Докторска дисертација - Реконфигурабилне архитектуре за хардверску акцелерацију предиктивних модела машинског учења

кандидат: Вук Врањковић
ментор: проф. др Ладислав Новак

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска публикација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Докторска дисертација
Аутор, **АУ**:		Вук Врањковић
Ментор, **МН**:		др Ладислав Новак, редовни професор
Наслов рада, **НР**:		Реконфигурабилне архитектуре за хардверску акцелерацију предиктивних модела машинског учења
Језик публикације, **ЈП**:		Српски
Језик извода, **ЈИ**:		Српски/Енглески
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2015
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Факултет техничких наука, Трг Доситеја Обрадовића 6, Нови Сад
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		8/139/156/8/33/35/0
Научна област, **НО**:		Електроника
Научна дисциплина, **НД**:		Електроника
Предметна одредница/Кључне речи, **ПО**:		Стабла одлука, SVM, вештачке неуронске мреже, хардверска акцелерација, ансамбли класификатора, реконфигурабилни хардвер
УДК		
Чува се, **ЧУ**:		Библиотека Факултета техничких наука у Новом Саду, Трг Доситеја Обрадовића 6, 21000 Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У овој дисертацији представљене су универзалне реконфигурабилне архитектуре грубог степена гранулације за хардверску имплементацију DT (decision trees), ANN (artificial neural networks) и SVM (support vector machines) предиктивних модела као и хомогених и хетерогених ансамбала. Коришћењем ових архитектура реализоване су две врсте DT модела, две врсте ANN модела, две врсте SVM модела и седам врста ансамбала на FPGA (field programmable gate arrays) чипу. Експерименти, засновани на скуповима из стандардне UCI базе скупова за машинско учење, показују да FPGA имплементација омогућава значајно убрзање (од 1 до 6 редова величине) просечног времена потребног за предикцију, у поређењу са софтверским решењима.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	др Растислав Струхарик, доцент
	Члан:	др Предраг Теодоровић, доцент
	Члан:	др Иван Мезеи, доцент
	Члан:	др Теуфик Токић, редовни професор
	Члан, ментор:	др Ладислав Новак, редовни професор

Потпис ментора

Образац **Q2.НА.06-05**- Издање 1

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		Monograph
Type of record, **TR**:		Printed text
Contents code, **CC**:		Ph.D. Thesis
Author, **AU**:		Vuk Vranjkovic
Mentor, **MN**:		Ladislav Novak, Ph.D., full professor
Title, **TI**:		
Language of text, **LT**:		Serbian
Language of abstract, **LA**:		Serbian/English
Country of publication, **CP**:		Serbia
Locality of publication, **LP**:		Vojvodina
Publication year, **PY**:		2015
Publisher, **PB**:		Author's reprint
Publication place, **PP**:		Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		8/139/156/8/33/35/0
Scientific field, **SF**:		Electronics
Scientific discipline, **SD**:		Electronics
Subject/Key words, **S/KW**:		Decision trees, support vector machines, artificial neural networks, hardware acceleration, ensemble classifiers, reconfgurable hardware
UC		
Holding data, **HD**:		
Note, **N**:		
Abstract, **AB**:		This thesis proposes universal coarse-grained reconfigurable computing architectures for hardware implementation of decision trees (DTs), artificial neural networks (ANNs), support vector machines (SVMs), and homogeneous and heterogeneous ensemble classifiers (HHESs). Using these universal architectures, two versions of DTs, two versions of SVMs, two versions of ANNs, and seven versions of HHESs machine learning classifiers, have been implemented in field programmable gate arrays (FPGA). Experimental results, based on datasets of standard UCI machine learning repository database, show that FPGA implementation provides significant improvement (1–6 orders of magnitude) in the average instance classification time, in comparison with software implementations.
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		
Defended Board, **DB**:	President:	Rastislav Struharik, Ph.D., ass. professor
	Member:	Predrag Teodorovic, Ph.D., ass. professor
	Member:	Ivan Mezei, Ph.D., ass. professor
	Member:	Teufik Tokic, Ph.D., full professor
	Member, Mentor:	Ladislav Novak, Ph.D., full professor

Menthor's sign

Садржај

Листа слика

Листа табела

Скраћенице

ANN Artificial Neural Network.

ASIC Application Specific Integrated Circuits.

ASIP Application Specific Integrated Processor.

CART Classification and Regression Tree.

CMOS Complementary metal–oxide–semiconductor.

CPU Central Processing Unit.

CQP Constrained Quadratic Programming.

DSP Digital Signal Processor.

DT Decision Trees.

FPGA Field Programmable Gate Array.

FSM Finite State Machine.

FT Functiona Tree.

HPC High Performance Computing.

IP Intellectual Property.

LSB Least Significant Bit.

MLP Multi-Layer Perceptron.

MSB Most Significant Bit.

PE Processing Element.

RB Reconfigurable Block.

REC Reconfigurable Ensemble Classifier.

REN Reconfiguration Enable.

RI Reconfigurable Interconnection.

RMLC Reconfigurable Machine Learning Classifier.

RPE Reconfigurable Processing Element.

RTL Register Transfer Level.

SIMD Single Instruction, Multiple Data.

SMO Sequential Minimal Optimisation.

SoC System on Chip.

SVM Support Vector Machine.

TSMC Taiwan Semiconductor Manufacturing Company.

WEKA Waikato Environment for Knowledge Analysis.

Глава 1

Увод

У овој дисертацији приказане су две дигиталне реконфигурабилне архитектуре грубог степена гранулације које могу да се примене за имплементацију машинских предиктивних модела. Архитектура Reconfigurable Machine Learning Classifier (RMLC) састоји се од већег броја истоветних блокова и може се конфигурисати да имплементира ортогонална стабла одлука - Decision Trees (DT), неортогоналне DT-ове, нелинеарне DT-ове, функционалне DT-ове, регресионе DT-ове, линеарне и нелинеарне Support Vector Machine (SVM) предиктивне моделе са полиномијалним и радијалним кернелима, вишеслојне перцептроне и Artificial Neural Network (ANN) класификаторе засноване на радијалним функцијама. Архитектура Reconfigurable Ensemble Classifier (REC) се може конфигурисати да ради као ансамбл претходно наведених машинских класификатора или може да имплементира већи број оваквих класификатора, истовремено.

1.1 Мотивација

Машинско учење (Machine Learning) [1] [2] је област вештачке интелигенције која се бави анализом и синтезом вештачких система који могу да уче из података. Машинско учење се још може представити и као скуп метода за конструкцију математичких модела који могу аутоматски да препознају шаблоне у подацима. Ови математички модели се називају и предиктивни модели. Циљ синтезе ових модела је превиђање нових података и доношење одлука. Најчешће коришћени предиктивни модели су вештачке неуронске мреже - ANN [3], стабла одлука - DT [4] [5] и SVM [6]. У области рударења података (Data Mining) [7], тражења шаблона у великој количини података, методе DT и SVM су међу првих

10 коришћених данас [8] [9]. Међу истраживачима у рударењу података и ANN се често користе [7] [10].

Када се машински класификатори користе за решавање проблема велике величине (нпр. у рударењу података) или у апликацијама обраде података у реалном времену (нпр. компјутерска визија) [11] [12] [13] [14], време за које се обавља класификација је веома критична ставка. Скраћивање времена класификовања може се постићи развојем нових алгоритама и софтверских алата или имплементирањем класификатора директно у хардверу.

Алгоритамска решења за проблеме рударења података огромне величине постоје већ деценијама. Међутим, њихова примена је ограничена на специјализоване, високо буџетне системе. Неки од таквих система су финансијске и високо технолошке индустријске апликације. За овакве случајеве предложен је велики број алгоритамских решења [15] [16] . Компаније као што су Google, Yahoo, Facebook, LinkedIn, Twitter и eBay раде са "Hadoop" софтверским системом намењеног дистрибуираном рачунању. "Hadoop" омогућава ефикасно обрађивање велике количине података коришћењем великих мултипроцесорских система, процесорских фарми и читавих мрежа рачунара [17]. Компанија IBM у својој понуди има SPSS софтвер намењен за рад са подацима великог обима. Часописи "Wiley Interdisciplinary Review" и "Rexer Analytics" наводе као најчешће коришћене софтверске алате, у пројектима рударења података, RapidMiner, R project и Pentaho/WEKA платформу. Ови алати, између осталих, нуде велики број алгоритама за рад са машинским класификаторима као што су DT-ови, SVM-ови и ANN-ови.

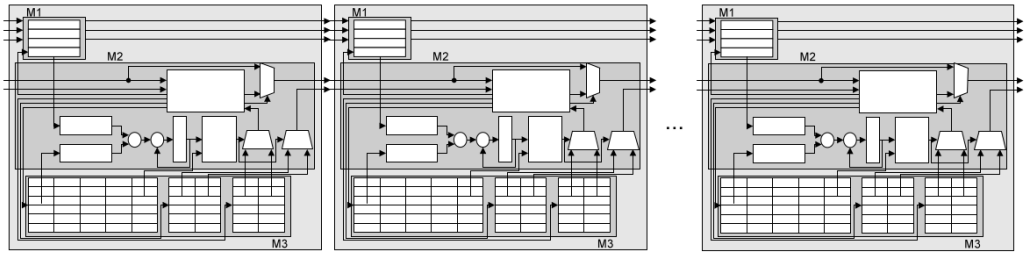
1.2 Преглед постојећих резултата

У области хардверских имплементација класификатора машинског учења постоји доста објављених радова и предложених архитектура. Велика пажња посвећена је стаблима одлука, SVM предиктивним моделима као и неуронским мрежама.

