TEHNIČKO REŠENJE

IP jezgra za hardversku realizaciju 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 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 stabala odluka koja u mnogome olakšavaju integraciju i rada sa stablima odluka prilikom projektovanja *embedded* sistema.

Tehničke karakteristike:

IP jezgra za hardversku realizaciju 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 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; 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

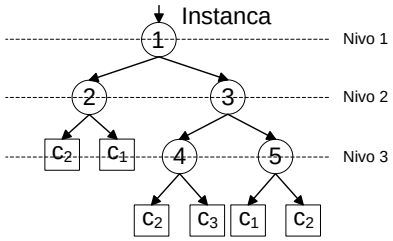
Projekat u okviru koga je realizovano tehničko rešenje:

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.

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

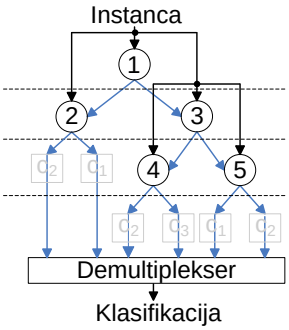
Funkcija *f* može biti bilo koja funkcija *n* atributa. Sada se prilikom particije hiperprostora atributa koristi hiperpovrš definisana funkcijom *f*. Ako je funkcija *f* nelinearna funkcija, stabla koja koriste takve testove se zovu *stabla odluka sa nelinearnim hiperpovršima*.

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.

U slučaju hardverske implementacije prvo rešenje koje se nameće jeste da se čitavo stablo odluke direktno implementira u hardveru. U ovom slučaju potrebno je implementirati svaki od čvorova kao nezavisan modul i izvršiti njihovo povezivanje. Arhitektura bazirana na ovom principu predložena je u radu [4]. Implementacija stabla odluke sa slike 2.1 korišćenjem arhitekture predložene u radu [4] prikazana je na slici 2.2.



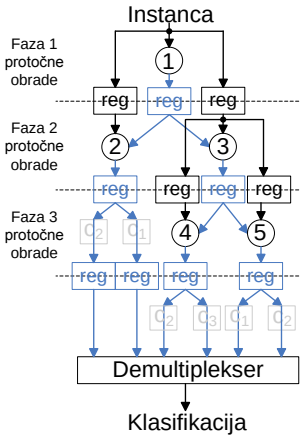
2.2 Implementacija stabla odluke korišćenjem arhitekture [4]

Kao što se na slici 2.2 može videti, arhitektura [4] zahteva implementaciju svakog od čvorova stabla odluke. Listove stabla odluke nije potrebno implementirati. Na ulaz svakog čvora dovodi se instanca koju je potrebno klasifikovati, odnosno vektor atributa pomoću kojega je definisana instanca (crne strelice na slici 2.2). Svaki od unutrašnjih čvorova stabla odluke, osim korena, pored ulaza za instancu ima još jedan, kontrolni ulaz i dva kontrolna izlaza (plave strelice na slici 2.2.). Reč je o jednobitnim signalima pomoću kojih se kontroliše rad čvorova i obezbeđuje ispravan rad sistema. Ukoliko kontrolni ulaz čvora ima vrednost ‘0’ tada oba kontrolna izlaza imaju vrednost ‘0’. U ovom slučaju posmatrani čvor je neaktivan. U slučaju da kontrolni ulaz ima vrednost ‘1’ čvor je aktivan i jedan od kontrolnih izlaza ima vrednost ‘1’, a drugi ima vrednost ‘0’, u zavisnosti od ishoda testa specificiranog u čvoru. Upotrebom kontrolnih signala ostvaruje se „uključivanje“ samo onih čvorova koji se nalaze na putu od korena stabla do lista kome će posmatrana instanca biti asocirana. Ostali čvorovi su neaktivni.

Kontrolni izlazi svih čvorova koji bi bili spojeni na listove stabla vode se na ulaz demultiplekserskog bloka. Zapravo je reč o konvertoru koda, ali je on u radu [4] označen kao demultiplekser. Zadatak ovog bloka je da na osnovu binarnog vektora dobijenog sjedinjavanjem svih izlaznih kontrolnih signala koji su asocirani listovima realizovanog stabla generiše kod koji odgovara klasi kojoj tekuća instanca pripada.

Arhitektura predložena u [4] ima niz nedostataka. Kao što je već rečeno, ova arhitektura zahteva implementaciju svakog čvora iz stabla odluke kao nezavisnog modula. Ovo može biti ozbiljan nedostatak, pogotovo ukoliko se hardverski realizuju stabla sa velikim brojem čvorova. Drugi nedostatak odnosi se na brzinu klasifikacije instanci. U slučaju arhitekture [4] vreme potrebno za klasifikaciju u najgorem slučaju proporcionalno je dubini implementiranog stabla odluke i vremenu potrebnom da se izračuna test u jednom čvoru stabla. U konkretnim aplikacijama dubina stabla može biti velika, pa će samim tim i vreme klasifikacije instance biti veliko, što može predstavljati ozbiljan nedostatak arhitekture [4].

Broj klasifikovanih instanci u jedinici vremena za arhitekturu [4] može se značajno povećati ukoliko se ona modifikuje tako da koristi protočnu obradu (*pipelining*). Modifikovana arhitektura [4] sa protočnom obradom prikazana je na slici 2.3.

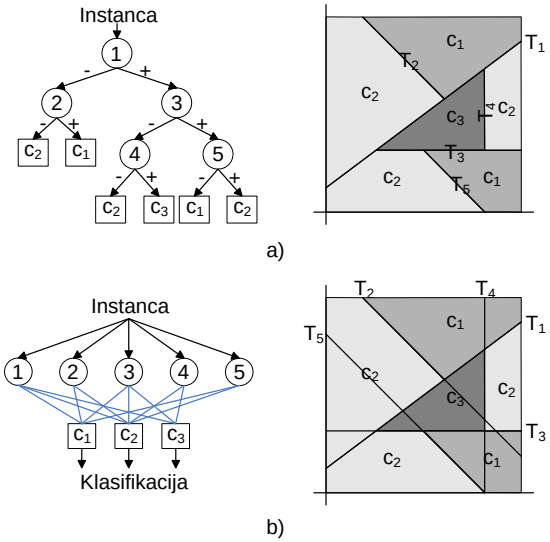


Slika 2.3Implementacija stabla odluke korišćenjem modifikovane arhitekture [4] sa protočnom obradom

Nakon svakog nivoa unutar stabla odluke ubačen je red registara, tako da je kritična putanja sistema koja je ranije bila jednaka dubini implementiranog stabla sada redukovana na samo jedan nivo. Broj faza protočne obrade jednak je broju nivoa u implementiranom stablu odluke. U ovom slučaju nije potrebno sačekati da se klasifikuje jedna instanca da bi se započelo sa klasifikacijom sledeće. Na primer, dok se jedna instanca nalazi u trećoj fazi protočne obrade, druga instanca se nalazi u drugoj fazi a treća instanca se nalazi u prvoj fazi protočne obrade.

Vreme klasifikacije modifikovane arhitekture [4] jednako je vremenu koje je potrebno da se izračuna test u jednom čvoru stabla. Ovo predstavlja značajno skraćenje u odnosu na originalnu arhitekturu predloženu u [4]. Bitno je naglasiti da je inicijalno kašnjenje modifikovane arhitekture [4], odnosno vreme koje je potrebno da se klasifikuje prva instanca, jednako kašnjenju originalne arhitekture [4]. Ovo je inherentno svim protočnim arhitekturama i ne može se izbeći.

Interesantna arhitektura za hardversku implementaciju stabala odluka predložena je u radu [5]. Ova arhitektura bazira se na korišćenju ekvivalencije između stabala odluka i *threshold* mreža (*threshold networks*). Ova ekvivalencija prvi put je uočena u radovima [6] i [7]. Slika 2.4 prikazuje stablo odluke sa slike 2.1 i njemu ekvivalentnu *threhold* mrežu.



Slika 2.4 Ekvivalencija između stabla odluke i threshold *mreže*

Na slici 2.4a prikazano je stablo odluke sa slike 2.1 i particija prostora atributa pomoću njega. Pretpostavljeno je da se radi o klasifikacionom problemu opisanom pomoću dva atributa te je stoga prostor atributa dvodimenzionalan. Takođe je pretpostavljeno da se particija prostora atributa vrši pomoću neortogonalnih hiperravni (u slučaju dvodimenzionalnog prostora hiperravni su zapravo prave). Karakteristika binarnog stabla odluke jeste da se nakon svakog testa definisanog u posmatranom čvoru vrši podela (particija) prostora atributa na dva regiona. Svako podstablo koje se nalazi ispod posmatranog čvora vrši dalju particiju samo njemu pridruženog regiona a ne čitavog prostora atributa. Ovo se lepo može videti na slici 2.4a gde se samo prava asocirana testu T_1 proteže čitavom dužinom 2D prostora atributa, a sve ostale prave asocirane testovima T_2, T_3, T_4 i T_5 su zapravo poluprave ili duži.

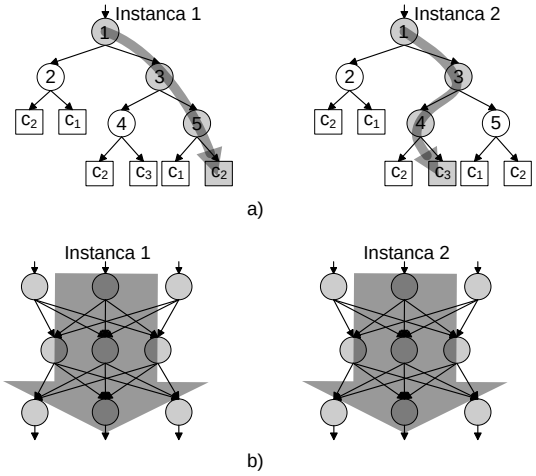
Ista particija prostora može se izvršiti i korišćenjem threshold mreže koja je ekvivalentna posmatranom stablu odluke. Threshold mreža ekvivalentna stablu odluke sa slike 2.4a prikazana je na slici 2.4.b zajedno sa particijom

prostora atributa koji se dobija pomoću nje. Threshold mreža uvek se sastoji iz dva nivoa. U prvom nivou nalaze se svi čvorovi iz stabla odluke, a u drugom nivou nalaze se logičke mreže koje kombinuju potrebne izlaze čvorova iz prvog nivoa. Broj kombinacionih mreža u drugom nivou odgovara broju klasa posmatranog klasifikacionog problema. Glavna prednost threshold mreže nalazi se u činjenici da je čitava struktura stabla odluke sažeta u samo jedan sloj unutar threshold mreže. Svaki čvor iz prvog nivoa threshold mreže realizuje po jedan test iz ekvivalentnog stabla odluke. Izlaz svakog čvora je 1-bitni signal koji prenosi informaciju da li je tekuća instanca zadovoljila test definisan u posmatranom čvoru ili nije. Obzirom da u threshold mreži ne postoji hijerarhija čvorova, kao što je slučaj kod stabla odluke, svakom testu odgovara po jedna hiperravan koja se proteže duž čitavog prostora atributa. Na slici 2.4b se to može videti, jer svakom od testova T1, T2, T3, T4 i T5 odgovara po jedna prava koja se proteže duž čitavog 2D prostora atributa. Kombinacione mreže u drugom sloju threshold mreže imaju zadatak da zapravo izvrše particiju prostora atributa na način identičan sa onim koji se postiže pomoću stabla odluke. Pri tome kao svoje ulaze koriste 1-bitne signale generisane od strane čvorova iz prvog sloja, koji su na slici 2.4b označeni sa plavom bojom.

Stablo odluke realizovano pomoću arhitekture [5] takođe zahteva da se svaki čvor stabla odluke realizuje kao poseban modul, isto kao i u slučaju arhitekture [4]. Međutim, brzina klasifikacije arhitekture [5] znatno je veća od brzine klasifikacije arhitekture [4], jer se sada svako stablo odluke realizuje kao dvoslojna *threshold* mreža. Ovo je glavna prednost arhitekture [5] u odnosu na arhitekturu [4].

3. IP jezgra za hardversku realizaciju stabala odluka

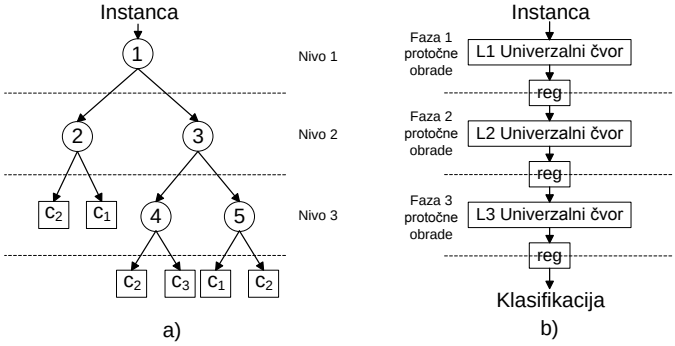
Arhitekture [4] i [5] predstavljaju jedine dve arhitekture koje su predložene za implementaciju stabala odluka u dostupnoj literaturi. Obe ove arhitekture zapravo su zasnovane na vrlo neefikasnom pristupu za hardversku implementaciju stabala odluka koji se bazira na implementiranju svih čvorova stabla 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. Proces klasifikacije instance pomoću veštačke neuronske mreže i stabla odluke prikazan je na slici 3.1. Čvorovi označeni sivom bojom su oni koje je potrebno evaluirati prilikom klasifikovanja posmatrane instance.



Slika 3.1 Način klasifikacije instance korišćenjem: a) stabla odluke, b) veštačke neuronske mreže

Kao što se može videti sa slike 3.1, stablo odluke je za razliku od neuronske mreže, sistem gde se operacije odvijaju serijski, jedna nakon druge. Na primer, ako posmatramo levo stablo sa slike 3.1a prilikom klasifikovanja instance 1 prvo je potrebno evaluirati čvor 1, nakon njega čvor 3 i na kraju čvor 5. Svi ostali čvorovi u ovom slučaju su neaktivni, i nema potrebe za njihovom evaluacijom. U slučaju klasifikacije instance 2, potrebno je evaluirati čvorove 1, 3 i 4 upravo u tom redosledu, a preostali čvorovi su neaktivni i ne treba ih evaluirati. Za razliku od stabla odluke, u slučaju klasifikacije instanci 1 i 2 pomoću ANN, i u jednom i u drugom slučaju neophodno je evaluirati sve neurone iz mreže, što je prikazano na slici 3.1b.

Arhitektura koja bi uzimala u obzir činjenicu da je prilikom klasifikacije bilo koje instance dovoljno evaluira samo podskup svih čvorova u stablu odluke bila bi mnogo efikasnija u pogledu zahtevanih hardverskih resursa u odnosu na ranije predložene arhitekture [4] i [5]. Princip rada ove arhitekture prikazan je na slici 3.2. Umesto da se implementira svaki čvor u stablu odluke kao nezavisan modul, predložena arhitektura implementira samo po jedan univerzalni čvor u svakom od nivoa stabla odluke. Ova arhitektura nazvana je *SMPL* (*Single Module Per Layer*).

