za određivanje predikcija se sastoji iz N nezavisnih modula baziranih na originalnim SMPL i UN arhitekturama koji u paraleli evaluiraju članove ansambla. Kao i

u slučaju serijske arhitekture, i u slučaju paralelne arhitekture mogu se definisati ukupno četiri različite arhitekture, bazirane na postojećim SMPL i UN arhitekturama: SMPL1-P, SMPL2-P, UN1-P i UN2-P.

U tehničkom rešenju takođe je predložen niz arhitektura za hardversku implementaciju algoritama za kombinovanje predikcija individualnih stabala odluka iz ansambla. Predložene su arhitekture za hardversku implementaciju sledećih algoritama: većinskog odlučivanja u tri varijante: jednoglasno odlučivanje, odlučivanje prostom većinom i većinsko odlučivanje; težinskog većinskog odlučivanja i BKS algoritma. Za svaki od navedenih algoritama predložena je serijska i paralelna arhitektura. Serijske arhitekture namenjene su za korišćenje u slučaju serijske realizacije članova ansambla, dok su paralelne arhitekture namenjene za korišćenje u kombinaciji sa paralalnom implementacijom članova ansambla.

U skladu sa gore iznetim činjenicama tehničko rešenje ispunjava uslove da bude priznato kao softver (odnosno M85 u skladu sa Oravilnikom o postupku i načinu vrendovanja i kvantitativnom iskazivanju naučnoistraživačkih rezultata istraživača, Sl. gl. RS br. 38/08).

Dr Dragan Vasiljević, dipl. el. inž.

Redovni profesor Elektrotehničkog fakulteta, Beograd

Dragan V. Vasiljević

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
Трг Доситеја Обрадовића 6, 21000 Нови Сад, Република Србија
Деканат: 021 6350-413; 021 450-810; Централа: 021 485 2000
Рачуноводство: 021 458-220; Студентска служба: 021 6350-763
Телефакс: 021 458-133; e-mail: ftndean@uns.ac.rs

Наш број: 01.сл
Ваш број:
Датум: 2012-12-28

ИЗВОД ИЗ ЗАПИСНИКА

Наставно-научног већа Факултета техничких наука у Новом Саду, на 3 редовној седници одржаној дана 26.12.2012. године, донело је следећу одлуку:

-непотребно изостављено-

Тачка 14.1.6. Питања научноистраживачког рада и међународне сарадње / верификација нових техничких решења

Одлука

На основу позитивног извештаја рецензената прихвата се
техничко решење – (М85) под називом:

ИП ЈЕЗГРА ЗА ХАРДВЕРСКУ РЕАЛИЗАЦИЈУ АНАСАМБАЛА СТАБАЛА ОДЛУКА

Аутори техничког решења: доцент др Растислав Струхарик, проф. др Ладислав Новак.

-непотребно изостављено-

Записник водила:

Јасмина Димић, дипл. правник

Тачност података оверава:
Секретар

Иван Нешковић, дипл. правник

Декан

Проф. др Раде Дорославачки