Стабла одлука су машински класификатори са веома једноставном идејом. Иако су примењују веома много, прошло је доста времена док се нису појавиле прве хардверске архитектуре за ову врсту класификатора. Хардверска имплементација алгоритма за изградњу стабала одлука дата је у [18]. Предложена архитектура погодна је за релизацију како на Field Programmable Gate Array (FPGA), тако и на Application Specific Integrated Circuits (ASIC) чиповима. Имплементација познатог Classifica-

tion and Regression Tree (CART) алгоритма за изградњу стабала одлука описана је у [19]. Приказано решење користи FPGA чип као и Central Processing Unit (CPU) са више језгара да постигне значајно убрзање у односу на софтверска решења. Приказано решење намењено је High Performance Computing (HPC) системима. Реализација ортогоналног стабла одлуке погодног за системе ниске потрошње предложена је у [20]. Систем је имплементиран као ASIC чип урађен у 0.18 µm Complementary metal–oxide–semiconductor (CMOS) технологији и употребљен је за идентификацију гасова. Још једна имплементација ортогоналног стабла одлука приказана је у [21]. Предложена паралелна проточна архитектура имплементирана је на FPGA чипу.

Решење за DT предиктивне моделе са најширом применом предложено је у [22] и [23]. Приказано Intellectual Property (IP) језгро може да реализује ортогонална, неортогонална и нелинеарна стабла одлука и погодно је за интеграцију у System on Chip (SoC) решења. Архитектура се састоји од низа универзалних чворова уз помоћ којих могу да се имплементирају све три наведене врсте стабала. При томе, предложене архитектуре су погодне за имплементацију како на FPGA тако и на ASIC чипу. На слици 1.1 приказана је архитектура која се састоји од секвенце универзалних чворова.



Слика 1.1: Архитектура састављена од низа универзалних чворова (Преузето из [22])

Архитектура, чији циљ је да се имплементирају хомогени ансамбли који се састоје од DT предиктивних модела предложена је у [21].

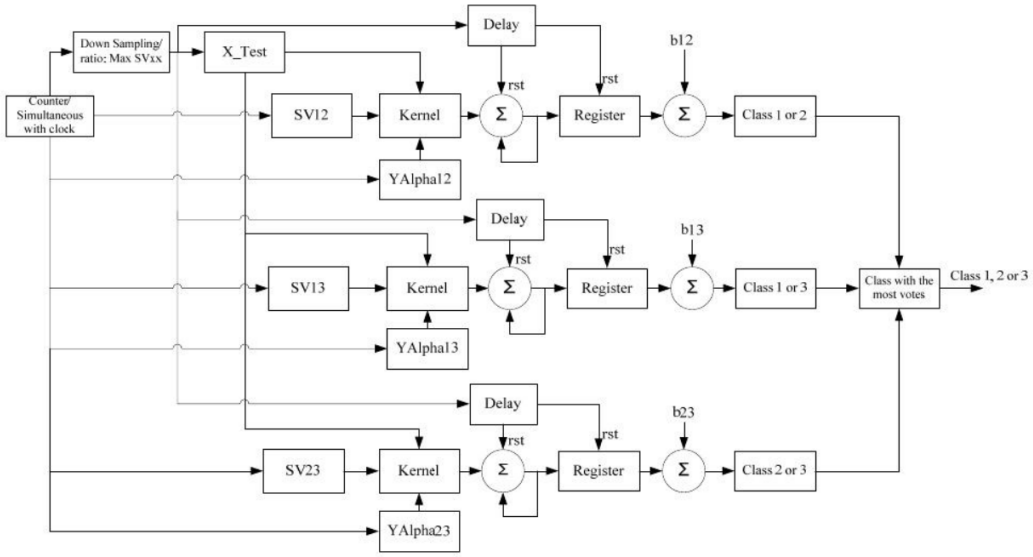
SVM предиктивни модели су се показали као веома добри у практичној примени али је њихово срачунавање доста комплексно. Из тог разлога су хардверске архитектуре предложене убрзо након првог приказивања ових класификатора у [6]. Прва хардверска архитектура за тренирање SVM класификатора предложена је у [24]. У том раду развијен је посебан алгоритам, погодан за паралелну хардверску имплементацију и описан је његов Register Transfer Level

(RTL) модел. На жалост, предложени алгоритам има експоненцијалну комплексност. Дигиталне архитектуре за убрзање најефикаснијег алгоритма за тренирање SVM класификатора предложене су у [25, 26]. Решење приказано у [26] имплементира чувени Sequential Minimal Optimisation (SMO) алгоритам и реализовано је као SoC решење у 90 nm Taiwan Semiconductor Manufacturing Company (TSMC) технологији. Решење приказано у [25] реализовано је као IP језгро и релизован је FPGA прототип.

Ако се посматрају само класификатори, један генератор SVM предиктивних модела предложен је у [27]. Предложени генератор прави модел који је погодан за FPGA имплементацију и примењује се у уграђеним системима. Једна паралелна дигиталана архитектура за ову врсту класификатора предложена је у [28] и примењена је на детекцију објеката. Архитектура користи дељење ресурса између паралелних јединица и имплементирана је на FPGA чипу. Једна архитектура која може да ради са SVM предиктивним моделима чији је кернел радијалан предложена је у [29].

SVM класификатори се обично имплементирају као бинарни класификатори. SVM предиктивни модел који може да предвиђа више од две резултујуће класе предложен је у [30]. На слици 1.2 приказана је архитектура предложеног класификатора. Може се видети да је предикција на више класа постигнута паралелном имплементацијом више бинарних класификатора. Коначна класа се одређује простим гласањем што је једна од идеја која се користи код ансамбала класификатора.

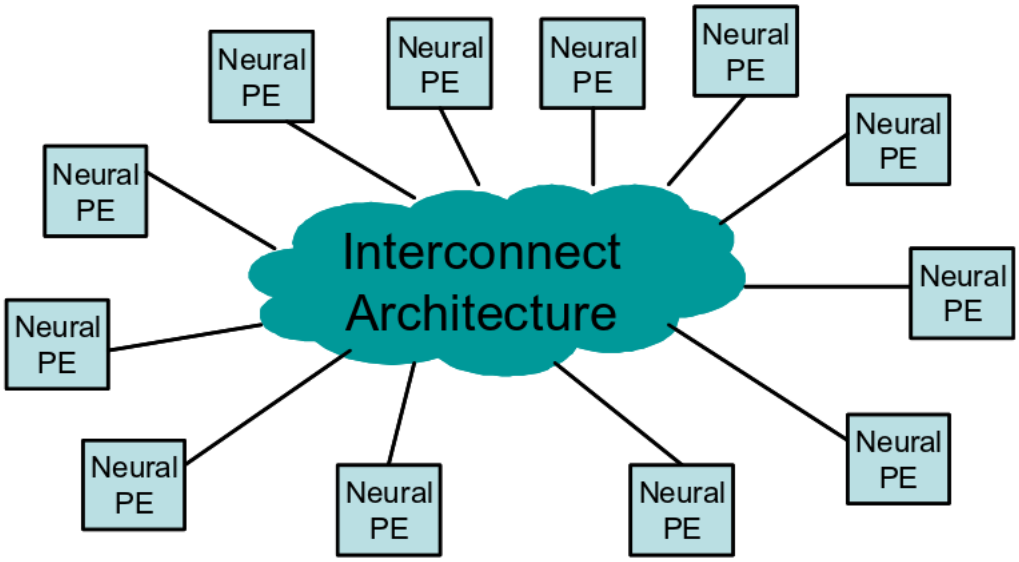
ANN предиктивни модели имају структуру као неуронске мреже биолошких система и најпознатији су међу класификаторима. Хардверска решења за ову врсту класификатора у употреби су преко 20 година. Преглед постојећих решења до 2010. године дат је у [31]. У тим решењима за имплементацију неуронских мрежа коришћена су аналогна, дигитална као и хибридна решења. Дигиталне неуронске мреже имплементиране су како на FPGA тако и на ASIC чипу. Од архитектура коришћене су, паралелне као и Single Instruction, Multiple Data (SIMD) архитектуре. Више књига се бави хардверском имплементацијом неуронских мрежа међу којима је нпр. [32]. У књизи се описују архитектуре неуронских мрежа са посебним нагласком на реализацији на FPGA чиповима. Тврди се да су хардверске имплементације неуронских мрежа постале поново атрактивне захваљујући повећању капацитета FPGA чипова и да су почетни неуспеси хардверског приступа везани са покушајима да се мреже имплементирају искључиво коришћењем ASIC чипова.



Слика 1.2: SVM архитектура за предикцију више од две класе (Преузето из [30])

Паралелна имплементација неуронске мреже намењене за обраду видео сигнала у реалном времену предложена је у [33]. Архитектура користи фиксну репрезентацију за представљање бројева ширине 16 бита, организована је по слојевима и имплементирана је на FPGA чипу. У чланку [34] приказана је архитектура за Multi-Layer Perceptron (MLP) неуронске мреже. Предложена архитектура реализована је на FPGA чипу, организована је по слојевима и користи паралелизам својствен неуронским мрежама. Другачији приступ реализацији ANN мрежа показан је у чланку [35] где је предложено идејно решење за имплементацију ANN класификатора као SoC архитектуре. Концепт овог идејног решења приказан је на слици 1.3. Предложена архитектура користи огроман број елемената који су названи неуронски Processing Element (PE). Ти елементи су преко једне мреже за међусобно повезивање организовани у јединствен систем. У истом чланку анализира се најекономичнији поступак добијања јединствене мреже са милионима неурона.

Као што се може видети из доступне литературе, хардверске имплементације предиктивних модела машинског учења су од истраживачког интереса више од 20 година. Предложен је веома велики број архитектура, са најразличитијим циљевима имплементације. За



Слика 1.3: Неуронска мрежа на SoC (Преузето из [35])

све архитектуре заједничко је да имају знатно краће време предикције у односу на софтверска решења, од 1 до 3 реда величине мања.