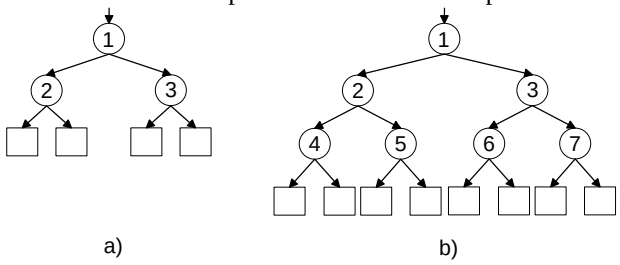


Slika 3.2 Konceptualni izgled SMPL *arhitekture za realizaciju stabla odluke*

SMPL arhitektura sastoji se od niza univerzalnih čvorova čiji broj je jednak dubini stabla odluke koje se implementira. Svaki od univerzalnih čvorova asociran je svim čvorovima iz odgovarajućeg nivoa stabla odluke i u stanju je da evaluira sve testove koji su u njima specificirani. Na primer, u slučaju implementacije stabla odluke prikazanog na slici 3.2a pomoću *SMPL* arhitekture potrebno je koristiti ukupno tri univerzalna čvora, L1, L2 i L3. Univerzalni čvor L1 u stanju je da evaluira sve čvorove koji je nalaze u prvom nivou stabla odluke, u ovom slučaju to je samo koren stabla, odnosno čvor 1. Univerzalni čvor L2 u stanju je da evaluira sve čvorove koji se nalaze u drugom nivou stabla odluke, odnosno čvorove 2 i 3. Konačno, univerzalni čvor L3 u stanju je da evaluira sve čvorove koji se nalaze u trećem nivou stabla odluke sa slike 2.7a, a to su u ovom slučaju čvorovi 4 i 5.

SMPL arhitektura zahteva značajno manje hardverske resurse od arhitektura [4] i [5], jer je u slučaju *SMPL* arhitekture, u svakom od nivoa stabla potrebno realizovati samo po jedan modul za evaluaciju testova (2.1), (2.2) ili (2.4). Pošto svaki od ovih testova zahteva izvođenje operacija sabiranja, množenja i poređenja, *SMPL* arhitektura će zahtevati znatno manji broj sabirača, množača i komparatora od ranije predloženih arhitektura [4] i [5]. Bitno je naglasiti da su memorijski resursi neophodni za smeštanje koeficijenata koji definišu testove isti i u slučaju *SMPL* arhitekture i arhitektura predloženih u [4] i [5]. Ovo je sasvim razumljivo, obzirom da je svako stablo odluke zapravo jednoznačno definisano preko skupa svih koeficijenata koji se nalaze u testovima unutar svih čvorova stabla odluke i načinom na koji su čvorovi međusobno povezani. Svaka modifikacija broja ili vrednosti koeficijenata, a samim tim i smanjenje njihovog broja, vodila bi ka realizaciji nekog drugog a ne željenog stabla odluke, te stoga nije ni moguća. Takođe je bitno naglasiti da pored toga što *SMPL* arhitektura predstavlja efikasniju arhitekturu za realizaciju stabala odluka od arhitektura [4] i [5], njena brzina klasifikacije je ista kao i u slučaju arhitektura [4] i [5].

Ušteda koja se može postići korišćenjem *SMPL* arhitekture u odnosu na arhitekture [4] i [5] naravno zavisi od strukture konkretnog stabla koje se implementira. Detaljna teorijska analiza performansi kao i analiza performansi na konkretnim test primerima *SMPL* arhitekture i arhitektura predloženih u radovima [4] i [5] biće izložena kasnije. Međutim, na ovom mestu može se izvršiti procena maksimalne uštede hardverskih resursa prilikom korišćenja *SMPL* arhitekture u slučaju implementacije kompletnih binarnih stabala odluka. Kompletno binarno stablo odluke je puno binarno stablo kod kojega se svi listovi nalaze na istoj dubini. Puno binarno stablo je stablo kod kojega svaki čvor ima tačno dva naslednika. Primeri kompletnih binarnih stabala prikazani su na slici 3.3.



Slika 3.3 Kompletna binarna stabla: a) dubine 2, b) dubine 3

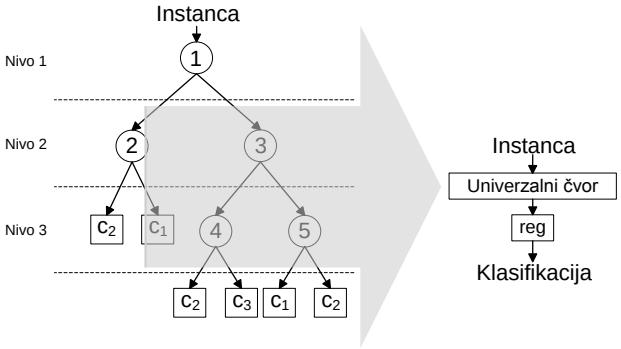
U tabeli 3.1 prikazan je broj neophodnih čvorova koje je potrebno realizovati u slučaju hardverske implementacije kompletnih binarnih stabala odluke različitih veličina korišćenjem arhitektura predloženih u radovima [4] i [5] i *SMPL* arhitekture.

Tabela 3.1 Broj čvorova koje je potrebno implementirati u slučaju realizacije kompletnog binarnog stabla odluke za *SMPL* i arhitekture [4] i [5]

Dubina stabla	**Arhitekture [4] i [5]**	***SMPL* arhitektura**	**Ušteda**
3	7	3	57.1%
4	15	4	73.3%
6	63	6	90.5%
8	255	8	96.9%
10	1023	10	99.0%
12	4095	12	99.7%
14	16383	14	99.9%

Kao što tabela 3.1 pokazuje, ušteda koja se može postići korišćenjem *SMPL* arhitekture u odnosu na postojeće arhitekture prilikom hardverske implementacije stabla odluke je značajna i eksponencijalno se uvećava kako raste dubina stabla. Međutim, ove brojke je potrebno uzeti sa izvesnom dozom rezerve, jer se u praksi retko sreću kompletna binarna stabala odluke. Tabela 3.1 zapravo pokazuje kolika je najveća moguća ušteda koja se može postići korišćenjem *SMPL* arhitekture. Stvarne uštede će biti manje i one će biti prezentovane kasnije.

Sekvenca univerzalnih čvorova na kojoj se bazira *SMPL* arhitektura, može se zameniti samo sa jednim univerzalnim čvorom. Tako dolazimo do *UN* arhitekture koja je prikazana na slici 3.4.



Slika 3.4 Konceptualni izgled UN *arhitekture za realizaciju stabla odluke*

U slučaju *UN* arhitekture jedan univerzalni čvor koristi se za evaluaciju testova koji se nalaze u svim čvorovima stabla odluke. Testovi se evaluiraju sekvencijalno, počevši od korena stabla. U zavisnosti od ishoda testa evaluira se test definisan u jednom od dva čvora naslednika. Ovaj postupak se nastavlja sve dok se ne dođe do nekog od listova stabla odluke. *UN* arhitektura zapravo vrši evaluaciju stabla odluke sekvencijalno, čvor po čvor. Ovo je identičan način na koji se stablo evaluira u slučaju softverske implementacije. U slučaju softverske implementacije neophodno je postojanje nekog mikroprocesora da bi se izvršio program koji evaluira stablo odluke. Međutim mikroprocesor predstavlja vrlo neefikasno rešenje sa stanovišta neophodnih hardverskih resursa, jer je on projektovan tako da je u stanju da izvrši proizvoljni program, a ne samo onaj koji evaluira stablo odluke. Za razliku od mikroprocesorskog sistema, *UN* arhitektura je projektovana samo sa jednom namenom, da efikasno evaluira stablo odluke, te je njena složenost višestruko manja od složenosti čak i najjednostavnijeg mikroprocesora. Takođe i brzina klasifikacije *UN* arhitekture bi trebalo da bude veća od brzine klasifikacije sistema baziranog na mikroprocesoru, iz istog razloga.

UN arhitektura takođe zahteva značajno manje resursa u poređenju sa *SMPL* arhitekturom, a pogotovo u poređenju sa arhitekturama predloženim u radovima [4] i [5]. Međutim, kada se poredi brzina klasifikacije, *UN* arhitektura ima značajno duže vreme klasifikacije.

SMPL i *UN* arhitekture predstavljaju dobar primer različitih kompromisa između brzine rada i neophodnih hardverskih resursa. *SMPL* arhitektura optimizovana je za postizanje velike brzine klasifikacije po cenu velike hardverske složenosti. Za razliku od nje, *UN* arhitektura optimizovana je tako da zahteva minimalne resurse, ali je to „plaćeno“ malom brzinom klasifikacije.

Tabela 3.2 sadrži glavne karakteristike razmatranih arhitektura u ovom poglavlju. U koloni „resursi“ prikazani su hardverski resursi neophodni za implementaciju stabla odluke pomoću svake od razmatranih arhitektura. Resursi su izraženi u terminima potrebnog broja čvorova koje je neophodno realizovati kao nezavisne module. Kolona „propusna moć“ sadrži podatke o brzini klasifikacije instanci za svaku arhitekturu, izražene brojem instanci koje se mogu klasifikovati u jednoj sekundi. Na kraju, kolona „inicijalno kašnjenje“ sadrži podatke o vremenu koje je neophodno da bi se klasifikovala prva instanca. Svi podaci izraženi su u terminima:

- broja čvorova u stablu odluke, N_{dt}
- dubini stabla odluke, M
- vremenu potrebnom za evaluaciju testa (2.1), (2.2) ili (2.4), T_{node}
- broju posećenih čvorova prilikom klasifikacije tekuće instance, N_{vn}

Tabela 3.2 Potrebni resursi, propusna moć i inicijalno kašnjenje različitih arhitektura za hardversku implementaciju stabala odluka

Arhitektura	Resursi (broj potrebnih modula)	Propusna moć (instanci u sekundi)	Inicijalno kašnjenje
predložena u [4]	N_{dt}	$1/M \cdot T_{node}$	$M \cdot T_{node}$
predložena u [5]	N_{dt}	$1/T_{node}$	$2 \cdot T_{node}$
SMPL	M	$1/T_{node}$	$M \cdot T_{node}$
UN	1	$1/N_{vn} \cdot T_{node}$	$N_{vn} \cdot T_{node}$

Kada su hardverski resursi u pitanju arhitekture predložene u radovima [4] i [5] očigledno su najlošije, jer zahtevaju implementaciju svakog čvora iz stabla odluke kao nezavisnog modula. Kao što je ranije već rečeno, ovakav pristup je vrlo neefikasan kada je u pitanju hardverska implementacija stabla odluke, imajući u vidu način rada stabla odluke. *SMPL* arhitektura projektovana je uzimajući u obzir serijsku prirodu rada stabla odluke, što rezultuje značajnim smanjenjem neophodnih hardverskih resursa, koji su sada jednaki dubini stabla koje se

realizuje. *UN* arhitektura zahteva realizaciju samo jednog, univerzalnog čvora i predstavlja najbolju arhitekturu ukoliko se kao kriterijum usvoji minimizacija neophodnih resursa.

Što se tiče brzine klasifikacije, mereno brojem instanci koje se mogu klasifikovati u jednoj sekundi, najlošija je arhitektura predložena u radu [4]. Nakon nje sledi *UN* arhitektura, dok *SMPL* arhitektura i arhitektura predložena u radu [5] imaju najveću brzinu klasifikacije.

Ukoliko se analizira inicijalno kašnjenje, najbolja je arhitektura predložena u radu [5]. Ovo je bilo očekivano, jer se pomoću nje bilo koje stablo realizuje pomoću dvoslojne mreže. Ostale tri arhitekture imaju približno ista inicijalna kašnjenja.

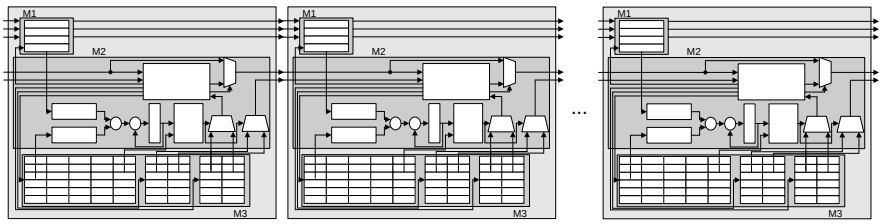
Međutim, bitno je naglasiti da je u većini aplikacija inicijalno kašnjenje najmanje bitan parametar od navedena tri. Imajući u vidu ovu činjenicu, kao najbolja arhitektura, ukoliko želimo postići veliku brzinu klasifikacije, što je vrlo često dominantan uslov, ističe se *SMPL* arhitektura. *UN* arhitektura je očigledno najbolji izbor ukoliko je dominantan kriterijum minimizacija neophodnih hardverskih resursa. Detaljnije poređenje *SMPL* i *UN* arhitektura, kako u pogledu teorijskih performansi tako i u slučaju performansi na konkretnim klasifikacionim problemima biće predstavljeno kasnije.

3.1. Detalji *SMPL* i *UN* arhitektura

U ovoj glavi detaljno su opisane *SMPL* i *UN* arhitekture za hardversku implementaciju stabala odluka. Kroz niz primera takođe će biti ilustrovan način konfigurisanja i korišćenja *SMPL* i *UN* arhitektura. Na početku, biće predstavljene *SMPL* i *UN* arhitekture koje se mogu koristiti za hardversku implementaciju neortogonalnih stabla odluka, kod kojih su testovi definisani u čvorovima stabla opisani jednačinom (2.2). Ovakva stabla se najčešće sreću u praksi. Modifikacije koje je neophodno uraditi na predloženim *SMPL* i *UN* arhitekturama, da bi se mogle koristiti za hardversku implementaciju ortogonalnih i nelinearnih stabala odluke biće izložene nakon toga.

3.1.1. *SMPL* arhitektura

SMPL arhitektura sastoji se iz *M* identičnih modula, koji zapravo čine faze u protočnoj obradi prilikom klasifikacije instance pomoću stabla odluke. Parametar *M* predstavlja dubinu stabla odluke koje se implementira. Osnovi izgled *SMPL* arhitekture prikazan je na slici 3.5.



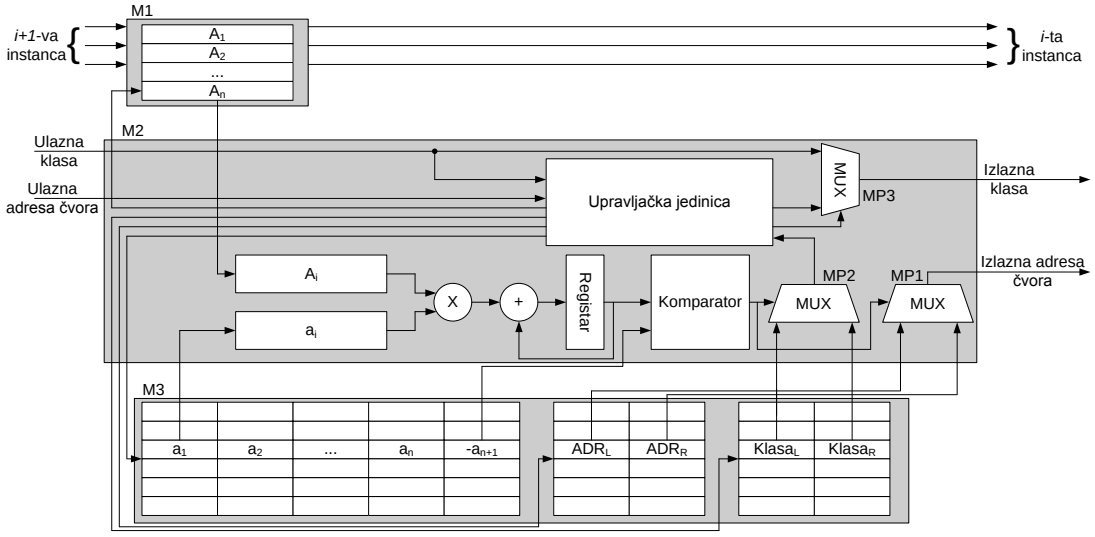
Slika 3.5 Generalni izgled SMPL *arhitekture*

Svakom od nivoa u stablu odluke asociran je po jedan univerzalni čvor. Univerzalni čvor je u stanju da evaluira testove asocirane svim čvorovima koji se nalaze na istom nivou u stablu odluke. Svaki univerzalni čvor čini jednu fazu u protočnoj obradi i sastoji se iz tri glavna modula:

- M1 - memorije za smeštanje vrednosti atributa tekuće instance,
- M2 - modula za evaluiranje čvora ili propuštanje relevantnih informacija iz prethodne faze obrade,
- M3 - memorije za smeštanje informacija o čvorovima iz istog nivoa stabla odluke.