1.3 Главни резултати докторске дисертације

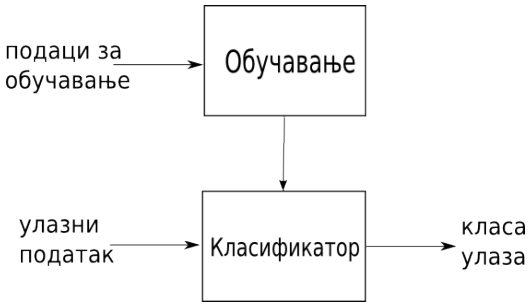
Све постојеће дигиталне архитектуре за хардверску имплементацију машинских класификатора могу да реализују само један тип класификатора, на пример DT, SVM или ANN. Обзиром на могуће компаративне предности коришћења хибридних модела који укључују различите типове класификатора, од интереса је дигитална архитектура која би могла да имплементира више различитих врста машинских класификатора. У овој докторској дисертацији представљена је једна таква универзална реконфигурабилна архитектура, грубог степена гранулације, која има могућност да имплементира више типова различитих машинских класификатора, укључујући и DT-ове, SVM-ове као и ANN-ове. Предложена хардверска дигитална архитектура може да имплементира ортогоналне DT-ове, неортогоналне DT-ове, нелинеарне DT-ове, функционалне DT-ове, регресионе DT-ове, линеарне и нелинеарне SVM-ове са полиномијалним и радијалним кернелима, вишеслојне перцептроне (врста ANN) и ANN-ове засноване

на радијалним функцијама. Колико је нама познато ово је прва хардверска архитектура која има овакве могућности.

1.4 Три најчешће коришћена предиктивна модела

Машинско учење је научна област која анализира и развија алгоритме, чијим извршавањем се омогућава машинско учење. Циљ је да се изграде системи, који само на основу улазних података, могу да доносе интелигентне одлуке. Област машинског учења је у тесној вези са областима као што су теорија вероватноће, вештачка интелигенција, итд.

Системи машинског учења имају два дела (Слика 1.4). Први део служи за формирање предиктивних модела на основу података за тренирање коришћењем неког од алгоритама за обучавање. Други део обухвата употребу претходно изграђеног предиктивног модела који се може интерпретирати као знање или интелигенција.



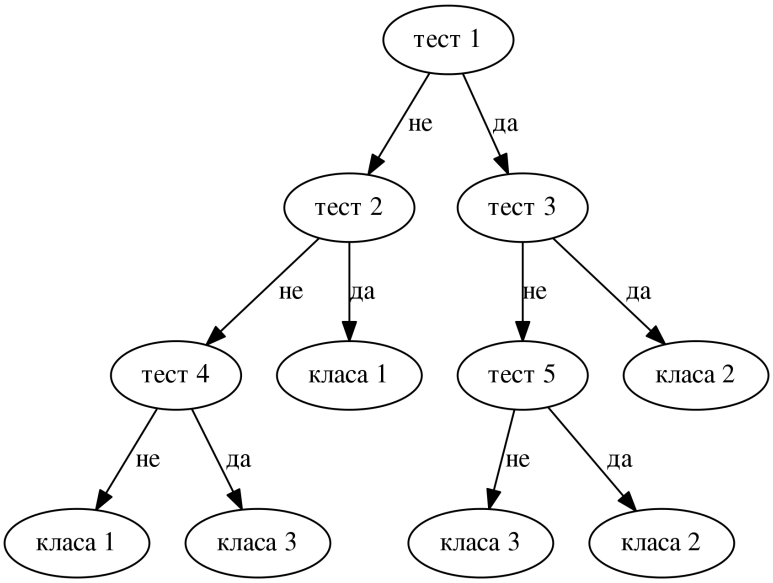
Слика 1.4: Систем машинског учења

Постоји велики број алгоритама машинског учења. Три главне групе су неуронске мреже, стабла одлука и Support Vector Machine.

1.4.1 Стабла одлука

Стабла одлука (DT) су једни од првих машинских модела учења који су ушли у ширу примену. DT додељују неку класу улазној инстанци на основу серије тестова у чворовиима стабла (Слика 1.5). На основу теста у чвору стабла бира се једна од две или више могућих грана и тим избором преноси се даље тест у наредни суседни чвор. У овој дисертацији разматрају се само стабла код којих су могуће две наредне гране. Овакви тестови понављају се док се не наиђе до листа стабла.

Лист стабла заправо представља резултат предикције који се додељује улазној инстанци. На овај начин, процес класификовања своди се на обилазак стабла одлуке.



Слика 1.5: Стабло одлуке

У сваком од чворова стабла, више атрибута улазне инстанце може бити подвргнуто тесту. Уколико су атрибути нумерички, у зависности од врсте теста, стабла могу бити ортогонална, неортогонална или нелинеарна [21].

У случају неортогоналних стабала, тест у чворовима стабла проверава да ли је вредност функције 1.1 позитивна или не.

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b \tag{1.1}$$

У формули 1.1 вектор $\mathbf{w}$ представља вектор нормалне раздвајајуће хиперравни у простору атрибута, $\mathbf{x}$ је улазни вектор (инстанца) док је b скалар.

У зависности од тога шта се налази у листу стабла предиктивни модели стабала могу се поделити на две класе:

- Класификациона стабла, која у својим листовима имају скаларне вредности које представљају класу која се додељује инстанци из дискретног скупа вредности.
- Регресиона стабла, која у својим листовима стабла процењују реалну вредност као резултат предикције.

У употреби су још и функционална стабла - Functiona Tree (FT) уведена у [36].

1.4.2 Support Vector Machines

SVM су алгоритми за машинско учење који су засновани на статистичкој теорији учења [37]. У литератури се први пут појављују у [6]. Такви алгоритми постижу одличне резултате јер имају две добре особине. Прва особина је та што увек налазе глобални минимум циљне функције класификације а друга је да постижу максималну маргину класификације. Ове особине ће даље бити објашњене у наредним параграфима.

Обично се о SVM машинама говори као о бинарним класификаторима. Уз одређене модификације ове машине могу давати и континуалан излаз и тада се такве машине називају регресионе SVM машине. У наставку ће прво бити укратко описане бинарне а затим регресионе машине.

SVM алгоритми апроксимирају непознату функцију класификације $T : X \to Y$ са неком функцијом $F : X \to Y$. Улазни подаци за алгоритам обучавања SVM-а је тренинг скуп величине n:

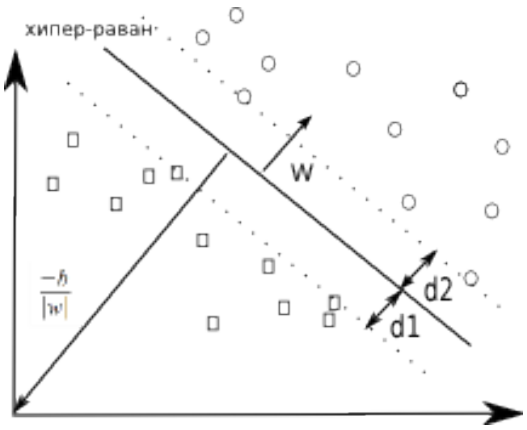
$$S = \{x_i, y_i\}, i = 1, \ldots, n, x_i \in X \subseteq \mathbb{R}^n, y_i \in Y = \{-1, 1\}, T : x_i \to y_i$$

Линеарна верзија функције класификације је облика

$$F(x) = w \cdot x + b, w \in \mathbb{R}^n, b \in \mathbb{R}$$

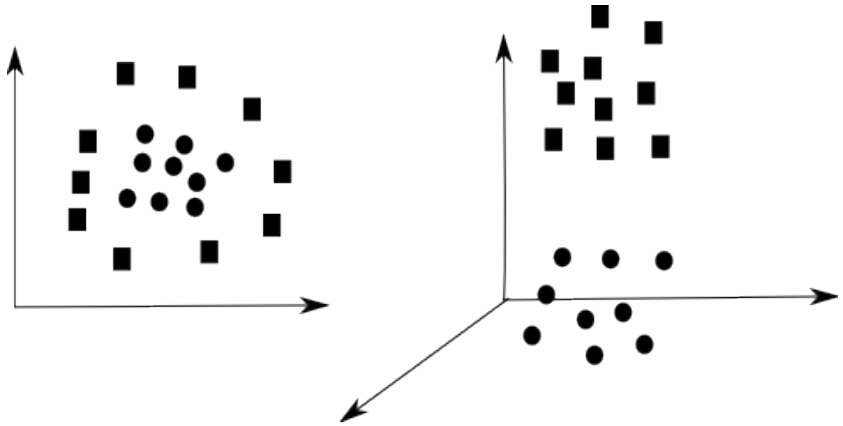
Линеарна функција класификације представља хиперраван (Слика 1.6). Са једне "стране" равни налазе се тачке једне класе а са супротне стране налазе се тачке друге класе. SVM алгоритми налазе хиперраван која је максимално удаљена од тачака обе класе. То је особина максималне маргине. Ова се обично сматра одличном особином пошто повећава моћ генерализације машине. Моћ генерализације је особина којом се на основу једног броја случајева може направити правило које покрива све или већи број случајева.

Улазни простор X може се нелинеарним мапирањем ϕ превести у простор $Z \subseteq \mathbb{R}^l, l \gg n$, чиме се уводи нелинеарна функција класификације. То омогућава да се подаци које није било могуће раздвојити у простору X, уколико се примени одговарајуће пресликавање, раздвоје у простору Z који има више димензија од простора X. Укупни резултат је да се добија нелинеарна раздвајајућа



Слика 1.6: Хиперраван

површ (Слика 1.7). Коришћењем "трика језгра" (Kernel Trick), избегава се рад са нелинеарним функцијама и задржава се сва једноставност линеарног случаја. Функција класификације се у том случају изражава као $F(x) = w \cdot \phi(x) + b$.



Слика 1.7: Трик језгра

У принципу, потпуно раздвајање тачака није могуће ни у просторима са великим димензијама што значи да неке тачке могу бити погрешно класификоване. Због тога се уводи околина хиперравни (описује се ϵ параметрима) у којој је допуштена погрешна класификација. Када се ови пареметри уведу у већ постојеће једначине, SVM је могуће представити у форми ограниченог квадратног програмирања - Constrained Quadratic Programming (CQP).

$$\min_{w,b}[\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\epsilon_i]$$
$$y_i(w\cdot\phi(x_i)+b) \geq (1-\epsilon_i)$$
$$C>0, \epsilon_i \geq 0, \forall i \in 1,\ldots,m$$

Први део функције за минимизацију обезбеђује највећу маргину а други одређује колико тачке могу бити погрешно класификоване. Параметар C балансира оба циља. Применом Лагранжове теорије множитеља, може се добити дуал овог проблема који је једноставнији за решавање.

$$\min_{\alpha}[\frac{1}{2}\alpha^T Q\alpha + r^T\alpha]$$
$$0 \leq \alpha_i \leq C, r_i = -1 \forall i \in 1,\ldots,m,$$
$$y^T\alpha = 0$$

Матрица Q је симетрична, позитивна, семи-дефинитна матрица, величине $m \times m$ са елементима

$$q_{ij} = y_i y_j K(x_i, x_j)$$

Функција K се зове језгро (kernel) и она имплицитно дефинише пресликавање ϕ , $K(x_i, x_j) = \phi(x_i)\cdot\phi(y_i)$. Избором језгра бирамо простор у који ће тачке бити пресликане. Нека од популарних језгара су радијално и полиномијално језгро:

$$k_{radial}(x_i, x_j) = e^{-(\frac{\|x_i - x_j\|^2}{2\sigma^2})}$$
$$k_{poly}(x_i, x_j) = (x_i \cdot x_j + a)^b$$

У случају SVM машина, систем за обучавање најчешће решава дуални Лангранжов CQP проблем. Резултат обучавања је SVM класификатор који срачунава следећи израз:

$$v(s) = \sum_{i=1}^{m} y_i\alpha_i K(x_i, s) + b \tag{1.2}$$

Улазни податак означен је са s . Знак $v(s)$ је класа улазног примера и на тај начин добија се бинарна класификација.

Код машина за регресију излаз је континуална вредност, $y_i \in Y \subseteq \mathbb{R}$. Ове машине предвиђају колики ће бити излаз непознате реалне функције на произвољан улаз.

Машине за регресију користе сложенију функцију за оцену колико је тачака погрешно класификовано. Уколико је предвиђена вредност на раздаљини мањој од неког ϵ тада се грешка игнорише. На тај начин се добија "ϵ неосетљива цев". Слично, као за бинарне машине добија се проблем квадратног програмирања:

$$\begin{aligned} \min_{w,b}[\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i^+ + \xi_i^-)] \\ t_i \leq y_i + \epsilon + \xi_i^+ \\ t_i \geq y_i - \epsilon - \xi_i^- \end{aligned}$$

Вредности t_i су тачне вредности за тренинг пример i а ξ_i^+ и ξ_i^- су вредности које одређују колика је допуштена грешка за дати пример. Применом Лагранжове теорије множитеља, добија се следећи дуални проблем:

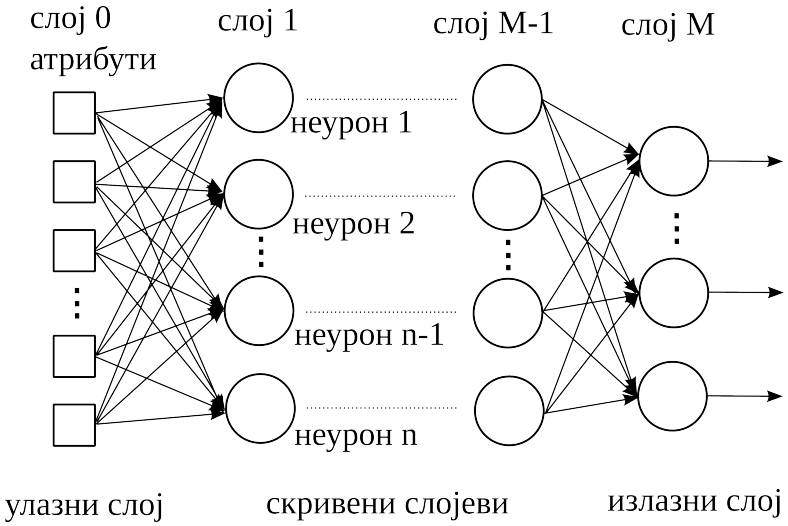
$$\begin{aligned} \min_{\alpha_i^+,\alpha_i^-}[\frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^+)x_i \cdot x_j + \epsilon\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) + \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)t_i] \\ \forall i, 0 \leq \alpha_i^+ \leq C, 0 \leq \alpha_i^- \leq C \\ \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) = 0 \end{aligned}$$

1.4.3 Вештачке неуронске мреже

Вештачке неуронске мреже су машински модели за предикцију чија је структура слична неуронским структурама живих бића. На језику графова ANN-ови се могу интерпретирати као тежински графови где су вештачки неурони као чворови графа повезани тежинским гранама. У зависности од тога да ли је граф мреже цикличан или не, ANN се могу поделити на рекурзивне ANN-ове (енглески recurrent ANN) и ANN-ове без повратних грана (енглески feed-forward ANN). У овој дисертацији неће бити речи о рекурзивним ANN-овима.

Два најчешћа коришћена типа мрежа су MLP мреже и RB (Radial Basis) мреже. У случају MLP мрежа, који су широко коришћена врста неуронских мрежа, неурони су организовани у слојеве. Једини

допуштени спојеви међу неуронима су они између суседних слојева мреже. Не постоје ни спојеви између неурона истог слоја, као ни повратни спојеви од слоја M ка слојевима M-1, M-2... (Слика 1.8).



Слика 1.8: Структура неуронске MLP мреже

У структури MLP ANN предиктивног модела, постоје три врсте слојева: улазни, скривени и излазни слој. Улазни слој састоји се од n неурона где је n број атрибута улазне инстанце. Након улазног слоја следи један или више скривених слојева. Сваки скривени слој састоји се од произвољног броја неурона. Излазни слој мреже израчунава коначну вредност класификовања за улазну инстанцу.

Сваки неурон, било у скривеном, било у излазном слоју, рачуна следећи израз:

$$f(\mathbf{w} \cdot \mathbf{x} + b) \tag{1.3}$$

где је $\mathbf{x}$ је улазни вектор, вектор $\mathbf{w}$ представља вектор тежине док је b скалар, који се још назива и додатна вредност. Функција f, која се назива и активациона функција, може бити линеарна или нелинеарна реална функција реалне скаларне променљиве: $f : \mathbb{R} \to \mathbb{R}$. За неуроне улазног слоја вектор $\mathbf{x}$ заправо је вектор атрибута улазне инстанце. За неуроне скривених слојева и излазног слоја вектор $\mathbf{x}$ се састоји од резултата прорачуна претходних слојева мреже. Вектор $\mathbf{w}$ има исте димензије као и вектор $\mathbf{x}$.

У употреби MLP ANN класификатора, коришћене су многе активационе функције. Неке од њих су хиперболични тангенс и сигмоид функција 1.4.

$$f_{tanh}(x) = tanh(x)$$
$$f_{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{1.4}$$

Други, веома важан, тип неуронских мрежа су RB ANN мреже. Оне се типично састоје од три слоја неурона: улазни, скривени и излазни. Сваки неурон у скривеном слоју рачуна следећи израз:

$$f\left(\|\mathbf{x}-\mathbf{w}\|\right) \tag{1.5}$$

где је $\mathbf{x}$ је улазни вектор (инстанца), вектор $\mathbf{w}$ представља централни вектор док је функција f нека радијална функција, на пример:

$$f_{gaussian}(\|\mathbf{x}-\mathbf{w}\|) = e^{-\frac{\|\mathbf{x}-\mathbf{w}\|^2}{2\sigma^2}} \tag{1.6}$$

Неурон из излазног слоја рачуна:

$$f(x) = \mathbf{x} \cdot \mathbf{w} \tag{1.7}$$

где је $\mathbf{x}$ вектор који се састоји од резултата прорачуна скривеног слоја, а $\mathbf{w}$ је тежински вектор излазног неурона, док је $\cdot$ скаларни производ. Вектор $\mathbf{x}$ има димензију једнаку броју неурона у скривеном слоју мреже. Вектори $\mathbf{x}$ у формулама 1.5 и 1.7, у општем случају имају различите димензије. Исто важи и за векторе $\mathbf{w}$ из истих формула.

1.5 Реконфигурабилне хардверске архитектуре

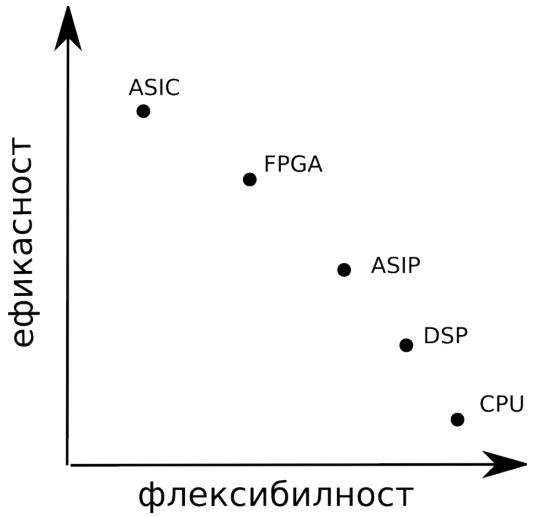
Област реконфигурабилног хардвера је у великом развоју. Објављено је више књига и радова на ову тему, [38, 39, 40]. У овом поглављу биће дат мали увод који ће омогућити да се опише каквог типа су реконфигурабилни системи представљене у овој дисертацији.