Moduli M1 i M2 iz svih univerzalnih čvorova povezani su u dva lanca, što se može videti na slici 3.5. Ova dva lanca čine protočnu strukturu koja obezbeđuje veliku brzinu klasifikacije instanci.

Na slici 3.6 prikazana je detaljna struktura univerzalnog čvora, odnosno jedne faze u protočnoj obradi.



Slika 3.6 Detaljna struktura univerzalnog čvora

Modul M1 predstavlja memoriju za smeštanje vrednosti atributa tekuće instance koja se obrađuje u univerzalnom čvoru. Njena veličina određena je brojem atributa klasifikacionog problema, *n*, i brojem bitova koji se koriste za predstavljanje vrednosti atributa, N_a. Ovo zapravo nije obična RAM memorija već je reč o ukupno n N_a-bitnih registara čiji su ulazi povezani sa izlazima odgovarajućih registara iz prethodne faze protočne obrade, a njihovi izlazi su povezani sa ulazima odgovarajućih registara iz naredne faze protočne obrade. Na ovaj način formiran je jedan od protočnih lanaca.

Modul M2 je zapravo glavni modul univerzalnog čvora koji može da radi u dva režima. Modul M2 normalno radi u režimu evaluacije odabranog čvora iz stabla odluke. Ovaj režim rada je aktivan uvek kada je vrednost ulaznog signala *„Ulazna klasa“* jednaka nuli. U ovom režimu modul M2 računa poziciju tekuće instance u odnosu na hiperravan asociranu sa odabranim čvorom stabla odluke. Na osnovu ove pozicije donosi se odluka o čvoru iz narednog sloja koji će biti posećen, ili, u slučaju da je naredni čvor zapravo list, odluka o klasi kojoj pripada tekuća instanca. Ove dve informacije prosleđuju se pomoću dva izlazna signala, *„Izlazna klasa“* i *„Izlazna adresa čvora“*. Adresa čvora iz narednog sloja koji će biti posećen prosleđuje se preko izlaznog signala *„Izlazna adresa čvora“*. Ukoliko je naredni čvor koji treba posetiti zapravo list, tada se tekuća instanca može klasifikovati, a njena klasa se prosleđuje narednoj fazi obrade pomoću izlaznog signala *„Izlazna klasa“*. U slučaju da list još nije dostignut, vrednost ovog signala treba da je nula, u protivnom signal *„Izlazna klasa“* treba da ima vrednost binarnog koda koji je asociran odabranoj klasi. U slučaju da je vrednost ulaznog signala *„Ulazna klasa“* različita od nule, modul M2 radi u režimu propuštanja. U ovom režimu vrednost ulaznog signala *„Ulazna klasa“* prosto se prosleđuje na izlazni signal *„Izlazna klasa“*, dok izlazni signal *„Izlazna adresa čvora“* ima proizvoljnu vrednost. Izlazni signali univerzalnog čvora iz *n*-tog sloja, *„Izlazna klasa“* i *„Izlazna adresa čvora“*, povezani su na ulazne signale univerzalnog čvora iz *n+1*-vog sloja, *„Ulazna klasa“* i *„Ulazna adresa čvora“*. Na ovaj način formiran je drugi lanac u protočnoj obradi.

Kada radi u režimu evaluacije, modul M2 računa poziciju tekuće instance u odnosu na odabranu hiperravan sekvencijalno evaluirajući jednačinu (2.2), koristeći jedan množač i jedan sabirač. Registar služi za čuvanje međurezultata prilikom računanja izraza (2.2). Kada se izvrši izračunavanje sume u izrazu (2.2), pomoću komparatora njena vrednost se poredi sa slobodnim članom a_{n+1}. Izlaz komparatora govori o tome sa koje strane hiperravni se nalazi tekuća instanca. Vreme potrebno da se izvrši ova operacija jednako je $T_{node}=n+1$ perioda sinhronizacionog signala.

Podatak o poziciji instance u odnosu na tekuću hiperravan se zatim koristi da se izabere ispravan čvor naslednik u sledećem sloju stabla odluke. Za odabir čvora naslednika koristi se multiplekser MP1 koji bira jednu od dve moguće adrese čvorova naslednika tekućeg čvora. Ova adresa se zatim prosleđuje sledećem univerzalnom čvoru preko izlaznog signala *„Izlazna adresa čvora“*. Drugi multiplekser MP2, koristi se za odabir potencijalne klase kojoj tekuća instanca pripada. I ovaj multiplekser ima dva ulaza, jer svaki čvor ima tačno dva naslednika i svakom od njih je pridružena informacija o klasi kojoj pripada. Kao što je ranije rečeno, samo listovi imaju sebi asocirane klase. Svim unutrašnjim čvorovima asocirana je klasa „0“ kao indikacija da se ne radi o listovima i da proces klasifikacije još nije završen. Izlaz multipleksera MP2 vodi se u upravljačku jedinicu univerzalnog čvora. Treći multiplekser, MP3, koristi se za ispravno generisanje izlaznog signala *„Izlazna klasa“*. Ovaj multiplekser je pod kontrolom upravljačke jedinice. Ukoliko je vrednost ulaznog signala *„Ulazna klasa“* različita od nule, to je znak da je tekuća instanca već klasifikovana i upravljačka jedinica u ovom slučaju konfiguriše multiplekser MP3 tako da on propušta ulazni signal *„Ulazna klasa“* na izlazni signal *„Izlazna klasa“*. U slučaju da je vrednost ulaznog signala *„Ulazna klasa“* jednaka nuli, upravljačka jedinica konfiguriše multiplekser MP3 tako da on na izlazni signal *„Izlazna klasa“* propušta izlaz iz multipleksera MP2.

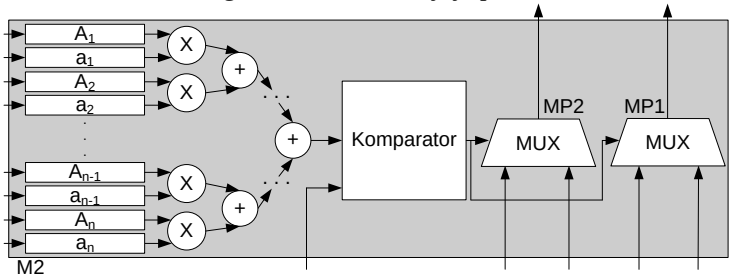
Modul M3 skladišti informacije o čvorovima koji pripadaju nivou u stablu odluke kojem je posmatrani univerzalni čvor asociran. Ovaj modul sastoji se iz tri RAM memorije. Prva memorija ima ukupno $(n+1)\cdot N_{dt}(l)$ lokacija, pri čemu je svaka od njih veličine N_c bita. $N_{dt}(l)$ predstavlja broj čvorova koji pripadaju l-tom nivou stabla odluke. Ova memorija služi za čuvanje vrednosti koeficijenata hiperravni asociranih posmatranim čvorovima. Memorija je organizovana u $N_{dt}(l)$ blokova, pri čemu svaki sadrži ukupno $(n+1)$ lokacija. Svaki blok zapravo skladišti koeficijente koji definišu jednu hiperravan. Bitno je napomenuti da se vrednost slobodnog člana iz jednačine (2.2) smešta sa promenjenim znakom, kako je naznačeno na slici 3.6. U drugoj memoriji nalaze se adrese čvorova naslednika za svaki čvor iz l-tog sloja stabla odluke. Veličina ove memorije iznosi $2\cdot N_{dt}(l)$ lokacija, pri čemu je svaka veličine $\lceil ld(N_{dt})\rceil$ bita. U slučaju da je neki od naslednika zapravo list, njegova adresa može biti postavljena na proizvoljnu vrednost. Treća memorija koristi se za smeštanje informacija o klasama koje su asocirane naslednicima čvorova iz l-tog nivoa stabla odluke. Veličina ove memorije je takođe $2\cdot N_{dt}(l)$ lokacija, pri čemu je svaka od njih veličine $\lceil ld(N_{cl})\rceil$ bita, gde N_{cl} predstavlja broj različitih klasa u koje se mogu klasifikovati instance posmatranog problema. Ukoliko naslednik čvora nije list, tada se kao njemu asocirana klasa mora koristiti klasa nula, koja označava da tekuća instanca još nije klasifikovana.

Propusna moć predložene *SMPL* arhitekture obrnuto je srazmerna vremenu potrebnom da se odredi pozicija instance u odnosu na hiperravan. Obzirom da se u ovom slučaju pozicija određuje serijskom evaluacijom izraza (2.2) vreme klasifikacije iznosi

$$Propusna\_moć = \frac{1}{T_{node}\cdot CP} = \frac{1}{(n+1)\cdot CP} \tag{3.1}$$

Dobijena vrednost u skladu je sa izrazom za propusnu moć *SMPL* arhitekture, prikazanim u tabeli 3.2, sa tom razlikom da smo sada u mogućnosti da precizno odredimo vreme potrebno da se tekuća instanca obradi u jednom čvoru stabla odluke, T_{node}. Kao što se iz izraza (2.2) može videti, u slučaju *SMPL* arhitekture propusna moć ne zavisi od veličine stabla odluke, konkretno od broja čvorova i od dubine stabla. Ovo je zbog toga što je *SMPL* arhitektura bazirana na protočnoj obradi podataka i zbog toga što takođe koristi činjenicu da za klasifikaciju instance pomoću stabla odluke nije potrebno evaluirati sve čvorove stabla. O ovome je već bilo reči ranije.

Propusna moć *SMPL* arhitekture može se dodatno skratiti, na taj način što će se pozicija instance u odnosu na hiperravan određivati paralelnom evaluacijom testa (2.2). Umesto da se suma u izrazu (2.2) izračunava serijski, član po član, ona se može evaluirati u paraleli korišćenjem n množača i mreže sabirača. Ovaj pristup zahteva modifikaciju modula M2 unutar univerzalnog čvora na način koji je prikazan na slici 3.7.

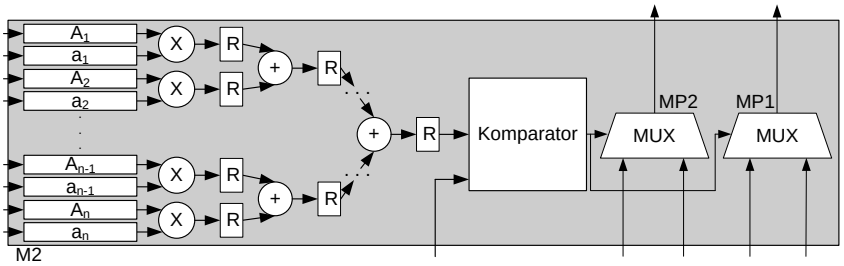


Slika 3.7 Modifikovana struktura M2 modula koja omogućava paralelnu evaluaciju

Modifikovani M2 modul koristi n množača da u paraleli izračuna vrednosti svih n članova sume u izrazu (2.2). Nakon toga, koristeći mrežu sabirača organizovanih u strukturu stabla vrši se sabiranje sabiraka. Potreban broj sabirača da bi se formirala ova mreža iznosi $n-1$. Korišćenjem modifikovanog M2 modula vreme evaluacije testa u svakom od čvorova stabla sada iznosi samo jednu periodu globalnog sinhronizacionog signala, što predstavlja ubrzanje od $n+1$ puta u odnosu na slučaj kada se evaluacija vrši serijskim putem.

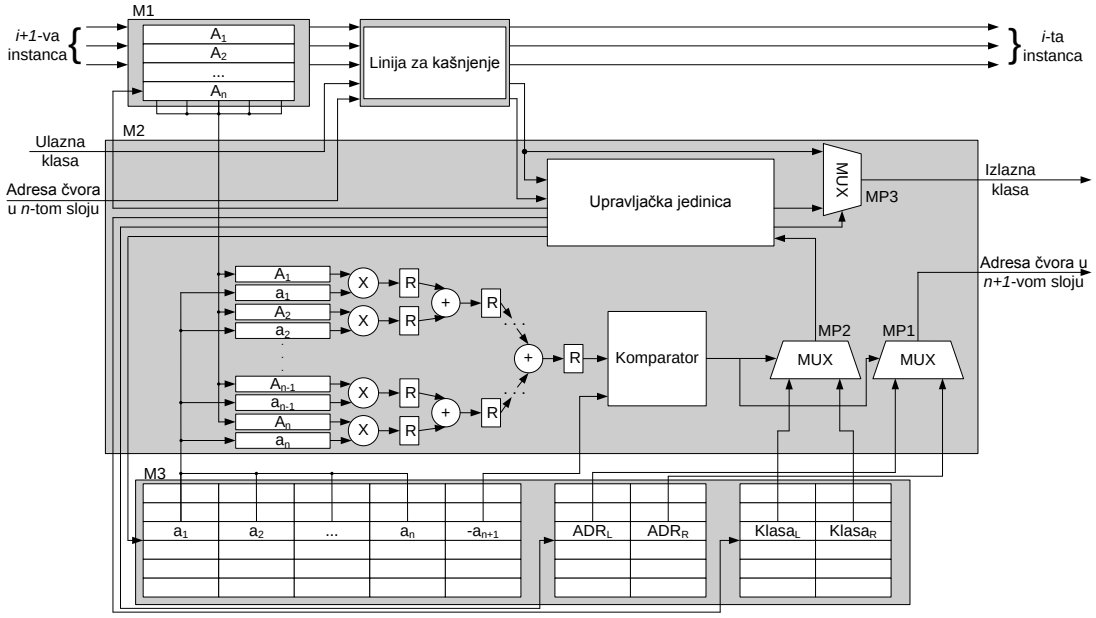
$$T_{node} = 1 \tag{3.2}$$

Međutim, treba imati u vidu da će stvarno ubrzanje biti značajno manje. Razlog za to je što modifikovani M2 modul ne može raditi na tako visokoj učestanosti kao osnovni M2 modul. Ovo je zbog toga što je kritična putanja signala u slučaju modifikovanog M2 modula značajno duža od kritične putanje osnovnog M2 modula. U slučaju osnovnog M2 modula, kritična putanja predstavlja vreme propagacije signala kroz jedan množač i jedan sabirač. Kritična putanja u slučaju modifikovanog M2 modula predstavlja vreme propagacije kroz jedan množač i kroz mrežu sabirača koja ima $\lceil ld(n)\rceil$ nivoa, što je značajno duži put. Da bi se postiglo ubrzanje od $n+1$ puta, modifikovani M2 modul sa slike 3.7 potrebno je dodatno modifikovati ubacivanjem registara u kritičnu putanju. Na ovaj način se kritična putanja kroz sistem smanjuje i samim tim se povećava učestanost na kojoj M2 modul može da radi. Ovo ubacivanje registara zapravo predstavlja uvođenje protočne obrade u M2 modul. Struktura modifikovanog M2 modula sa protočnom obradom prikazana je na slici 3.8.