Обично се сматра да се неки алгоритам може имплементирати на два начина. Први начин, широко распрострањен, је реализација алгоритма као софтвера који се извршава на неком процесору опште намене - CPU. Други начин је имплементација алгоритма као специјализованог хардвера у виду ASIC чипа. Софтверка имплементација пружа велику флексибилност приликом реализације алгоритма али су перформансе такве релизације вишеструко лошије од ASIC имплементације.

Реконфигурабилна хардверска решења пружају флексибилност која је приближна софтверским решењима, уз перформансе које се могу

мерити са ASIC решењима. У поређењу са CPU имплементацијом, реконфигурабилна решења пружају перформасе које су неколико редова величине боље али је сам развој значајно комплекснији. Када се реконфигурабилна решења пореде са ASIC имплементацијом, перформансе су лошије ред величине али је развој драстично поједностављен.

Уколико се још дубље уђе у разматрање компромиса између ефикасности система и флексибилности његовог развоја може се видети да нису само CPU и ASIC имплементације једина решења која имају велику примену. Уз ове две могућности за релизацију алгоритама користе се још у значајној мери и FPGA, Application Specific Integrated Processor (ASIP) и Digital Signal Processor (DSP) решења (Слика 1.9). FPGA системи су пример реконфигурабилног система са великом флексибилности. Реконфигурабилни системи развијени специјално за неки проблем могу се поставити између ASIC и FPGA решења, што се тиче перформанси и флексибилности.



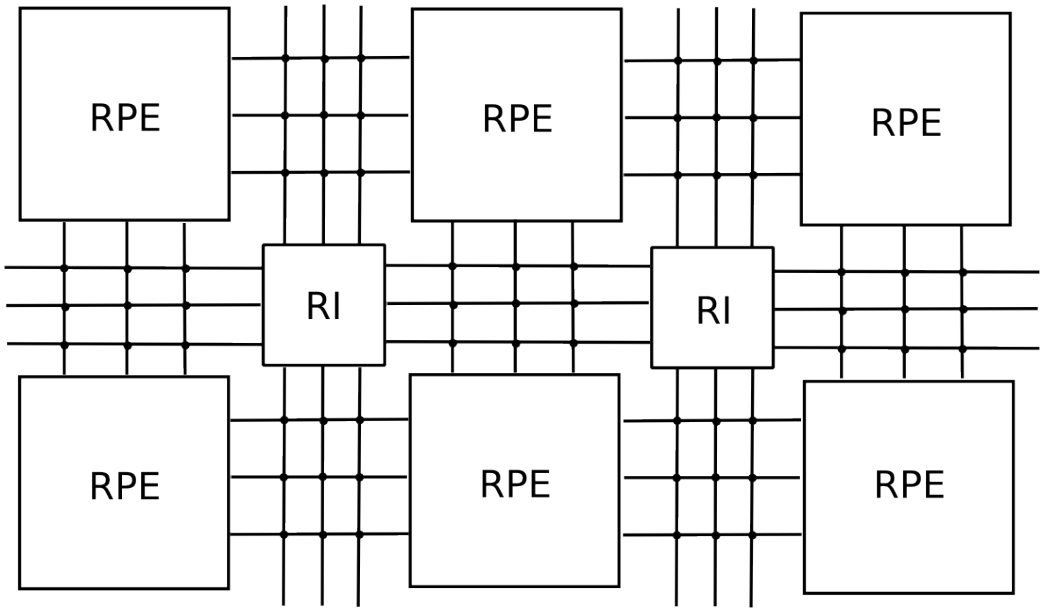
Слика 1.9: Компромис између ефикасности и флексибилности

Реконфигурабилни систем састоји се од већег броја програмабилних елемената за процесирање - Reconfigurable Processing Element (RPE) и програмабилних интерконекција - Reconfigurable Interconnection (RI). RPE се по потреби могу конфигурисати да реализују одговарајућу логичку операцију док се RI могу конфигурисати да образују међусобну везу два RPE. Имплементација потребног алгоритма постиже се тако што се упишу одговарајуће вредности у конфигурационе бите који одређују како ће се понашати RPE и RI елементи.

На основу тога да ли се реконфигурациони систем конфигурише на нивоу бита или на нивоу речи, може се направити следећа класификација ових система:

- Реконфигурабилни системи финог степена гранулације
- Реконфигурабилни системи грубог степена гранулације

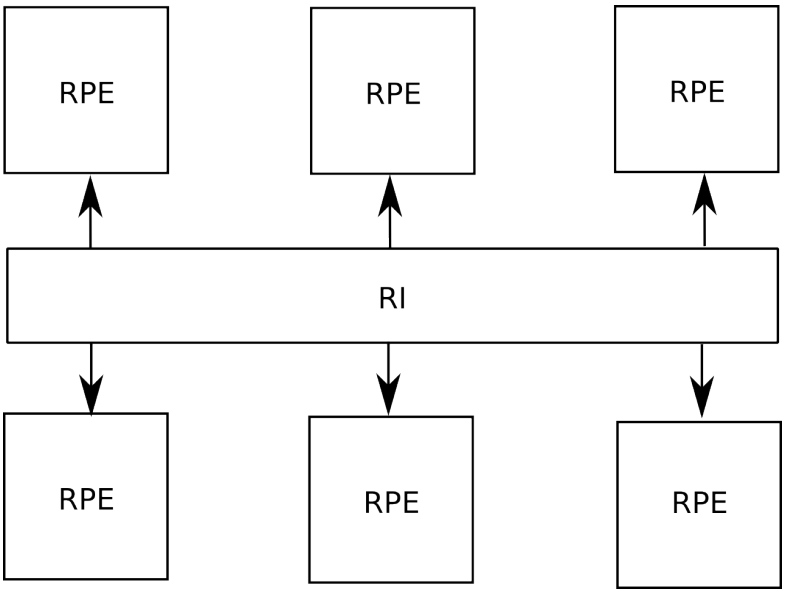
Реконфигурабилни системи финог степена гранулације су они код којих се и RPE и RI елементи могу конфигурисати на нивоу појединачних бита. Пример система финог степена гранулације је FPGA чип. На слици 1.10 може се видети генерална структура једног реконфигурабилног система финог степена гранулације. Сваки RPE елемент може се конфигурисати тако да имплементира произвољну једнобитну Булову функцију. Уопштено, RPE може имплементирати више различитих Булових функција одједном. RI елементи омогућавају да се произвољни улази/илази RPE елемената повежу на нивоу бита и број ових веза је веома велик.



Слика 1.10: Генерални приказ реконфигурабилног система финог степена гранулације

Реконфигурабилни системи грубог степена гранулације су системи чији RPE и RI елементи се конфигуришу на нивоу речи. Ови системи

се пројектују циљано за одређени домен проблема. На слици 1.11 се може видети генерална структура једног реконфигурабилног система грубог степена гранулације. У случају ових система, RPE елементи обављају задатак специјализован за домен проблема. У овом случају они се конфигуришу целим речима и не могу да обављају произвољну функцију. Конфигурација се за RPE елементе у овом случају смешта у RAM меморије. Број RI елемената знатно је мањи него у случају система финог степена гранулације. Код система грубог степена гранулације ови елементи конфигуришу целе магистрале. Зато што су системи грубог степена гранулације орјентисани ка само једном домену проблема, елементи за конфигурисање немају велику флексибилност као елементи система финог степена гранулације. Мањи степен флексибилности омогућава овим системима да имају перформансе приближне ASIC решењима.



Слика 1.11: Генерални приказ реконфигурабилног система грубог степена гранулације

Према овој класификацији, архитектуре представљене у овој дисертацији су реконфигурабилни системи грубог степена гранулације примењени на домен проблема машинског учења.

Глава 2

Универзална реконфигурабилна архитектура за убрзавање три предиктивна модела

2.1 Универзални математички модел за три предиктивна модела

У овом поглављу детаљно ће бити описана универзална реконфигурабилна дигитална архитектура која може да реализује DT, SVM и ANN класификаторе. У наставку текста ову архитектуру ћемо звати RMLC. Главна идеја за реализацију оваквог класификатора је чињеница да формуле 1.1, 1.2, 1.3 и 1.5 имају неколико заједничких операција. За DT моделе, SVM моделе који имају кернел у којем се користи скаларни производ и за MLP моделе, основна операција је скаларни производ вектора. Уједно ово је и операција која је рачунски најзахтевнија а у случају DT-ова и линеарних SVM-ова ово је и једина операција која треба да се уради. У случају формула за нелинеарне SVM-ове и MLP-ове на резултат скаларног производа се примењује још и додатна нелинеарна функција. За два модела, SVM које користе радијални кернел и RB мреже, основна операција је рачунање модула разлике вектора. Све наведене формуле могу да се представе комплексном формулом 2.1.

$$cls(\mathbf{x}, cfg) = \begin{cases} cfg = 0 : \mathbf{w} \cdot \mathbf{x} + b \\ cfg = 1 : f(\mathbf{w} \cdot \mathbf{x} + b) \\ cfg = 2 : \sum_w \alpha f(\mathbf{w} \cdot \mathbf{x}) + b \\ cfg = 3 : \sum_w \alpha f(\|\mathbf{w} - \mathbf{x}\|) + b \\ cfg = 4 : f(\mathbf{w} \cdot \mathbf{x} + b) \\ cfg = 5 : f(\|\mathbf{w} - \mathbf{x}\|) \end{cases} \tag{2.1}$$

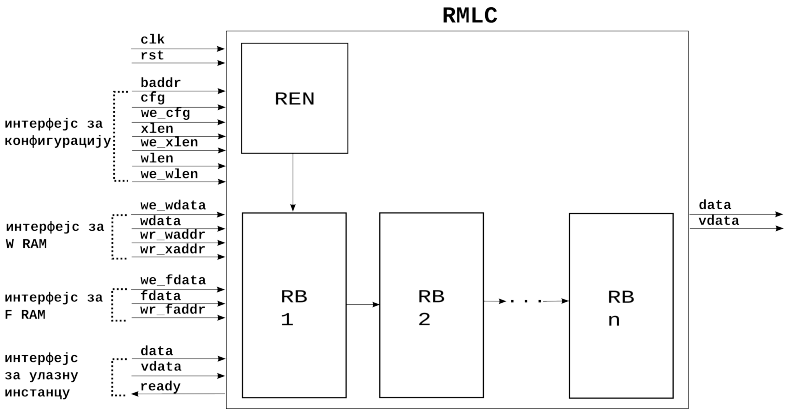
Када се анализира формула 2.1 може се закључити да сви случајеви, унутар комплексне формуле, деле сличне операције. Ова чињеница открива да је могуће дизајнирати јединствено реконфигурабилно језгро које би могло да реализује поменуте предиктивне моделе. Ово језгро треба да има могућност да ради одузимање два вектора, скаларни производ два вектора, додавање броја и рачунање произвољне нелинеарне функције. По потреби би се користиле само неке од свих ових операција уз одговарајући редослед њиховог извршавања. У фази реконфигурисања универзална дигитална архитектура би се програмирала да имплементира један тачно одређен тип класификатора. У наредним поглављима је представљена једна имплементација таквог реконфигурабилног дигиталног хардверског језгра.

2.2 Детаљи RMLC архитектуре

RMLC архитектура (Слика 2.1) састоји се од везе истоветних блокова поређаних у низ. Ове блокове ћемо звати Reconfigurable Block (RB) модули. Број ових блокова је параметар архитектуре и може се подесити током дизајнирања система. Сваки RB модул прима податке од претходног модула, потом ради прорачун над тим подацима и прослеђује их до наредног модула. Поред RB модула, RMLC архитектура има и један додатни модул који се зове Reconfiguration Enable (REN). Модул REN бира, током програмирања система, који појединачни RB модул се конфигурише.

Архитектура RMLC има неколико улазних интерфејса: интерфејс за конфигурацију, интерфејс за W RAM, интерфејс за F RAM и интерфејс за улзану инстанцу.

Интерфејс за конфигурацију служи да програмира функцију појединачних RB модула, одређује величину улазне инстанце и подешава број активних колона унутар W RAM-а. Интерфејси за W RAM и F RAM конфигуришу W RAM и F RAM меморије, респективно. О овим

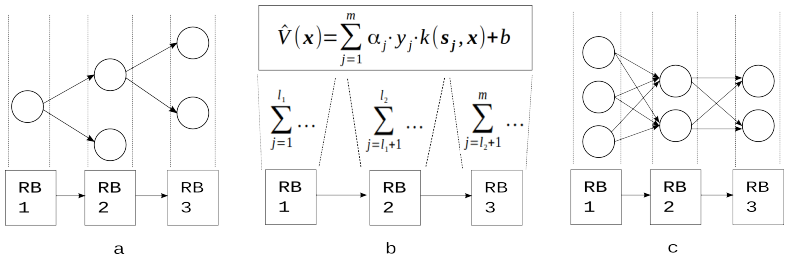


Слика 2.1: RMLC архитектура

меморијама биће више речи у наредним параграфима. Интерфејс за улазну инстанцу прослеђује улазну инстанцу до класификатора.

Модул RMLC има два излазна порта, data и vdata. Порт data садржи резултат предикције док порт vdata означава да ли је вредност на порту data важећа. Порт data се различито интерпретира у зависности од тога који тип класификатора имплементира RMLC архитектура. У случају да је RMLC архитектура конфигурисана као DT, тада ће на излазном порту бити срачуната класа улазне инстанце. Када је архитектура RMLC конфигурисана да ради као SVM, тада се на излазу класификатора налази срачуната сума функције за предикцију 1.2. И на крају, уколико је RMLC архитектура конфигурисана да ради као ANN, на излазном порту ће бити излазне вредности за сваки неурон излазног слоја.

Начин мапирања сваког од типова класификатора илустрован је на наредној слици 2.2.



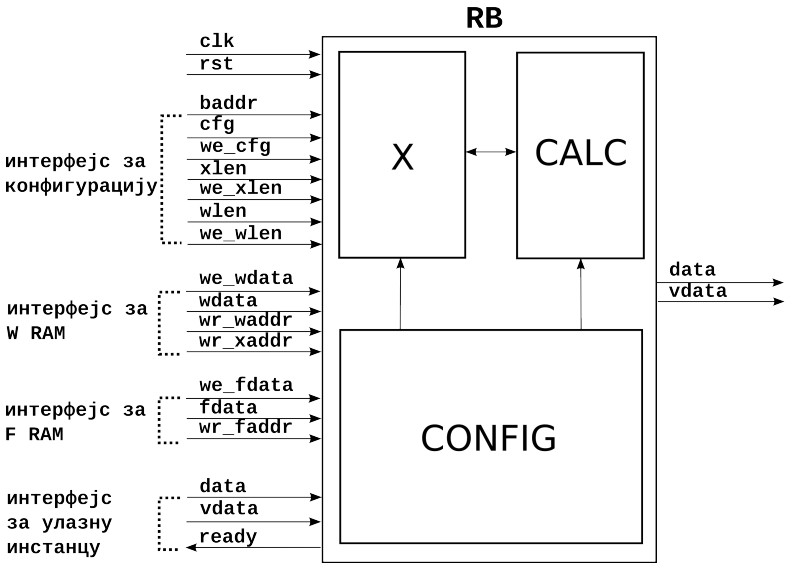
Слика 2.2: Мапирање на RMLC архитектуру

У случају када је RMLC архитектура конфигурисана као DT, сви чворови и листови у једном нивоу стабла су пресликани у један RB модул (2.2 а). Када је архитектура конфигурисана да ради као SVM, формула 1.2 се дели на делимичне суме. Свака од добијених делимичних сума

се потом срачунава у појединачном RB модулу (2.2 b). На крају, када је архитектура програмирана да ради као ANN, тада су сви неурони из једног нивоа мреже пресликани у један RB модул (2.2 c).

2.3 RB модул

Дизајн модула RB организован је хијерархијски. Сваки RB модул садржи по једну инстанцу следећа три модула (Слика 2.3): X, CALC, CONFIG.

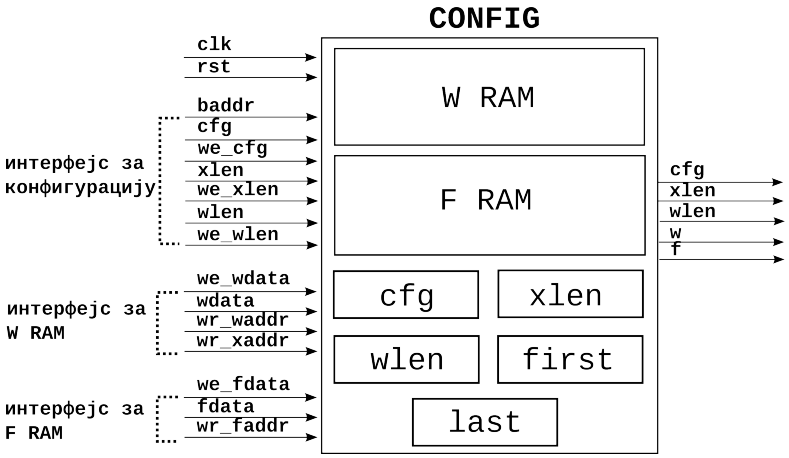


Слика 2.3: Структура RB модула

Модул X прима улазне вредности и смешта их у интерну RAM меморију. Ову меморију ћемо звати X RAM. У случају када је архитектура конфигурисана као DT или SVM, улазна инстанца се прослеђује наредним RB модулима у низу, који вредности инстанце смешта у X RAM. Када је архитектура програмирана да ради као ANN, тада се вредности излазних неурона тренутног RB модула смештају у X RAM наредног RB модула.

Модул CONFIG контролише начин рада RB модула. Модул CONFIG садржи неколико основних конфигурационих регистара (cfg, xlen, wlen, first, last, pass) и две RAM меморије (W RAM и F RAM). Конфигурациони регистар cfg одређује режим у ком ради RB модул. У зависности од тога која је вредност у овом регистру, добија се различит класификатор: DT, SVM или ANN. Регистар xlen одређује број атрибута у инстанцама. Вредност у регистру wlen одређује број вредности унутар

W RAM-а. Регистар first означава да ли је модул RB први у низу елемената, док регистар last означава да ли је модул RB последњи у низу. Регистар pass одређује да ли модул треба само да проследи резултат класификовања до наредног модула. Овај регистар има највећи приоритет. Уколико је он постављен на вредност један, вредности осталих регистара нису од значаја.



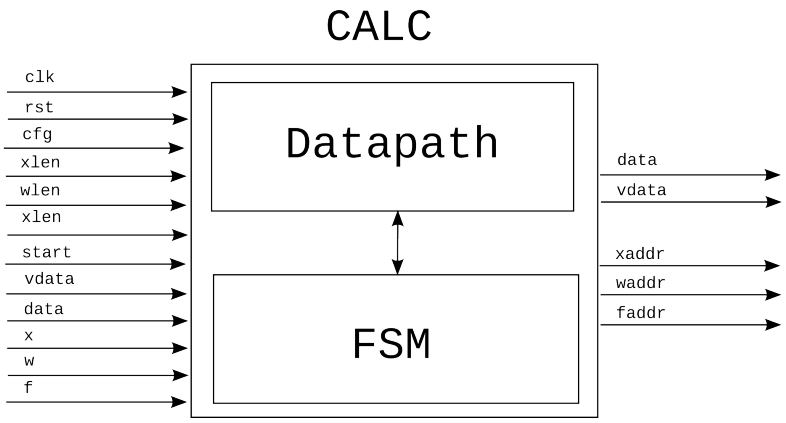
Слика 2.4: Структура CONFIG модула

RAM меморије унутар модула CONFIG садрже разне вредности у зависности од тога како је програмиран целокупан RB модул. Уопштено, W RAM је организован као дводимензионални низ који садржи вредности неопходне за рачунање скаларног производа. F RAM је једно-димензиони низ у коме се налазе одбирци функција.