Slika 3.8 Modifikovana struktura M2 modula sa protočnom obradom podataka

Uvođenje protočne obrade u M2 modul povlači za sobom potrebu da se modifikuje i struktura čitavog univerzalnog čvora, prikazana na slici 3.6. Modifikacija je neophodna da bi se očuvao sinhronizam između instanci i ulazno/izlaznih signala univerzalnog čvora. Brzina protoka instanci kroz univerzalni čvor je u slučaju originalne arhitekture sa slike 3.6 iznosila jedan instanca na svakih *n+1* taktova. Međutim, u slučaju korišćenja modifikovanog M2 modula sa protočnom obradom, ova brzina se povećava na jednu instancu po taktu, sa inicijalnim kašnjenjem od *n+1* taktova. Isto ovo kašnjenje mora se uvesti i za instance koje prolaze kroz univerzalni čvor i za ulazne signale *„Ulazna klasa“* i *„Ulazna adresa čvora“* da bi se očuvao sinhronizam u čitavom sistemu. Ovo zahteva uvođenje linije za kašnjenje u arhitekturu univerzalnog čvora sa slike 3.8. Takođe je neophodno modifikovati modul M1 i deo modula M3 da bi se mogli povezati sa modifikovanim M2 modulom. Modul M1 mora biti u stanju da istovremeno dostavi vrednosti svih atributa modulu M2. Isto važi i za memoriju za smeštanje koeficijenata hiperravni unutar modula M3. I ona mora biti u stanju da istovremeno dostavi vrednosti svih koeficijenata odabrane hiperravni modulu M2. Ovo je neophodno zbog toga što se unutar modifikovanog M2 modula u istom trenutku, odnosno u paraleli, izračunavaju vrednosti svih članova sume iz izraza (2.2). Nova arhitektura univerzalnog čvora sa svim pomenutim modifikacijama prikazana je na slici 3.9.



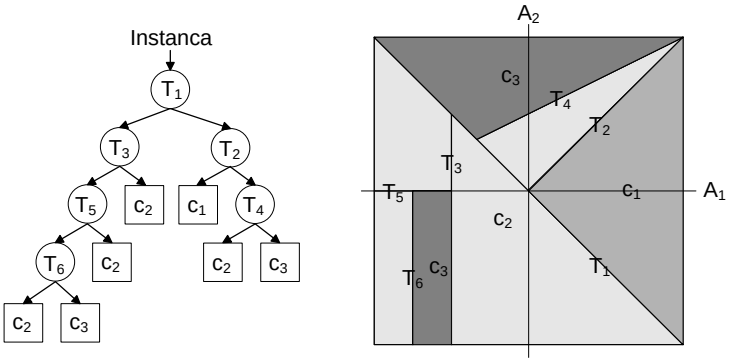
Slika 3.9 Modifikovana struktura univerzalnog čvora

SMPL arhitektura bazirana na modifikovanom univerzalnom čvoru sa slike 3.9 zapravo koristi dvostruku protočnu obradu. Jedna protočna obrada je na nivou univerzalnih čvorova, a druga je na nivou evaluacije testova unutar univerzalnih čvorova. Kombinovanjem ove dve protočne obrade može se postići propusna moć određena sledećim izrazom.

$$Propusna\_moć = \frac{1}{T_{node} \cdot CP} = \frac{1}{CP} \tag{3.3}$$

3.1.2. Primer korišćenja *SMPL* arhitekture

Način konfigurisanja i rada predložene *SMPL* arhitekture za hardversku implementaciju stabala odluka najbolje će biti ilustrovan na jednom konkretnom primeru. Na slici 3.10 prikazano je jedno stablo odluke kao i particija prostora atributa koja se dobija pomoću njega. Klasifikacioni problem koji se rešava opisan je pomoću dva atributa, A_1 i A_2, a svaka instanca se može klasifikovati u jednu od tri klase, c_1, c_2, c_3.



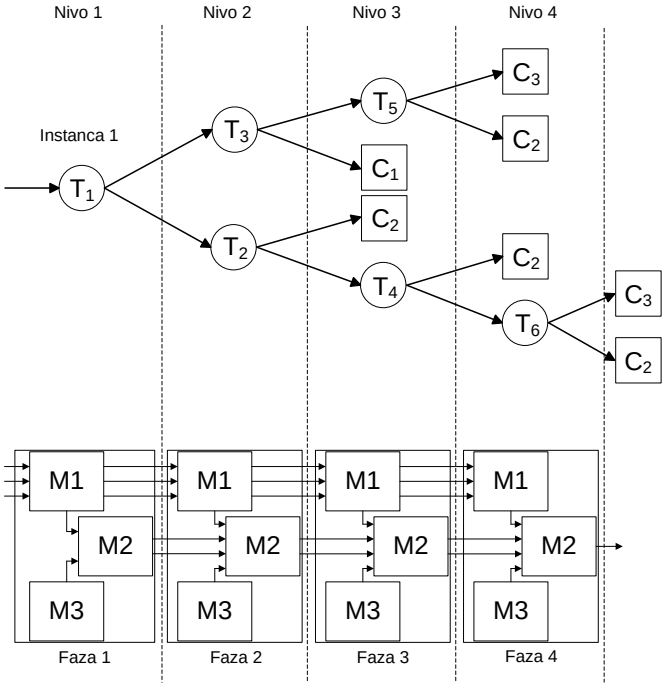
Slika 3.10 Stablo odluke i klasifikacija prostora atributa pomoću njega

Kao što se sa slike 3.10 može videti, stablo odluke ima ukupno šest čvorova i sedam listova koji su organizovani u četiri nivoa. Dvodimenzionalni prostor atributa podeljen je na ukupno četiri regiona pomoću šest pravih koje odgovaraju sledećim testovima u šest čvorova.

$$
\begin{aligned}
&T_1 : 1 \cdot A_1 + 1 \cdot A_2 + 0 = 0 \\
&T_2 : -1 \cdot A_1 + 1 \cdot A_2 + 0 = 0 \\
&T_3 : 1 \cdot A_1 + 0 \cdot A_2 + 0.5 = 0 \\
&T_4 : -0.5 \cdot A_1 + 1 \cdot A_2 - 0.5 = 0 \\
&T_5 : 0 \cdot A_1 + 1 \cdot A_2 + 0 = 0 \\
&T_6 : 1 \cdot A_1 + 0 \cdot A_2 + 0.75 = 0
\end{aligned}
\qquad (3.4)
$$

Na slici 3.10 debljim linijama prikazani su segmenti ovih pravih koji su od interesa prilikom klasifikacije.

Ukoliko se za realizaciju stabla odluke sa slike 3.10 koristi *SMPL* arhitektura, biće nam potrebna ukupno četiri univerzalna čvora, odnosno imaćemo četiri faze protočne obrade.



Slika 3.11 Izgled hardverske implementacije stabla odluke sa slike 3.10 bazirane na SMPL *arhitekturi*

Univerzalni čvor u fazi 1 realizuje samo koren stabla odluke, odnosno čvor sa testom T_1. Univerzalni čvor u fazi 2 realizuje dva čvora iz drugog nivoa, odnosno testove T_2 i T_3. Univerzalni čvor u fazi 3 realizuje čvorove sa

testovima T_4 i T_5, i na kraju, univerzalni čvor u fazi 4 realizuje čvor sa testom T_6. Sadržaj memorija iz M3 modula unutar svakog univerzalnog čvora u slučaju realizacije stabla odluke sa slike 3.10 prikazan je u tabeli 3.3.

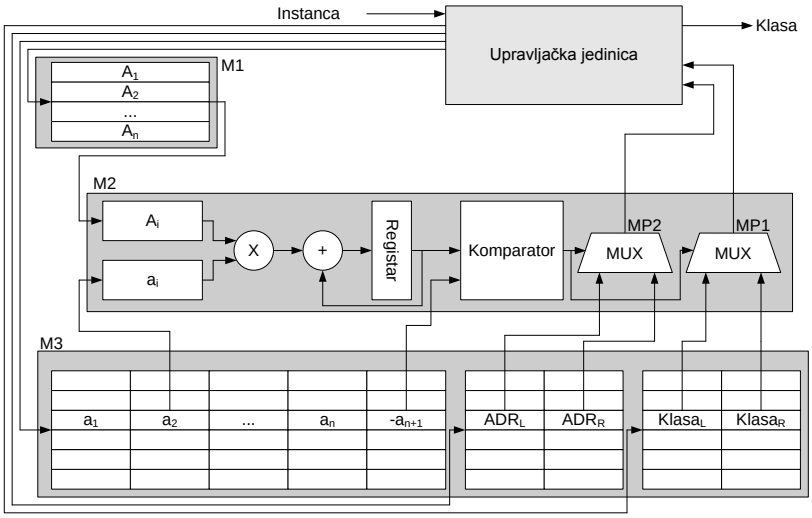
Tabela 3.3 Sadržaj memorija iz M3 modula u slučaju realizacije stabla odluke sa slike 3.10

Faza protočne obrade	Memorija koeficijenata hiperravni			Memorija adresa čvorova naslednika	Memorija klasa asociranih naslednicima
Faza 1	*1*	*1*	*0*	*0 & 1*	*0 & 0*
Faza 2	*-1*	*1*	*0*	*0 & X*	*0 & c_2*
	1	*0*	*-0.5*	*X & 1*	*c_1 & 0*
Faza 3	*-0.5*	*1*	*0.5*	*0 & X*	*0 & c_2*
	0	*1*	*0*	*X & X*	*c_2 & c_3*
Faza 4	*1*	*0*	*-0.75*	*X & X*	*c_2 & c_3*

Sadržaj memorije koeficijenata hiperravni odgovara koeficijentima pomoću kojih je definisana svaka od ukupno šest hiperravni iz (3.4), uz jedan izuzetak. Kao što je ranije već naglašeno, slobodan član je potrebno smestiti sa promenjenim predznakom. Ovo pravilo korišćeno je prilikom popunjavanja tabele 3.3. Simbol '*&*' korišćen prilikom popunjavanja tabele 3.3 predstavlja operator konkatenacije, dok simbol '*X*' predstavlja oznaku za proizvoljnu vrednost (0 ili 1).

3.1.3. *UN* arhitektura

Kao što je ranije već rečeno, *UN* arhitektura sastoji se samo od jednog univerzalnog čvora koji se koristi za evaluaciju svih testova koji postoje u stablu odluke. Struktura *UN* arhitekture prikazana je na slici 3.12.



Slika 3.12 Detaljan prikaz UN *arhitekture*

Struktura *UN* arhitekture vrlo je slična strukturi univerzalnog čvora iz *SMPL* arhitekture, što se moglo i očekivati.

Kao i u slučaju *SMPL* arhitekture, modul M1 predstavlja memoriju za smeštanje vrednosti atributa tekuće instance koja se obrađuje u univerzalnom čvoru. Njegova veličina određena je brojem atributa klasifikacionog problema, *n*, i brojem bita koji se koriste za predstavljanje vrednosti atributa, N_a. Za razliku od M1 modula unutar univerzalnog čvora u *SMPL* arhitekturi, u ovom slučaju reč je o običnoj RAM memoriji.

Za razliku od *SMPL* arhitekture, modul M2 ovaj put može da radi samo u režimu evaluacije testova. U ovom režimu modul M2 računa poziciju tekuće instance u odnosu na hiperravan asociranu odabranom čvoru stabla odluke. Na osnovu ove pozicije donosi se odluka o čvoru iz narednog sloja koji će biti posećen, ili, u slučaju da je naredni čvor zapravo list, odluka o klasi kojoj pripada tekuća instanca.

Modul M2 računa poziciju tekuće instance u odnosu na odabranu hiperravan sekvencijalno evaluirajući jednačinu (2.2), koristeći jedan množač i jedan sabirač. Registar služi za čuvanje međurezultata prilikom računanja izraza (2.2). Kada se izvrši izračunavanje sume u izrazu (2.2), pomoću komparatora njena vrednost se poredi sa slobodnim članom a_{n+1}. Izlaz komparatora govori o tome sa koje strane hiperravni se nalazi tekuća instanca. Vreme potrebno da se izvrši ova operacija jednako je $T_{node}=n+1$ perioda globalnog sinhronizacionog signala.

Podatak o poziciji instance u odnosu na hiperravan se zatim koristi da se izabere ispravan čvor naslednik u sledećem sloju stabla odluke. Za odabir čvora naslednika koristi se multiplekser MP2 koji bira jednu od dve

moguće adrese čvorova naslednika tekućeg čvora. Ova adresa se zatim prosleđuje upravljačkoj jedinici. Multiplekser MP1, koristi se za odabir klase kojoj tekuća instanca pripada. I ovaj multiplekser ima dva ulaza, jer svaki čvor ima tačno dva naslednika i svakom od njih je pridružena informacija o klasi kojoj pripada. Kao što je ranije naglašeno, samo listovi imaju sebi asocirane klase. Izlaz MP1 multipleksera takođe se vodi u upravljačku jedinicu.

Modul M3 skladišti informacije o svim čvorovima stabla odluke. Kao i kod *SMPL* arhitekture i ovaj modul sastoji se od tri RAM memorije. Prva memorija ima ukupno $(n+1)\cdot N_{dt}$ lokacija, pri čemu je svaka od njih veličine N_c bita, a N_{dt} predstavlja ukupan broj čvorova stabla odluke. Ova memorija služi za čuvanje vrednosti koeficijenata hiperravni asociranih čvorovima u stablu. Memorija je organizovana u N_{dt} blokova, pri čemu svaki sadrži ukupno $(n+1)$ lokacija. Svaki blok zapravo skladišti koeficijente koji definišu jednu hiperravan. Kao i u slučaju *SMPL* arhitekture, vrednost slobodnog člana iz jednačine (2.2) čuva se sa promenjenim znakom, kako je naznačeno na slici 3.12. U drugoj memoriji nalaze se adrese čvorova naslednika za svaki čvor iz stabla odluke. Veličina ove memorije iznosi $2\cdot N_{dt}$ lokacija, veličine $\lceil ld(N_{dt}) \rceil$ bita. Način na koji se formiraju adrese naslednika drugačiji je u odnosu na način opisan kod *SMPL* arhitekture. U slučaju da je neki od naslednika zapravo list, njegova adresa mora biti postavljena na nulu, za razliku od *SMPL* arhitekture gde je mogla imati proizvoljnu vrednost. Adresa nula je zapravo adresa korena stabla. Pošto je tekuća instanca stigla do jednog od listova stabla može biti klasifikovana, a sistem treba da se vrati u koren stabla da bi mogao da počne sa klasifikacijom sledeće instance. Ovo je razlog zašto se kao adresa naslednika u slučaju listova mora koristiti adresa nula. Treća memorija koristi se za smeštanje informacija o klasama koje su asocirane naslednicima čvorova iz stabla odluke. Veličina ove memorije je takođe $2\cdot N_{dt}$ lokacija, pri čemu je svaka od njih veličine $\lceil ld(N_{cl}) \rceil$ bita, gde N_{cl} predstavlja broj različitih klasa u koje se mogu klasifikovati instance posmatranog problema. Opet postoji razlika u načinu kako se popunjavaju one lokacije koje su asocirane čvorovima u odnosu na način opisan kod *SMPL* arhitekture. Ukoliko naslednik čvora nije list, tada se kao njemu asocirana klasa može koristiti bilo koja klasa, za razliku od *SMPL* arhitekture gde se morala koristiti klasa nula. Do ovih razlika dolazi zbog toga što se u slučaju *UN* arhitekture koriste drugačiji mehanizmi signalizacije da je tekuća instanca stigla do lista i da može biti klasifikovana. U slučaju *SMPL* arhitekture ova signalizacija išla je preko podataka o klasama koje su asocirane čvorovima, dok se u slučaju *UN* arhitekture koristi podatak o adresi narednog čvora koji treba posetiti da bi se odredilo da li smo stigli do lista ili ne.

Vreme klasifikacije *UN* arhitekture jednako je vremenu potrebnom da se odredi pozicija instance u odnosu na hiperravan, T_{node}, pomnoženom sa brojem čvorova koji su bili evaluirani pre nego što je neki od listova dostignut, N_{vn}. Obzirom da se u ovom slučaju pozicija određuje serijskom evaluacijom izraza (2.2) vreme klasifikacije iznosi

$$Vreme\_klasifikacije = N_{vn}\cdot T_{node} = N_{vn}\cdot(n+1) \tag{3.5}$$

Dobijena vrednost u skladu je sa izrazom za propusnu moć *UN* arhitekture, prikazanim u tabeli 3.2, sa tom razlikom da smo sada u mogućnosti da precizno odredimo vreme potrebno da se tekuća instanca obradi u jednom čvoru stabla odluke, T_{node}. Kao što se iz izraza (3.5) može videti, u slučaju *UN* arhitekture vreme klasifikacije instance zavisi od putanje koja je pređena, i ono nije isto za sve instance. Ovo je bitna razlika u odnosu na *SMPL* arhitekturu, kod koje je vreme klasifikacije isto za sve instance. Razlog za ovakav rezultat leži u činjenici da se stablo odluke unutar *UN* arhitekture evaluira sekvencijalno, čvor po čvor, na isti način kao i u slučaju softverske implementacije.