W RAM је организован као дводимензионални низ, зато има два улазна порта за адресе (wr_waddr и wr_xaddr) чијим коришћењем се бира локација унутар меморије у коју ће се уписати вредност. Адресни улази за читање су одвојени (rd_waddr и rd_xaddr). Током адресирања вредности које се налазе на портовима *_waddr и *_xaddr (* се односи или на rd или на wr) се комбинују унутар модула у једну адресну вредност. Коначна адреса се добија тако што се *_xaddr интерпретира као мање значајна вредност док се *_waddr посматра као вредност већег значаја. На пример, када би обе адресне линије биле широке 4 бита, тада би се меморија W RAM адресирала са 8 бита. Уколико би вредност *_waddr била 0xA а вредност *_xaddr износила 0x5, тада би се меморија W RAM адресирала вредношћу 0xA5.

Модул CALC је најкомплекснији део модула RB. Он се састоји од аутомата стања - Finite State Machine (FSM), регистара и комбинационих мрежа. На слици 2.5 аутомат стања је означен са FSM док су

комбинациони део система као и регистри обележени са Datapath. Регистри и комбинационе мреже су такви да могу да обаве произвољан прорачун неопходан да се реализује функција неопходна за рад класификатора. FSM контролише ток података између регистара и комбинационих мрежа тако да се добије жељени резултат. Рад аутомата стања зависи од тренутне конфигурације класификатора.

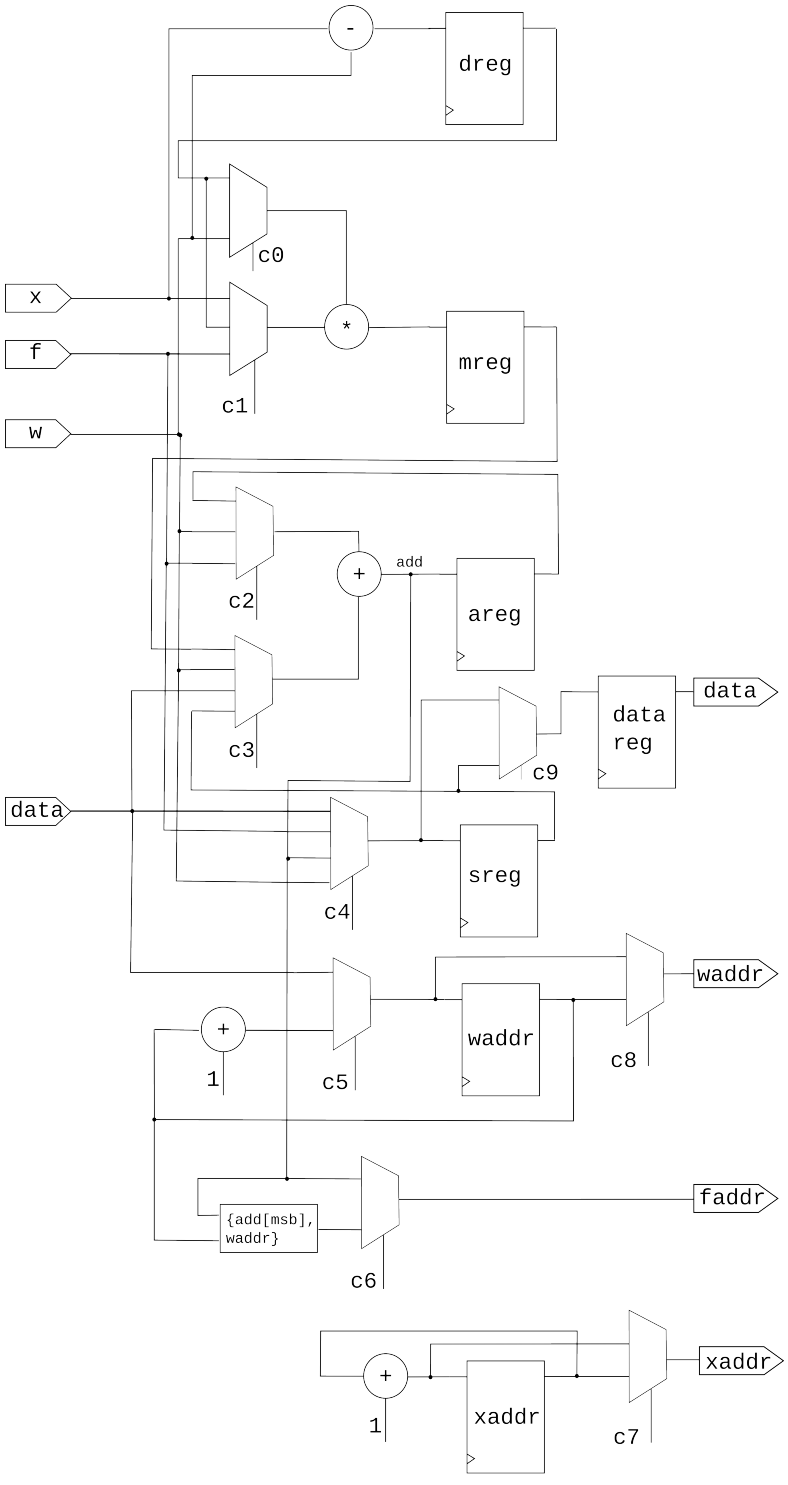


Слика 2.5: Структура CALC модула

Модул CALC (Слика 2.5) садржи неколико регистара (MREG, AREG, SREG, DREG, WADDR, XADDR, DATAREG), један множач, један сабирач, један одузимач и два инкрементера.

Регистри MREG, AREG и DREG чувају вредости из множача, сабирача и одузимача, респективно (Слика 2.6). Регистар DATAREG је излазни регистар RB модула. У зависности од конфигурације RB модула, вредност регистра RB има другачију интерпретацију. У случају када је архитектура конфигурисана да ради као DT, вредност регистра DATAREG се интерпретира као вредност класе или као базна адреса за W RAM, за наредни RB модул. Када је архитектура програмирана да ради као SVM, регистар DATAREG садржи парцијалну суму срачунату за све векторе подршке смештене у W RAM тренутног RB модула. Класификатор испрограмиран да ради као ANN у регистру DATAREG, садржи излазну вредност тренутног неурона. Регистар SREG акумулира парцијалну суму, када класификатор ради као SVM. Регистар XADDR садржи вредност улазне адресе за X RAM док комбиноване вредности регистара WADDR и XADDR представљају вредност улазне адресе за W RAM.

Машина стања модула RB контролише како подаци теку кроз комбинационе мреже и регистре. У зависности од конфигурације RB



Слика 2.6: Регистри и комбинационе мреже модула CALC

модула, машина стања мења ток података тако да се одређују потребни кораци да би се спровео прорачун за предикцију.

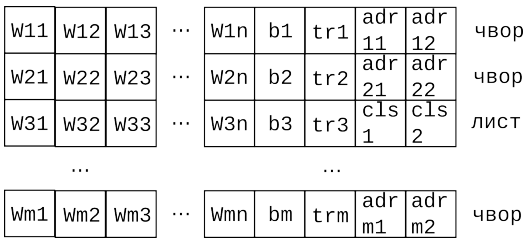
У наредним поглављима биће дат детаљнији опис рада машине стања модула RB када је архитектура RMLC конфигурисана као DT, SVM или ANN.

Глава 3

Конфигурација универзалне архитектуре као DT

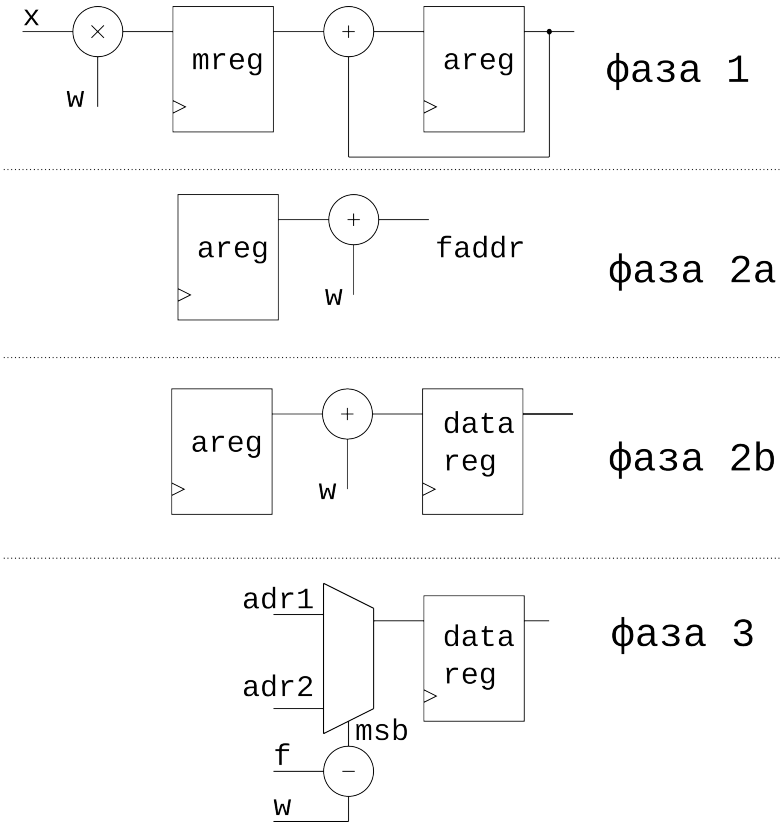
3.1 Архитектура као DT

Модул RB, када је програмиран да ради као DT, срачунава излазне вредности свих чворова и листова који су у истом нивоу структуре стабла. Меморија X RAM садржи вредности атрибута улазне инстанце, **x**. За сваки чвор тренутног слоја стабла у W RAM-у се чувају информације о тежинама **w** и додатној вредности b потребних да би се срачунала формула 1.1 (Слика 3.1). У специјалном случају када RB модул имплементира ортогоналне DT-ове све тежине за вектор **w** су једнаке нули осим једне. Уз вредности неопходне да се срачуна формула, у W RAM-у се налазе и константа прага tr као и базне адресе намењене наредном модулу, adr1 и adr2. Вредности tr1, adr1 и adr2 су намењене да се омогући гранање у стаблу. За свако од два могућа гранања у стаблу, у меморији се чувају две вредности (adr1, adr2). Коначна излазна вредност модула RB, када ради као DT, је једна од адреса, adr1 или adr2. Ова излазна вредност представља доношење одлуке у сваком чвору стабла који се посети током класификовања.