Vreme klasifikacije *UN* arhitekture može se skratiti, slično kao i u slučaju *SMPL* arhitekture, na taj način što će se pozicija instance u odnosu na hiperravan određivati paralelnom evaluacijom testa (2.2). Umesto da se suma u izrazu (2.2) izračunava serijski, član po član, one se mogu evaluirati u paraleli korišćenjem n množača i mreže sabirača. Ovaj pristup zahteva modifikaciju modula M2 unutar univerzalnog čvora na isti način kao i u slučaju *SMPL* arhitekture koji je bio prikazan na slici 3.7.

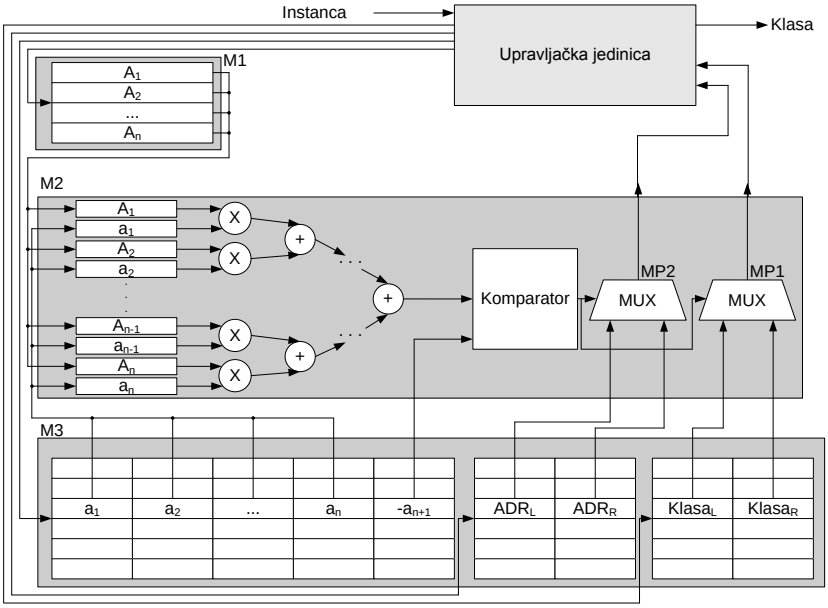
Korišćenjem modifikovanog M2 modula vreme evaluacije testa u svakom od čvorova stabla sada iznosi samo jednu periodu globalnog sinhronizacionog signala, što predstavlja ubrzanje od $n+1$ puta u odnosu na slučaj kada se evaluacija vrši serijskim putem.

$$T_{node} = 1 \tag{3.6}$$

Kao i u slučaju *SMPL* arhitekture, treba imati u vidu da će stvarno ubrzanje biti značajno manje. Kao što je već rečeno ovo je zbog toga što je kritičan put signala u slučaju modifikovanog M2 modula značajno duži od kritičnog puta osnovnog M2 modula. Međutim u slučaju *UN* arhitekture, nije moguće izvršiti skraćivanje kritičnog puta u modifikovanom M2 modulu ubacivanjem registara iza množača i nakon svakog nivoa sabirača. Razlog zbog čega ovo nije moguće jeste što se ovaj put time ne bi ništa dobilo, jer se instance u *UN* arhitekturi obrađuju sekvencijalno i nije moguće koristiti tehnike protočne obrade podataka.

Struktura *UN* arhitekture koja je bazirana na modifikovanom M2 modulu prikazana je na slici 3.13. Kao i u slučaju *SMPL* arhitekture, opet je neophodno modifikovati modul M1 i deo modula M3 da bi se mogli povezati sa modifikovanim M2 modulom. Modul M1 mora biti u stanju da istovremeno dostavi vrednosti svih atributa modulu M2. Isto važi i za memoriju za smeštanje koeficijenata hiperravni unutar modula M3. I ova memorija mora biti u stanju da istovremeno dostavi vrednosti svih koeficijenata odabrane hiperravni modulu M2. Ovo je neophodno

zbog toga što se unutar modifikovanog M2 modula u istom trenutku, odnosno u paraleli, izračunavaju vrednosti svih članova sume iz izraza (2.2). Za razliku od *SMPL* arhitekture, u slučaju *UN* arhitekture nije potrebno uvođenje linije za kašnjenje radi očuvanja sinhronizma, jer u ovom slučaju ne postoji protočna obrada instanci.



Slika 3.13 UN *arhitektura sa paralelnom evaluacijom testova u čvorovima stabla odluke*

Vreme klasifikacije modifikovane *UN* arhitekture jednako je vremenu potrebnom da se odredi pozicija instance u odnosu na hiperravan, T_{node}, pomnoženom sa brojem čvorova koji su bili evaluirani pre nego što je neki od listova dostignut, N_{vn}. Obzirom da se u ovom slučaju pozicija određuje paralelnom evaluacijom izraza (2.2) vreme klasifikacije iznosi

$$Vreme\_klasifikacije = N_{vn} \cdot 1 = N_{vn} \tag{3.7}$$

Još jednom treba naglasiti da stvarno ubrzanje *UN* arhitekture sa paralelnom evaluacijom testova u odnosu na *UN* arhitekturu sa serijskom evaluacijom testova neće biti *n+1* puta. Razlog za ovo je što će maksimalna učestanost rada *UN* arhitekture sa paralelnom evaluacijom testova biti manja u odnosu na *UN* arhitekturu sa serijskom evaluacijom zbog dužeg kritičnog puta signala.

3.1.4. Primer korišćenja *UN* arhitekture

Način konfigurisanja *UN* arhitekture za hardversku implementaciju stabala odluka biće ilustrovan na istom primeru stabla odluke sa slike 3.10 koje je bilo korišćeno za ilustraciju rada *SMPL* arhitekture, Sam način rada *UN* arhitekture neće biti detaljno analiziran.

Klasifikacioni problem koji se rešava opisan je pomoću dva atributa, A_1 i A_2, a svaka instanca se može klasifikovati u jednu od tri klase, c_1, c_2, c_3. Stablo odluke ima ukupno šest čvorova i sedam listova koji su organizovani u četiri nivoa. Dvodimenzionalni prostor atributa podeljen je na ukupno četiri regiona pomoću šest pravih koje odgovaraju testovima definisanim izrazima (3.4).

Ukoliko se za realizaciju stabla odluke sa slike 3.10 koristi *UN* arhitektura, biće nam potreban jedan univerzalni čvor koji će se koristiti za evaluaciju testova definisanih u svim čvorovima unutar stabla odluke.

Pre korišćenja *UN* arhitekture neophodno je konfigurisati memorije unutar modula M3 na način da sadrže strukturne podatke o stablu sa slike 3.10. Sadržaj memorija iz M3 modula prikazan je u tabeli 3.4.

Tabela 3.4 Sadržaj memorija iz M3 modula u slučaju realizacije stabla odluke sa slike 3.10

	Memorija koeficijenata hiperravni		
Adresa	**A_1**	**A_2**	**A_3**
0	*1*	*1*	*0*
3	*-1*	*1*	*0*
6	*1*	*0*	*-0.5*
9	*-0.5*	*1*	*0.5*
12	*0*	*1*	*0*
15	*1*	*0*	*-0.75*

Memorija adresa čvorova naslednika

Adresa	Vrednost
0	*1 & 2*
1	*3 & 0*
2	*0 & 4*
3	*5 & 0*
4	*0 & 0*
5	*0 & 0*

Memorija klasa asociranih naslednicima

Adresa	Vrednost
0	*X & X*
1	*X & c_2*
2	*c_1 & X*
3	*X & c_2*
4	*c_2 & c_3*
5	*c_2 & c_3*

Sadržaj memorije koeficijenata hiperravni odgovara koeficijentima pomoću kojih je definisana svaka od ukupno šest hiperravni iz (3.4), vodeći računa da je slobodan član potrebno smestiti sa promenjenim predznakom. Ovo pravilo korišćeno je prilikom popunjavanja tabela 3.4. Simbol '&' korišćen prilikom popunjavanja tabela 3.4 predstavlja operator konkatenacije, dok simbol '*X*' predstavlja oznaku za proizvoljnu vrednost (0 ili 1).

3.2. *SMPL* i *UN* arhitekture za realizaciju ortogonalnih i nelinearnih stabala odluka

SMPL i *UN* arhitekture koje su do sada bile predstavljene mogu se koristiti za hardversku implementaciju neortogonalnih stabala odluka, odnosno stabala odluka kod kojih su svi testovi oblika (2.2). Međutim, u praksi se često koriste i ortogonalna stabla odluka, kod kojih su testovi opisani pomoću izraza (2.1), a takođe se koriste i nelinearna stabla odluka, kod kojih su testovi najčešće oblika (2.4). Predložene *SMPL* i *UN* arhitekture mogu se modifikovati tako da se mogu koristiti i za implementaciju ove dve vrste stabala odluka.

3.2.1. Hardverska implementacija ortogonalnih stabala odluka

Kod ortogonalnih stabala odluka, svi testovi se mogu opisati pomoću izraza (2.1). Izraz (2.1) je zapravo specijalni slučaj opšteg izraza (2.2), tako da se ortogonalna stabla odluka mogu implementirati korišćenjem predloženih *SMPL* i *UN* arhitektura. Jedina stvar o kojoj treba voditi računa jeste način na koji je potrebno inicijalizovati memoriju za smeštanje koeficijenata hiperravni unutar modula M3. U izrazu (2.1) koristi se samo jedan od ukupno *n* atributa i on se poredi sa slobodnim članom. Ova situacija odgovara hiperravni kod koje su svi koeficijenti koji množe atribute jednak nuli, osim jednog, koji ima vrednost jedan i množi atribut koji se pojavljuje u testu. Slobodni član ima vrednost koja je specificirana u testu.

$$\begin{gathered} A_i > a_i \\ \Leftrightarrow \\ 0 \cdot A_1 + 0 \cdot A_2 + ... + 1 \cdot A_i + ... 0 \cdot A_n - a_i > 0 \end{gathered} \tag{3.8}$$

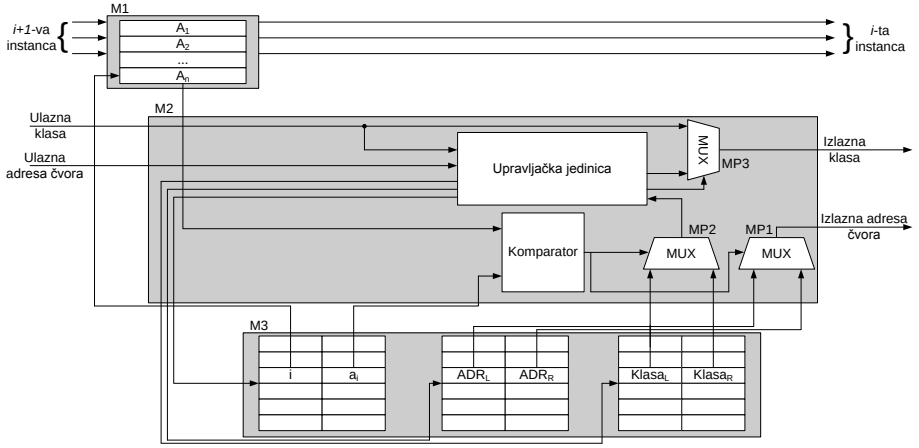
Iako moguća, ovakva realizacija bi bila izuzetno neefikasna. Kao prvo memorija za smeštanje koeficijenata hiperravni bila bi uglavnom popunjena nulama i nepotrebno velika. Kao drugo, testovi (2.1) bi se evaluirali koristeći strukturu koja je projektovana za evaluiranje testova oblika (2.2), što bi rezultuvalo u velikom broju množenja sa nulom i sabiranja sa nulom. U slučaju evaluacije testova oblika (2.1) izvođenje operacija množenja i sabiranja je potpuno nepotrebno i dovodi do utroška vremena u slučaju serijske evaluacije, odnosno trošenja hardverskih resursa u slučaju paralelne evaluacije. Sve ovo upućuje na zaključak da se u slučaju realizacije ortogonalnih stabala odluka postojeće *SMPL* i *UN* arhitekture mogu značajno uprostiti.

Slika 3.14 prikazuje strukturu univerzalnog čvora iz *SMPL* arhitekture koji je optimizovan za rad sa ortogonalnim stablima odluke. Sa slike se može videti da su moduli M2 i M3 pretrpeli značajne izmene.

Modul M2 je uprošćen i iz njega su izbačeni množač, sabirač i registar za čuvanje međurezultata. Zadržan je samo komparator jer je on dovoljan za izvođenje testa (2.1).

Memorija za smeštanje koeficijenata hiperravni unutar modula M3 je takođe znatno uprošćena. Za svaku ortogonalnu hiperravan dovoljno je čuvati samo dva podatka: indeks atributa koji učestvuje u testu i vrednost sa

kojom se dotični atribut poredi. Indeks atributa koji učestvuje u testu koristi se kao adresa u M1 modulu da bi se vrednost željenog atributa dovela na jedan od ulaza komparatora. Na drugi ulaz komparatora dovodi se vrednost sa kojom treba porediti odabrani atribut.



Slika 3.14 Struktura univerzalnog čvora u slučaju realizacije ortogonalnih stabala odluka

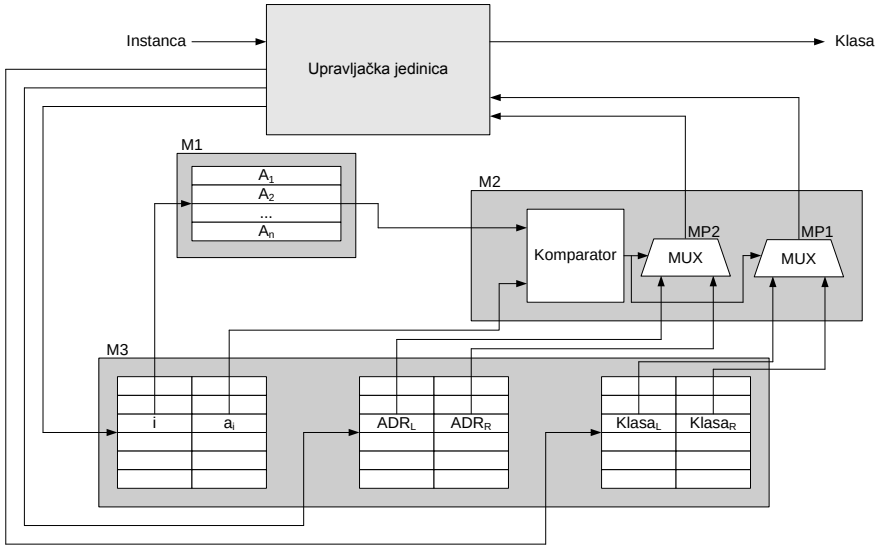
Postupak evaluacije testa (2.1) traje samo jedan takt globalnog sinhronizacionog signala, znatno kraće nego u slučaju evaluacije testova oblika (2.2). Takođe, u ovom slučaju nema mogućnosti za dodatnom protočnom obradom prilikom evaluacije testa. Imajući sve ovo u vidu, *SMPL* arhitektura prilagođena za realizaciju ortogonalnih stabala odluka u stanju je da klasifikuje instance brzinom od jedne instance po taktu globalnog sinhronizacionog signala, sa početnim kašnjenjem koje iznosi *M* taktova.

$$\begin{aligned} &Vreme\_klasifikacije \;=\; T_{node} = 1, \\ &Početno\_kašnjenje = M \end{aligned} \tag{3.9}$$

UN arhitektura se mora prilagoditi na isti način da bi se mogla koristiti za efikasnu realizaciju ortogonalnih stabala odluka. Izgled modifikovane *UN* arhitekture prikazan je na slici 3.15.

Vreme klasifikacije instance korišćenjem ortogonalnog stabla odluke realizovanog pomoću modifikovane *UN* arhitekture iznosi N_{nv} taktova, jer se test u svakom od čvorova evaluira u jednom taktu globalnog sinhronizacionog signala.

$$Vreme\_klasifikacije \;=\; N_{nv} \cdot T_{node} = N_{nv} \tag{3.10}$$



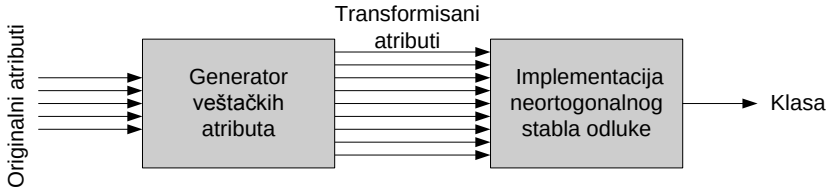
Slika 3.15 UN *arhitektura za realizaciju ortogonalnih stabala odluka*

3.2.2. Hardverska implementacija nelinearnih stabala odluka

U slučaju korišćenja nelinearnih stabala odluka kod kojih se testovi mogu opisati pomoću izraza (2.4), bitno je primetiti da se i oni mogu svesti na izraz (2.2) uvođenjem novih, veštački generisanih atributa. Ukoliko se svaki

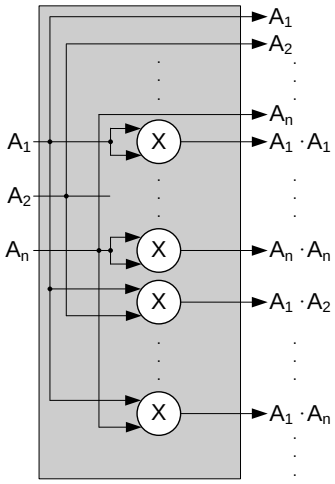
proizvod $A_i^{m_i} \cdot A_j^{m_j} \cdot A_k^{m_k}$, $m_i+m_j+m_k \leq 3$, zameni novim, veštačkim atributom, izraz (2.4) svodi se na izraz (2.2), pri čemu se broj atributa problema povećao. Ovo zapravo znači da se nelinearna stabla u originalnom prostoru atributa kod kojih se testovi mogu opisati izrazom (2.4) mogu posmatrati kao linearna (neortogonalna) stabla odluka u nekom novom prostoru atributa čija je dimenzionalnost veća od dimenzionalnosti originalnog prostora.

U slučaju hardverske realizacije nelinearnih stabala odluka, ova opservacija nam omogućava da koristimo arhitekture za hardversku implementaciju ortogonalnih stabala odluka bez ikakve modifikacije, pod uslovom da se prethodno izvrši proces generisanja veštačkih atributa. Struktura ovakvog sistema prikazana je na slici 3.16.



Slika 3.16 Struktura sistema za hardversku realizaciju nelinearnih stabala odluka

Generator veštačkih atributa ima zadatak da korišćenjem postojećih atributa generiše dodatne atribute formirajući proizvode oblika $A_i^{m_i} \cdot A_j^{m_j} \cdot A_k^{m_k}$, uz uslov $m_i+m_j+m_k \leq 3$. Generalna struktura modula za generisanje veštačkih atributa prikazana je na slici 3.17.



Slika 3.17 Struktura modula za generisanje veštačkih atributa

Nakon što se generišu dodatni atributi, transformisane instance mogu se klasifikovati pomoću nekih od ranije razmatranih *SMPL* i *UN* arhitektura za hardversku realizaciju ortogonalnih stabala odluka.

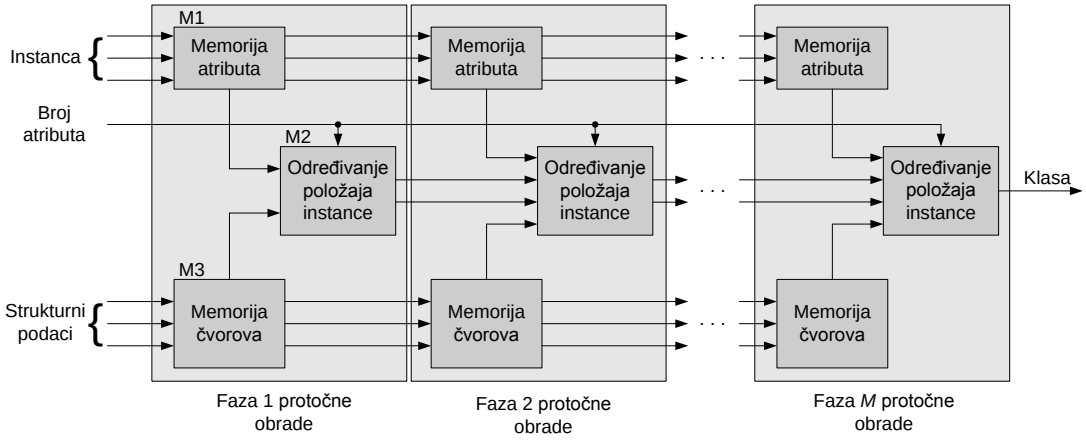
3.3. Programabilne *SMPL* i *UN* arhitekture

Arhitekture za hardversku implementaciju stabala odluka koje su do sada predložene nisu programabilne. Svaka od arhitektura je projektovana tako da se stablo odluke koje se implementira mora definisati pre same sinteze i nakon što se implementira u hardveru ne postoji način da se njegova struktura promeni. Takođe, nije moguće u toku samoga rada promeniti stablo odluke koje je trenutno realizovano. Kada se na početku odabere stablo odluke, više se ne mogu vršiti nikakve izmene na njegovoj strukturi.

Iako ovaj pristup ima svojih prednosti, jer rezultuje u krajnje optimizovanim strukturama u svakom konkretnom slučaju, u velikom broju aplikacija javlja se potreba za promenom prediktivnog modela koji se koristi za klasifikaciju. U ovu grupu spadaju sve vrste adaptivnih klasifikacionih sistema. U ovom poglavlju razmotrićemo modifikacije koje je neophodno izvesti na *SMPL* i *UN* arhitekturama da bi se one učinile programabilnim.

Osnovna modifikacija koju je neophodno izvršiti jeste da se omogući pristup memorijama unutar M3 modula. Na ovaj način bilo bi moguće promeniti njihov sadržaj, a samim tim i strukturu stabla odluke koje je realizovano. Pored toga, potrebno je imati mogućnost da se specificira i broj atributa klasifikacionog problema koji se trenutno rešava.

Slika 3.18 prikazuje modifikovanu *SMPL* arhitekturu koja omogućava promenu stabla odluke koje je realizovano tokom rada sistema.



Slika 3.18 Programabilna SMPL *arhitektura*

Osnovna struktura *SMPL* arhitekture je nepromenjena, međutim uvedeni su neki dodatni signali.

M2 moduli unutar univerzalnih čvorova imaju dodatni ulaz pomoću koga je moguće definisati broj atributa tekućeg klasifikacionog problema. Na ovaj način moguće je promeniti dimenzionalnost prostora atributa u kojem se vrši klasifikacija.

Memorije unutar M3 modula su takođe povezane u jedan pomerački lanac, koji omogućuje upisivanje strukturnih podataka o stablu odluke koje se želi realizovati. Bitno je naglasiti da se ovaj pomerački lanac koristi samo za upisivanje novih strukturnih podataka o stablu koje se realizuje, a ne i u toku samog rada sistema.

Pomoću ove dve modifikacije *SMPL* arhitektura postaje programabilna i u toku rada je moguće promeniti parametre klasifikacionog problema koji se rešava ili stablo odluke koje se koristi za klasifikaciju.

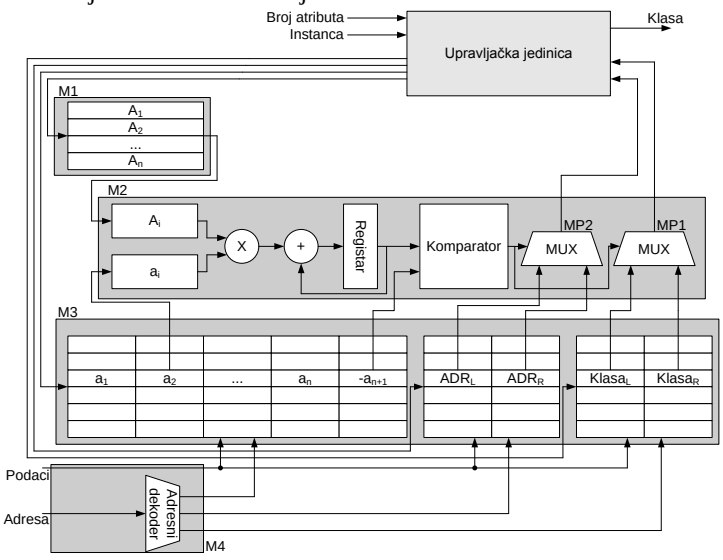
Dimenzionalnost klasifikacionog problema koji se može rešavati ograničena je veličinom memorije unutar M1 modula, dok je veličina stabla odluke koje se može realizovati ograničena veličinom memorija unutar M3 modula i brojem univerzalnih čvorova. Broj univerzalnih čvorova određuje maksimalnu dubinu stabla odluke koje se može realizovati. Veličina memorija unutar M3 modula sa druge strane određuje maksimalni broj čvorova koji se mogu nalaziti na istom nivou u stablu odluke.

Ukoliko, na primer, želimo da projektujemo sistem koji bi bio u stanju da realizuje bilo koje stablo odluke dubine 4, on bi morao da se sastoji iz četiri univerzalna čvora. Veličina memorija unutar M3 modula u prvom univerzalnom čvoru morala bi biti odabrana tako da omogući smeštanje podataka o jednom čvoru, dok bi veličine memorija iz drugog, trećeg i četvrtog univerzalnog čvora morale biti odabrane tako da omoguće smeštanje podataka o dva, četiri i osam čvorova respektivno.

Da bi se *UN* arhitektura učinila programibilnom, potrebno je izvršiti slične modifikacije. Slika 3.19 prikazuje programabilnu *UN* arhitekturu, dobijenu modifikacijom originalne *UN* arhitekture sa slike 3.12.

Slično kao i kod programabilne SMPL arhitekture, upravljačka jedinica ima jedan dodatni ulaz preko koga se može specificirati broj atributa klasifikacionog problema koji se želi rešavati.

Memorijama iz M3 modula sada se može pristupiti spolja i na taj način promeniti njihov sadržaj. Pomoću novog modula M4 koji sadrži adresni dekoder moguće je izvršiti selekciju memorije kojoj se želi pristupiti. Magistrala podataka zajednička je za sve tri memorije.



Slika 3.19 Programabilna UN *arhitektura*

3.4. Potrebni hardverski resursi i performanse *SMPL* i *UN* arhitektura

Ukoliko prilikom implementacije nekog algoritma na raspolaganju postoji mogućnost izbora između nekoliko alternativa arhitektura, moraju se usvojiti kriterijumi od interesa na osnovu kojih je moguće načiniti pravi izbor. Po pravilu ovi kriterijumi uključuju prostornu i vremensku kompleksnost svake od mogućih arhitektura. Pod prostornom kompleksnošću najčešće se podrazumeva veličina memorije za smeštanje programa i podataka, ukoliko se radi o arhitekturi baziranoj na korišćenju mikroprocesora, odnosno potrebnim hardverskim resursima za izvođenje aritmetičkih operacija i veličini memorije za smeštanje podataka, ukoliko se radi o arhitekturi koja se bazira na direktnoj hardverskoj implementaciji. Vremenska kompleksnost podrazumeva vreme potrebno za izvršavanje algoritma implementiranog pomoću odabrane arhitekture. Pored prostorne i vremenske kompleksnosti, prilikom izbora optimalne arhitekture često se kao dodatni kriterijumi uzimaju potrošnja, disipacija, fizičke dimenzije i cena realizovanog sistema.

Upravo iz ovih razloga u ovom poglavlju će biti analizirana prostorna i vremenska kompleksnost predloženih *SMPL* i *UN* arhitektura za hardversku realizaciju stabala odluka. Pod prostornom kompleksnošću podrazumevaćemo potreban broj množača i sabirača kao i količinu memorije za smeštanje relevantnih podataka. Pod vremenskom kompleksnošću podrazumevaćemo vreme potrebno za klasifikaciju pojedinačne instance, odnosno propusnu moć sistema.

Na početku će biti izložena teorijska analiza potrebnih resursa i performansi predloženih *SMPL* i *UN* arhitektura. Ovo je zapravo nastavak i profinjenje rezultata prikazanih u tabeli 3.2.

Nakon toga biće analizirani efekti kvantizacije kritičnih parametara stabla odluke usled korišćenja konačnog broja bita za predstavljanje njihovih vrednosti. Korišćenjem 29 odabranih test problema iz UCI baze problema [8] biće analizirani efekti predstavljanja koeficijenata hiperravni sa različitim brojem bitova na ukupnu tačnost stabla odluke. Ovaj podataka je veoma bitan prilikom hardverske implementacije, jer je potrebno koristiti što je moguće manji broj bitova za predstavljanje relevantnih podataka u sistemu da bi se smanjili potrebni hardverski resursi. Pri tome ova minimizacija ne sme ići na uštrb tačnosti realizovanog sistema.

Na kraju, biće izvršena estimacija neophodnih resursa i performansi *SMPL* i *UN* arhitektura prilikom hardverske implementacije stabala odluka formiranih za rešavanje 29 konkretnih UCI test problema. Kao platforma za hardversku implementaciju biće korišćene FPGA komponente kompanije *Xilinx*. Dobijeni rezultati biće upoređeni sa performansama koje postiže standardna softverska implementacija stabala odluka koja se izvršava na PC računaru.

3.4.1. Teorijska analiza potrebnih resursa i performansi

U tabeli 3.2 prikazana je estimacija potrebnih hardverskih resursa i performansi *SMPL* i *UN* arhitektura i arhitektura koje su ranije predložene za hardversku implementaciju stabala odluka. Kao mera potrebnih hardverskih resursa usvojen je broj čvorova koji je potrebno realizovati kao nezavisne module za svaku od predloženih arhitektura. Međutim, nakon upoznavanja sa detaljima *SMPL* i *UN* arhitektura, moguće je hardversku kompleksnost proceniti preciznije i izraziti je brojem zahtevanih množača, sabirača i veličini memorije neophodne za smeštanje relevantnih informacija. Ova procena je mnogo preciznija i pruža mnogo bolji uvid u razlike između različitih arhitektura. Ona pogotovo dolazi do izražaja imajući u vidu da i *SMPL* i *UN* arhitektura dolaze u dve varijante, sa serijskom i sa paralelnom evaluacijom testova, koje se znatno razlikuju po potrebnim hardverskim resursima kada se oni izražavaju u terminima množača, sabirača i memoriji, a uopšte se ne razlikuju kada se neophodni resursi izražavaju u terminima čvorova koje je potrebno implementirati.

Tabela 3.5 sadrži podatke o neophodnim hardverskim resursima izraženim u broju množača, sabirača i veličini memorije, kao i podatke o brzini klasifikacije izraženoj u terminima propusne moći za sledeće arhitekture:

1. *SMPL1 – SMPL* arhitektura kod koje se pozicija instance u odnosu na hiperravan računa serijski,
2. *SMPL2 – SMPL* arhitektura kod koje se pozicija instance u odnosu na hiperravan računa paralelno,
3. *UN1 – UN* arhitektura kod koje se pozicija instance u odnosu na hiperravan računa serijski,
4. *UN2 – UN* arhitektura kod koje se pozicija instance u odnosu na hiperravan računa paralelno.

Neophodni hardverski resursi, kao i performanse četiri navedene arhitekture izraženi su u terminima:

- broja atributa posmatranog klasifikacionog problema, n
- broja klasa posmatranog klasifikacionog problema, N_{cl}
- broja čvorova u stablu odluke, N_{dt}
- dubine stabla odluke, M
- broja posećenih čvorova prilikom klasifikacije tekuće instance, N_{nv}
- broja bitova koji se koriste za reprezentaciju vrednosti atributa klasifikacionog problema, N_a
- broja bitova koji se koriste za reprezentaciju vrednosti koeficijenata hiperpovrši u svakom od čvorova stabla odluke, N_c
- trajanja periode globalnog sinhronizacionog signala (takta), CP

U tabeli 3.5, u koloni koja sadrži podatke o potrebnim memorijskim resursima za svaku od četiri arhitekture 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 broju instanci koje se mogu klasifikovati u jedinici vremena.

Tabela 3.5 Potrebni resursi za hardversku implementaciju i performanse *SMPL* i *UN* arhitektura

Arhitektura	Memorija	Sabirači		Množači		Propusna moć
SMPL1	$M\cdot(n \times N_a)$, $(n+1)\cdot N_{dt} \times N_c$, $2\cdot N_{dt} \times \lceil ld(N_{dt}) \rceil$, $2\cdot N_{dt} \times \lceil ld(N_{cl}) \rceil$	M		M		$\frac{1}{(n+1)\cdot CP}$
SMPL2	$M\cdot(n \times N_a)$, $(n+1)\cdot N_{dt} \times N_c$, $2\cdot N_{dt} \times \lceil ld(N_{dt}) \rceil$, $2\cdot N_{dt} \times \lceil ld(N_{cl}) \rceil$	2.1	$M\cdot(n-1)$	2.2	$M\cdot n$	$\frac{1}{CP}$
UN1	$n \times N_a$, $(n+1)\cdot N_{dt} \times N_c$, $2\cdot N_{dt} \times \lceil ld(N_{dt}) \rceil$, $2\cdot N_{dt} \times \lceil ld(N_{cl}) \rceil$	1		1		$\frac{1}{N_{nv}\cdot(n+1)\cdot CP}$
UN2	$n \times N_a$, $(n+1)\cdot N_{dt} \times N_c$, $2\cdot N_{dt} \times \lceil ld(N_{dt}) \rceil$, $2\cdot N_{dt} \times \lceil ld(N_{cl}) \rceil$	2.3	$n-1$	2.4	n	$\frac{1}{N_{nv}\cdot CP}$

Na osnovu tabele 3.5 može se zaključiti sledeće.

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. Ovaj zaključak važi i za broj neophodnih sabirača u slučaju *SMPL2* i *UN2* arhitektura. Na osnovu ovih opservacija može se zaključiti da se sve četiri predložene arhitekture izuzetno dobro skaliraju sa porastom složenosti problema koji se rešava, što predstavlja izuzetno dobru osobinu pogotovo u slučaju hardverske implementacije.

Analizom *SMPL* arhitektura može se zaključiti da *SMPL1* i *SMPL2* arhitekture imaju identične memorijske zahteve, ali da se broj neophodnih sabirača i množača drastično razlikuje. Kao što se moglo i očekivati, *SMPL2* arhitektura koja koristi paralelni način evaluacije pozicije instance u odnosu na hiperravan zahteva značajno veći broj množača i sabirača u poređenju sa *SMPL1* arhitekturom.

Slični zaključci mogu se izvući i kada se posmatraju dve *UN* arhitekture za hardversku implementaciju stabla odluke.

UN1 arhitektura ima izuzetno dobru osobinu da je broj neophodnih sabirača i množača nezavisan od bilo kojeg parametra problema. Prilikom korišćenja *UN1* arhitekture za realizaciju stabla odluke uvek je neophodan jedan sabirač i jedan množač, nezavisno od karakteristika problema koji se trenutno rešava.

Upoređujući *SMPL* i *UN* arhitekture, može se primetiti da su im memorijski zahtevi gotovo identični. *SMPL* arhitekture zahtevaju nešto veću memoriju za smeštanje instanci koje se trenutno klasifikuju (tačno *M* puta veću) usled protočne obrade. Međutim, ova memorija u većini slučajeva nije dominantna u ukupnoj količini neophodne memorije. U ukupnoj zahtevanoj memoriji dominira član koji je određen strukturom stabla odluke (koeficijenti hiperravni, adrese čvorova naslednika i klase asocirane naslednicima), a on je identičan i za *SMPL* i *UN*

arhitekture. Kada uporedimo potreban broj množača i sabirača za *SMPL1* i *UN1* kao i za *SMPL2* i *UN2* arhitekture vidimo da odgovarajuće *SMPL* arhitekture zahtevaju tačno *M* puta više množača i sabirača. Ovo je očekivano obzirom da *SMPL* arhitekture imaju ukupno *M* univerzalnih čvorova, dok *UN* arhitekture imaju samo jedan. Iz istog razloga, propusna moć *SMPL* arhitektura u najgorem slučaju je *M* puta veća od odgovarajućih *UN* arhitektura. I ovaj rezultat je posledica protočne obrade koja se koristi u *SMPL* arhitekturama.

4. Primena

Nakon teorijske analize potrebnih hardverskih resursa i brzine klasifikacije *SMPL* i *UN* arhitektura, 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*“, [8]. 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* 9.1.03 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 u slučaju *UN1* arhitekture ima sledeći izgled.

```
entity serial_dt 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 := 9;                  -- number of bits used to represent class
      num_attributes_g: integer := 33;            -- number of attributes
      att_mem_bus_size_g: integer := 8;           -- size of the address bus for the attribute memory
      num_nodes_g: integer := 5;                  -- maximum number of nodes in the DT
      coef_mem_bus_size_g: integer := 17; -- size of the address bus for the coefficient memory
      rln_mem_bus_size_g: integer := 12      -- size of the address bus for the rigth-left node memory
            );
```

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

        load_att_i      : in std_logic;                                     -- load attribute signal
        start_i         : in std_logic;                                     -- start signal
        finished_o      : out std_logic;                                    -- finished classification signal

        A_i             : in  std_logic_vector( attribute_res_g - 1 downto 0); -- attribute input
        class_o         : out std_logic_vector( class_res_g - 1 downto 0)   -- output class
            );
end entity serial_dt;
```

Izgled *entity* deklaracije u slučaju *UN1* arhitekture

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

4.1. Neophodni hardverski resursi

Tabele 4.2-4.5 sadrže podatke o neophodnim hardverskim resursima za implementaciju *SMPL* 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 četiri tabele sadrži informacije o neophodnim hardverskim resursima za po jednu *SMPL* odnosno *UN* arhitekturu.

Tabela 4.2 Neophodni hardverski resursi u slučaju FPGA implementacije *SMPL1* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	0.82	2.43	6.02	6.02
bc	0.28	1.05	7.16	7.16
bsc	0.36	0.62	4.14	4.14
bcw	0.50	1.17	5.70	5.70
car	0.42	1.61	7.12	7.12
cmc	1.14	12.52	12.94	12.94
ger	1.60	6.15	6.82	6.82
gls	0.54	1.57	6.14	6.14
hep	0.30	0.32	1.60	1.60
hrtc	0.79	3.01	6.26	6.26
hrts	0.51	1.00	3.98	3.98
Ion	1.10	1.41	3.30	3.30
irs	0.12	0.18	3.08	3.08
liv	0.44	1.54	7.48	7.48
lym	0.59	0.95	3.38	3.38
page	0.94	6.43	9.58	9.58
pid	0.65	3.05	8.32	8.32
son	1.41	1.86	2.40	2.40
thy	0.11	0.15	2.20	2.20
ttt	0.41	0.71	4.66	4.66
veh	1.62	5.23	9.22	9.22
vote	0.30	0.34	1.90	1.90
vow	1.09	8.64	8.60	8.60
w21	2.54	16.72	12.40	12.40
w40	3.99	24.59	10.22	10.22
wdbc	0.84	0.95	2.86	2.86
wine	0.26	0.29	2.02	2.02
wpbc	1.03	1.58	3.20	3.20
zoo	0.61	1.13	3.70	3.70

Tabela 4.3 Neophodni hardverski resursi u slučaju FPGA implementacije *SMPL2* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	0.82	2.43	78.26	84.28
bc	0.28	1.05	21.48	28.64

bsc	0.36	0.62	33.12	37.26
bcw	0.50	1.17	45.60	51.30
car	0.42	1.61	35.60	42.72
cmc	1.14	12.52	103.52	116.46
ger	1.60	6.15	156.86	163.68
gls	0.54	1.57	49.12	55.26
hep	0.30	0.32	28.80	30.40
hrtc	0.79	3.01	75.12	81.38
hrts	0.51	1.00	47.76	51.74
ion	1.10	1.41	108.90	112.20
irs	0.12	0.18	9.24	12.32
liv	0.44	1.54	37.40	44.88
lym	0.59	0.95	57.46	60.84
page	0.94	6.43	86.22	95.80
pid	0.65	3.05	58.24	66.56
son	1.41	1.86	141.60	144.00
thy	0.11	0.15	8.80	11.00
ttt	0.41	0.71	37.28	41.94
veh	1.62	5.23	156.74	165.96
vote	0.30	0.34	28.50	30.40
vow	1.09	8.64	103.20	111.80
w21	2.54	16.72	248.00	260.40
w40	3.99	24.59	398.58	408.80
wdbc	0.84	0.95	82.94	85.80
wine	0.26	0.29	24.24	26.26
wpbc	1.03	1.58	102.40	105.60
zoo	0.61	1.13	59.20	62.90

Tabela 4.4 Neophodni hardverski resursi u slučaju FPGA implementacije *UN1* arhitekture za 29 odabranih UCI test problema

Problem	***M1 (kbits)***	***M3 (kbits)***	***Sabirači***	***Množači***
ausc	0.14	2.43	1.00	1.00
bc	0.04	1.05	1.00	1.00
bsc	0.09	0.62	1.00	1.00
bcw	0.09	1.17	1.00	1.00
car	0.06	1.61	1.00	1.00
cmc	0.09	12.52	1.00	1.00
ger	0.23	6.15	1.00	1.00
gls	0.09	1.57	1.00	1.00
hep	0.19	0.32	1.00	1.00
hrtc	0.13	3.01	1.00	1.00
hrts	0.13	1.00	1.00	1.00
ion	0.33	1.41	1.00	1.00
irs	0.04	0.18	1.00	1.00
liv	0.06	1.54	1.00	1.00
lym	0.18	0.95	1.00	1.00
page	0.10	6.43	1.00	1.00
pid	0.08	3.05	1.00	1.00
son	0.59	1.86	1.00	1.00
thy	0.05	0.15	1.00	1.00
ttt	0.09	0.71	1.00	1.00
veh	0.18	5.23	1.00	1.00
vote	0.16	0.34	1.00	1.00
vow	0.13	8.64	1.00	1.00
w21	0.21	16.72	1.00	1.00
w40	0.39	24.59	1.00	1.00
wdbc	0.29	0.95	1.00	1.00
wine	0.13	0.29	1.00	1.00
wpbc	0.32	1.58	1.00	1.00
zoo	0.17	1.13	1.00	1.00

Tabela 4.5 Neophodni hardverski resursi u slučaju FPGA implementacije *UN2* arhitekture za 29 odabranih UCI test problema

Problem	*M1 (kbits)*	*M3 (kbits)*	*Sabirači*	*Množači*
ausc	0.14	2.43	13.00	14.00
bc	0.04	1.05	3.00	4.00
bsc	0.09	0.62	8.00	9.00
bcw	0.09	1.17	8.00	9.00
car	0.06	1.61	5.00	6.00
cmc	0.09	12.52	8.00	9.00
ger	0.23	6.15	23.00	24.00
gls	0.09	1.57	8.00	9.00
hep	0.19	0.32	18.00	19.00
hrtc	0.13	3.01	12.00	13.00
hrts	0.13	1.00	12.00	13.00
ion	0.33	1.41	33.00	34.00
irs	0.04	0.18	3.00	4.00
liv	0.06	1.54	5.00	6.00
lym	0.18	0.95	17.00	18.00
page	0.10	6.43	9.00	10.00
pid	0.08	3.05	7.00	8.00
son	0.59	1.86	59.00	60.00
thy	0.05	0.15	4.00	5.00
ttt	0.09	0.71	8.00	9.00
veh	0.18	5.23	17.00	18.00
vote	0.16	0.34	15.00	16.00
vow	0.13	8.64	12.00	13.00
w21	0.21	16.72	20.00	21.00
w40	0.39	24.59	39.00	40.00
wdbc	0.29	0.95	29.00	30.00
wine	0.13	0.29	12.00	13.00
wpbc	0.32	1.58	32.00	33.00
zoo	0.17	1.13	16.00	17.00

Rezultati eksperimenata izvedenih na 29 odabranih UCI test problema prikazani u tabelama 4.2-4.5 u saglasnosti su sa opštim izrazima iz tabele 3.5. Ovi rezultati takođe mogu da posluže za bolju procenu neophodnih hardverskih resursa u slučaju rešavanja konkretnih, realnih problema. Na osnovu podataka iz tabela 4.2-4.5 može se zaključiti da zahtevani resursi ne prevazilaze resurse koje obezbeđuju savremene FPGA komponente. 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 *SMPL* i *UN* arhitekture za svaki od 29 razmatranih UCI test problema.

Ukoliko analiziramo količinu neophodne veličine memorijskih resursa najzahtevnije su *SMPL1* i *SMPL2* arhitekture za *w40* test problem, koje zahtevaju svaka po 28.6 kilobita. Ovaj količina je zanemariva u odnosu na ukupno 18576 kilobita memorije koje nam stoje na raspolaganju u *XC5VSX240T* komponenti.

Naviše sabirača zahteva *SMPL2* arhitektura takođe u slučaju *w40* test problema, u proseku 398.58. *XC5VSX240T* komponenta na raspolaganju ima ukupno 1056 hardverskih sabiračkih modula. Očigledno je da je ovaj broj više nego dovoljan za implementaciju posmatrane *SMPL2* arhitekture.

Najveći broj množača potreban je za implementaciju *SMPL2* arhitekture u slučaju *w40* test problema, u proseku 408.80. Situacija je slična kao i u slučaju sabirača, obzirom da *XC5VSX240T* komponenta na raspolaganju ima ukupno 1056 hardverskih množača, posmatrana *SMPL2* arhitektura se bez problema može realizovati pomoću ove FPGA komponente.

Već prilikom teorijske analize nagoveštено je da paralelne arhitekture, *SMPL2* i *UN2*, zahtevaju daleko više sabirača i množača od serijskih. Takođe je pokazano da što se tiče potrebnih memorijskih resursa, sve četiri arhitekture imaju podjednake zahteve. Analizom eksperimentalnih rezultata prikazanih u tabelama 4.2-4.5, zaključci teorijske analize mogu se potvrditi. Ono što rezultati eksperimenata još bolje pokazuju jeste konkretna količina resursa koji su neophodni da bi se bilo koja od predloženih arhitektura koristila u slučaju realnih problema. Na osnovu svih rezultata može se zaključiti da se sve četiri predložene arhitekture za hardversku realizaciju stabala odluka mogu koristiti bez većih problema, čak i u slučaju FPGA komponenti sa skromnijim mogućnostima.

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 četiri predložene arhitekture. Takođe je interesantno uporediti dobijene rezultate sa rezultatima postignutim u slučaju softverske implementacije stabala odluka u dva slučaja: kada se softverska implementacija izvršava na savremenom desktop računaru i kada se izvršava na savremenom *embedded* mikroprocesoru. U slučaju softverske implementacije stabala odluka implementirana 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 sa ukupno 3 GB operativne memorije. Ovaj računar trenutno predstavlja jednu od najmoćnijih raspoloživih desktop platformi.

SMPL1, *SMPL2*, *UN1* i *UN2* arhitekture implementirane su pomoću *Xilinx* FPGA komponente familije *Virtex 5*. Tabela 4.6 sadrži podatke o minimalnoj periodi globalnog sinhronizacionog signala, različitih sistema za hardversku implementaciju stabla odluke baziranih na četiri predložene arhitekture 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 4.7 predstavljaju prosečna ubrzanja u vremenu klasifikacije instance, za svaku od četiri predložene 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 po 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 formirano je stablo odluke, a zatim je izmereno prosečno vreme klasifikacije instance za svaku od razmatranih implementacija. Vrednosti prikazane u tabeli 4.7 izračunate su kao količnik prosečnog vremena klasifikacije instance pomoću odgovarajuće softverske implementacije i vremena klasifikacije instance pomoću hardverske implementacije.

Tabela 4.6 Minimalne periode takta sistema za hardversku realizaciju stabla odluke baziranih na *SMPL* i *UN* arhitekturama za odabrane UCI probleme

Test skup	**Minimalna perioda globalnog sinhronizacionog signala [ns]**			
	SMPL1	***SMPL2***	***UN1***	***UN2***
ausc	6.531	6.053	5.063	12.740
bc	4.972	4.265	4.853	9.524
bcw	4.998	4.184	3.981	12.040
bsc	4.989	5.000	4.006	14.016
car	6.365	4.165	4.812	10.120
cmc	7.066	7.131	5.548	14.277
ger	6.774	7.509	4.389	14.005
gls	6.461	6.958	4.115	14.016
hep	4.870	3.011	3.926	14.256
hrtc	6.833	6.248	4.248	13.773
hrts	4.989	4.544	4.323	13.773
ion	6.459	6.250	4.379	14.116
irs	4.945	4.081	3.941	7.019
liv	6.365	4.047	4.812	10.120
lym	4.995	6.448	4.278	14.269
page	6.928	6.247	5.372	11.464
pid	6.719	7.139	4.930	10.620
son	6.548	6.164	5.238	14.283
thy	4.976	3.998	3.976	9.618
ttt	4.974	4.347	4.006	14.016
veh	6.595	6.514	4.135	14.280
vote	4.978	4.906	3.940	11.607
vow	6.432	6.665	5.223	14.260
w21	7.348	6.793	5.439	14.250
w40	7.250	6.343	5.487	14.268
wdbc	6.211	6.672	4.145	11.812
wine	4.983	4.500	4.015	12.377
wpbc	6.465	6.666	3.969	14.786
zoo	4.990	5.875	4.063	14.235
AVG [ns]	5.966	5.611	4.504	12.757
AVG [MHz]	167.62	178.22	222.03	78.39

Tabela 4.7 Ubrzanje hardverskih u odnosu na softversku implementaciju

Test skup	PC			
	SMPL1	***SMPL2***	***UN1***	***UN2***
ausc	2.58	41.68	0.92	5.48
bc	5.57	32.47	1.81	4.60
bcw	2.28	27.20	1.41	4.66
bsc	3.94	39.27	1.48	4.22
car	3.53	37.77	1.55	5.15
cmc	7.01	69.43	1.24	4.84
ger	2.54	57.20	0.91	7.15
gls	3.54	32.91	1.35	3.97
hep	0.93	30.24	1.01	5.55
hrtc	2.76	42.19	1.13	4.88
hrts	2.54	39.07	1.10	4.83
ion	1.25	45.13	0.78	8.47
irs	3.35	20.28	2.15	6.05
liv	5.11	56.24	1.49	4.98
lym	1.86	27.32	1.00	5.69
page	3.96	48.33	0.95	4.91
pid	4.51	38.24	1.30	5.41
son	0.86	56.02	0.57	12.72
thy	2.74	20.46	1.80	4.45
ttt	2.91	33.29	1.39	3.97
veh	3.10	59.63	1.03	5.64
vote	1.26	21.76	1.09	6.30
vow	4.29	57.92	0.92	4.74
w21	3.37	80.19	0.83	6.96
w40	2.57	120.64	0.63	9.99
wdbc	1.03	29.83	0.88	9.52
wine	1.59	24.62	1.16	5.27
wpbc	1.22	40.21	0.88	8.02
zoo	2.31	35.30	1.10	5.65
AVG	2.91	43.61	1.17	6.00

Rezultati prikazani u tabeli 4.7 dobijeni su pod sledećim uslovima:

- četiri arhitekture za implementaciju stabla odluke realizovane su pomoću FPGA komponente familije *Virtex 5* koja je u proseku radila na sledećim učestanostima:
  - *SMPL1* arhitektura, 167.62 MHz
  - *SMPL2* arhitektura, 178.22 MHz
  - *UN1* arhitektura, 222.03 MHz
  - *UN2* arhitektura, 78.39 MHz
- učestanost *Intel Pentium* D 820 mikroprocesora na kome je izvršavana softverska implementacija iznosila je 2.8 GHz,

Kao što se može videti iz tabele 4.7 sve četiri arhitekture za hardversku implementaciju pružaju znatna ubrzanja u vremenu klasifikacije instance u odnosu na softverske implementacije.

U poređenju sa PC implementacijom, prosečno skraćenje vremena potrebnog za formiranje stabla odluke odnosno ansambla stabala odluka iznosi:

- 2.91 puta u slučaju korišćenja *SMPL1* arhitekture,
- 43.61 puta u slučaju korišćenja *SMPL2* arhitekture,
- 1.17 puta u slučaju korišćenja *UN1* arhitekture,
- 6.00 puta u slučaju korišćenja *UN2* arhitekture.

Čak i u poređenju sa jednim od trenutno najjačih desktop računara, paralelne arhitekture, implementirane pomoću FPGA komponenti, pružaju značajna skraćenja u vremenu klasifikacije instanci. Serijske arhitekture imaju tek nešto veću brzinu klasifikacije u odnosu na PC implementaciju, međutim treba imati na umu da one zahtevaju znatno manje hardverskih resursa. Pored toga potrebno je napomenuti da su hardverske implementacije *SMPL1*, *SMPL2*, *UN1* i *UN2* arhitektura radile u proseku na 16.7, 15.7, 12.6 i 35.7 puta manjoj učestanosti od softverske implementacije respektivno.

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] Lopez-Estrada, S., and Cumplido, R., "Decision Tree Based FPGA Architecture for Texture Sea State Classification", Reconfigurable Computing and FPGA's ReConFig 2006 IEEE International Conference, September 2006, pp. 1-7.

[5] 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.

[6] I. K. Sethi, "Entropy net: From decision trees to neural nets," Proc. IEEE, vol. 78, pp. 1605–1613, Oct. 1990.

[7] I. K. Sethi, "Neural implementation of tree classifiers," IEEE Trans. Syst., Man, Cybern., vol. 25, pp. 1243–1249, Aug. 1995.

[8] 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
ИНТЕГРИСАНИ СИСТЕМ МЕНАЏМЕНТА СЕРТИФИКОВАН ОД:

Наш број:
Ваш број:
Датум: 2012-02-02

ИЗВОД ИЗ ЗАПИСНИКА

Наставно-научног већа Факултета техничких наука у Новом Саду, на 30. редовној седници одржаној дана 25.01.2012. године, донело је следећу одлуку:

-непотребно изостављено-

Тачка 13. Питања научноистраживачког рада и међународне сарадње

У циљу доношења одлуке о прихватању
Техничког решења

ИП ЈЕЗГРО ЗА ХАРДВЕРСКУ РЕАЛИЗАЦИЈУ СТАБАЛА ОДЛУКА

Аутори: доц. др Растислав Струхарик и проф. др Ладислав Новак

именују се рецензенти:

1. Др Драган Васиљевић, ред. проф. Електротехнички факултет, Београд
2. Др Теуфик Токић, ред. проф. Електронски факултет, Ниш

Техничко решење је развијено у оквиру пројекта **ТР 32016**

-непотребно изостављено-

Записник водила:
Јасмина Димић, дипл. правник

Тачност података оверава:
Секретар
Иван Нешковић, дипл. правник

Декан
Проф. др Илија Ћосић

Softver:
IP jezgra za hardversku realizaciju 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: 2010 - 2011.
Primena: jun 2011.

Kratak opis

Hardverske implementacije 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 stabala odluka koja u mnogome olakšavaju integraciju i rada sa stablima odluka prilikom projektovanja *embedded* sistema.

Tehničke karakteristike:
IP jezgra za hardversku realizaciju 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 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; 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 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 ortogonalnih, neortogonalnih i nelinearnih stabala odluka. Analizom postojećih rešenja utvrđeno je da u dostupnoj literaturi postoje samo dva rada koji se bave problematikom hardverske implementacije stabala odluka. Obe postojeće arhitekture zapravo su zasnovane na vrlo neefikasnom pristupu za hardversku implementaciju stabala odluka koji se bazira na implementiranju svih čvorova 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.

Arhitektura koja bi uzimala u obzir činjenicu da je prilikom klasifikacije bilo koje instance dovoljno evaluira samo podskup svih čvorova u stablu odluke bila bi mnogo efikasnija u pogledu zahtevanih hardverskih resursa u odnosu na ranije predložene arhitekture. Umesto da se implementira svaki čvor u stablu odluke kao nezavisan modul, autori tehničkog rešenja predložili su arhitekturu koja implementira samo po jedan univerzalni čvor u svakom od nivoa stabla odluke. Ova arhitektura nazvana je *SMPL* (*Single Module Per Layer*). *SMPL* arhitektura sastoji se od niza univerzalnih čvorova čiji broj je jednak dubini stabla odluke koje se implementira. Svaki od univerzalnih čvorova asociran je svim čvorovima iz odgovarajućeg nivoa stabla odluke i u stanju je da evaluira sve testove koji su u njima specificirani.

SMPL arhitektura zahteva značajno manje hardverske resurse od ranije predloženih arhitektura, jer je u slučaju *SMPL* arhitekture, u svakom od nivoa stabla potrebno realizovati samo po jedan modul za evaluaciju testova u svim asociranim čvorovima na odgovarajućoj dubini u stablu odluke. Bitno je naglasiti da su memorijski resursi neophodni za smeštanje koeficijenata koji definišu testove isti i u slučaju *SMPL* arhitekture i arhitektura predloženih u dostupnoj literaturi. Ovo je sasvim razumljivo, s obzirom da je svako stablo odluke zapravo jednoznačno definisano preko skupa svih koeficijenata koji se nalaze u testovima unutar svih čvorova stabla odluke i načinom na koji su čvorovi međusobno povezani. Svaka modifikacija broja ili vrednosti koeficijenata, a samim tim i smanjenje njihovog broja, vodila bi ka realizaciji nekog drugog a ne željenog stabla odluke, te stoga nije ni moguća. Takođe je bitno naglasiti da iako *SMPL* arhitektura predstavlja efikasniju arhitekturu za realizaciju stabala odluka od ranije predloženih arhitektura, njena brzina klasifikacije je ista.

U slučaju *UN* arhitekture jedan univerzalni čvor koristi se za evaluaciju testova koji se nalaze u svim čvorovima stabla odluke. Testovi se evaluiraju sekvencijalno, počevši od korena stabla. U zavisnosti od ishoda testa evaluira se test definisan u jednom od dva čvora naslednika. Ovaj postupak se nastavlja sve dok se ne dođe do nekog od listova stabla odluke. *UN* arhitektura zapravo vrši evaluaciju stabla odluke sekvencijalno, čvor po čvor. Ovo je identičan način na koji se stablo evaluira u slučaju softverske implementacije. U slučaju softverske implementacije neophodno je postojanje nekog mikroprocesora da bi se izvršio program koji evaluira stablo odluke. Međutim mikroprocesor predstavlja vrlo neefikasno rešenje sa stanovišta neophodnih hardverskih resursa, jer je on projektovan tako da je u stanju da izvrši proizvoljni program, a ne samo onaj koji evaluira stablo odluke. Za razliku od mikroprocesorskog sistema, *UN* arhitektura je projektovana samo sa jednom namenom, da efikasno evaluira stablo odluke, te je njena složenost višestruko manja od složenosti čak i najjednostavnijeg mikroprocesora. Takođe i brzina klasifikacije *UN* arhitekture bi trebalo da bude veća od brzine klasifikacije sistema baziranog na mikroprocesoru, iz istog razloga.

UN arhitektura takođe zahteva značajno manje resursa u poređenju sa *SMPL* arhitekturom, a pogotovo u poređenju sa arhitekturama predloženim u radovima dostupnim u naučnoj literaturi. Međutim, kada se poredi brzina klasifikacije, *UN* arhitektura ima značajno duže vreme klasifikacije.

SMPL i *UN* arhitekture predstavljaju dobar primer različitih kompromisa između brzine rada i neophodnih hardverskih resursa. *SMPL* arhitektura optimizovana je za postizanje velike brzine klasifikacije po cenu velike hardverske složenosti. Za razliku od nje, *UN* arhitektura optimizovana je tako da zahteva minimalne resurse, ali je to „plaćeno" malom brzinom klasifikacije.

U skladu sa gore iznetim činjenicama tehničko rešenje ispunjava uslove da bude priznato kao softver (odnosno M85 u skladu sa Pravilnikom 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 M. Vasiljević

Softver:
IP jezgra za hardversku realizaciju 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: 2010 - 2011.
Primena: jun 2011.

Kratak opis

Hardverske implementacije 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 stabala odluka koja u mnogome olakšavaju integraciju i rada sa stablima odluka prilikom projektovanja *embedded* sistema.

Tehničke karakteristike:
IP jezgra za hardversku realizaciju 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 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; 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 stabala odluka" autora doc. dr Rastislava Struharika, i prof. dr Ladislava Novaka, realizovano 2010.-2011. godine, prikazano je na 31 stranici A4 formata, koje sadrže 23 slika i grafičkih ilustracija, 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 stabala odluka" autora doc. dr Rastislava Struharika, i prof. dr Ladislava Novaka, mogu se izvesti sledeći zaključci:
1. Dokumentacija tehnilkog rešenja jasno prikazuje kompletnu strukturu tehnilkog 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 stabala odluka", predstavlja efikasan alat za rešavanje problema u oblasti hardverske implementacije jedne grupe algoritama mašinskog učenja, 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 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, 14.02.2012. godine

Prof. dr Teufik Tokić,
Univerzitet u Nišu
Elektronski fakultet

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

ИНТЕГРИСАНИ СИСТЕМ МЕНАЏМЕНТА СЕРТИФИКОВАН ОД:

Наш број: _____
Ваш број: _____
Датум: 2012-02-27

ИЗВОД ИЗ ЗАПИСНИКА

Наставно-научног већа Факултета техничких наука у Новом Саду, на 31. редовној седници одржаној дана 22.02.2012. године, донело је следећу одлуку:

-непотребно изостављено-

Тачка 13. Питања научноистраживачког рада и међународне сарадње

На основу извештаја рецензената прихвата се
Техничко решење – софтвер (М85) под називом:

ИП ЈЕЗГРО ЗА ХАРДВЕРСКУ РЕАЛИЗАЦИЈУ СТАБАЛА ОДЛУКА

Аутори: доц. др Растислав Струхарик и проф. др Ладислав Новак

Техничко решење је развијено у оквиру пројекта **ТР 32016**

-непотребно изостављено-

Записник водила:
Јасмина Димић, дипл. правник

Тачност података оверава:
Секретар
Иван Нешковић, дипл. правник

Декан
Проф. др Илија Ћосић