Слика 3.1: Садржај W RAM-а у случају DT-ова

Прорачун унутар модула CALC пролази кроз три фазе (Слика 3.2). У фази 1 рачуна се скаларни производ $\mathbf{w} \cdot \mathbf{x}$. У фази 2, додатна вредност b се додаје на резултат скаларног производа. Добијени резултат се прослеђује на излаз faddr који се користи као улазна адреса за меморију F RAM. У фази 3, вредност прага tr се одузима од вредности прочитане из меморије F RAM. У зависности да ли је вредност након одузимања позитивна или негативна из W RAM меморије се захвата одговарајућа адреса, adr1 или adr2. Знак вредности се одређује на основу Most Significant Bit (MSB) вредности одузимања. Вредност одговарајуће адресе се потом уписује у регистар DATAREG.



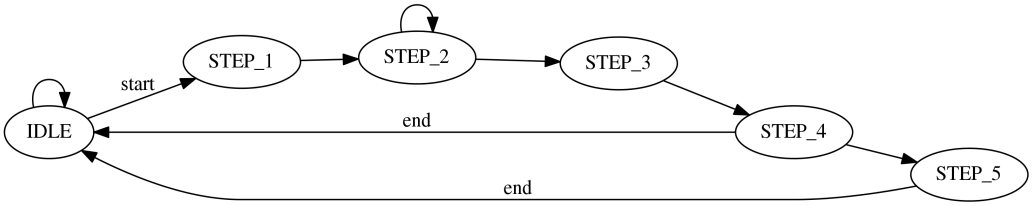
Слика 3.2: Фазе у прорачуну - DT

У фази 1, у којој се рачуна скаларни производ, постоје две могуће варијанте за имплементацију ланца за обраду. Прва варијанта, која је одабрана у RMLC архитектури, је да се ланац за обраду скаларног производа подели на два нивоа. Ово је постигнуто уметањем регистра MREG између множача и сабирача. Друга могућност је да између сабирача и множача не постоји нити један регистар што би свело ланац за обраду на само један ниво. Предност првог приступа је што

је комбинациони пут између улаза и регистра AREG скраћен па је могуће постићи већу фреквенцију рада целог система, уколико је ово и најкритичнији пут. Предност другог приступа је што има само један ниво па се цела обрада једног атрибута може урадити у само једном циклусу. Уједно други приступ би поједноставио и дизајн машине стања модула CALC. Ово је типичан случај компромиса између веће проточности система и смањења кашњења за које систем даје резултат. Пошто је цела архитектура RMLC пројектована са циљем што веће проточности, и пошто ова подела ланца на два нивоа, уноси само један циклус кашњења по инстанци, за имплементацију је одабран први приступ и поред усложњавања машине стања модула CALC.

У случају да архитектура RMLC имплементира класификационо стабло, тада прорачун на сваком нивоу стабла пролази кроз све три наведене фазе. Када се током прорачуна досегне лист стабла, тада се уместо адресе, у регистар DATAREG смешта вредност која се интерпретира као резултујућа класа. Уколико архитектура RMLC имплементира регресионо стабло и када се прорачун одвија у чвору тада се пролази кроз све три наведене фазе. Када се прорачун одвија у листу стабла тада се не пролази кроз све три фазе, већ само кроз фазу 1 и фазу 2b и вредност која се добија на излазу на крају фазе 2b се уписује као коначни резултат у регистар DATAREG.

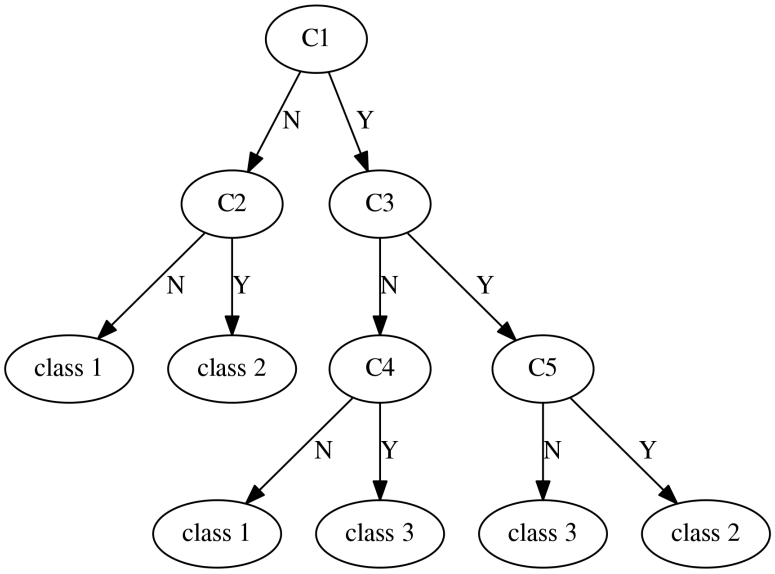
Када RB модул ради као DT класификатор, машина стања модула CALC има шест активних стања (Слика 3.3). FSM контролише процес рачунања који је описан раније. Прва фаза прорачуна одвија се кроз стања "STEP_1" и "STEP_2"; друга фаза се одвија у стању "STEP_3"; и трећа фаза се одвија у стањима "STEP_4" и "STEP_5".



Слика 3.3: Машина стања модула CALC као DT класификатора

3.2 Пример рада архитектуре као DT-а

Као пример рада архитектуре RMLC, конфигурисане да ради као DT, узећемо једно неортогонално стабло (Слика 3.4).



Слика 3.4: Стабло за илустративни пример

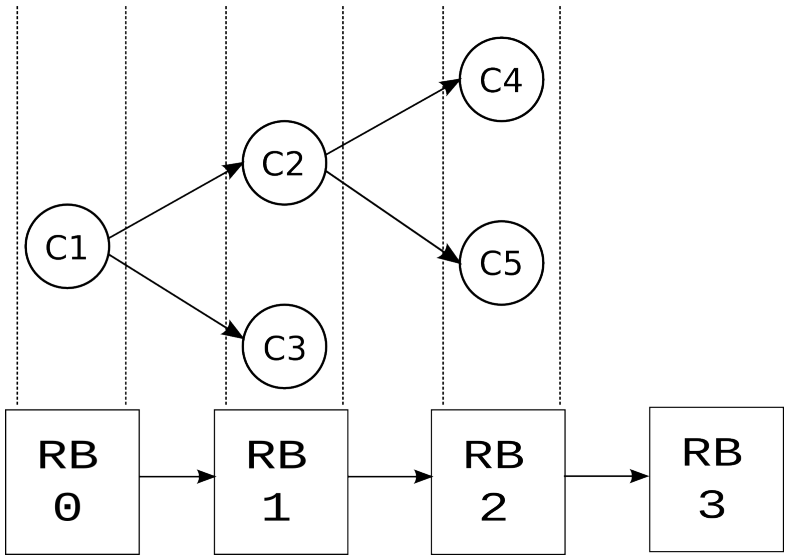
Чворови стабла су означени почетним словом “C” и бројем који их идентификује. На пример, “C1” означава први чвор, односно корен стабла. Листови стабла су обележени речима “class” након чега следи идентификациони број класе. На пример, “class 2” означава лист у коме је завршена класификација и излазна класа је 2. У сваком чвору се рачуна по један тест. Уколико је срачуната вредност за конкретну инстанцу за задату формулу истинита, тада је наредни чвор/лист онај на који се наиђе када се иде граном “Y”. У случају да је вредност неистинита, тада је наредни чвор/лист онај на који се наилази када се иде граном “N”.

За ово конкретно стабло формуле које се рачунају у чворовима су:

- C1: 0.1 a - 0.2 b + 0.3 c $>$ 0.2
- C2: -0.4 a + 0.3 b + 0.2 c $>$ -0.1
- C3: 0.2 a + 0.1 b - 0.5 c $>$ 0.3
- C4: c $>$ -0.1
- C5: b $>$ 0.2

Архитектура RMLC биће конфигурисана да има 4 RB блока. За ово конкретно стабло, то значи да ће се користити само прва 3 RB блока (Слика 3.5). Чвор стабла “C1” ће једини бити пресликан у први RB блок. Два чвора из другог нивоа стабла, “C2” и “C3”, биће пресликана

у други RB блок. У трећем нивоу стабла налазе се два чвора и два листа. Листови се не пресликавају у блокове. Они представљају резултат класификовања. То значи да ће се у трећи RB блок пресликати само чворови из трећег нивоа стабла, "C4" и "C5". У четвртом новоу стабла се налазе само листови. Они се не пресликавају у RB модуле, тако да ће четврти RB модул остати празан. Он ће само пропустити резултат класификовања претходних RB модула.

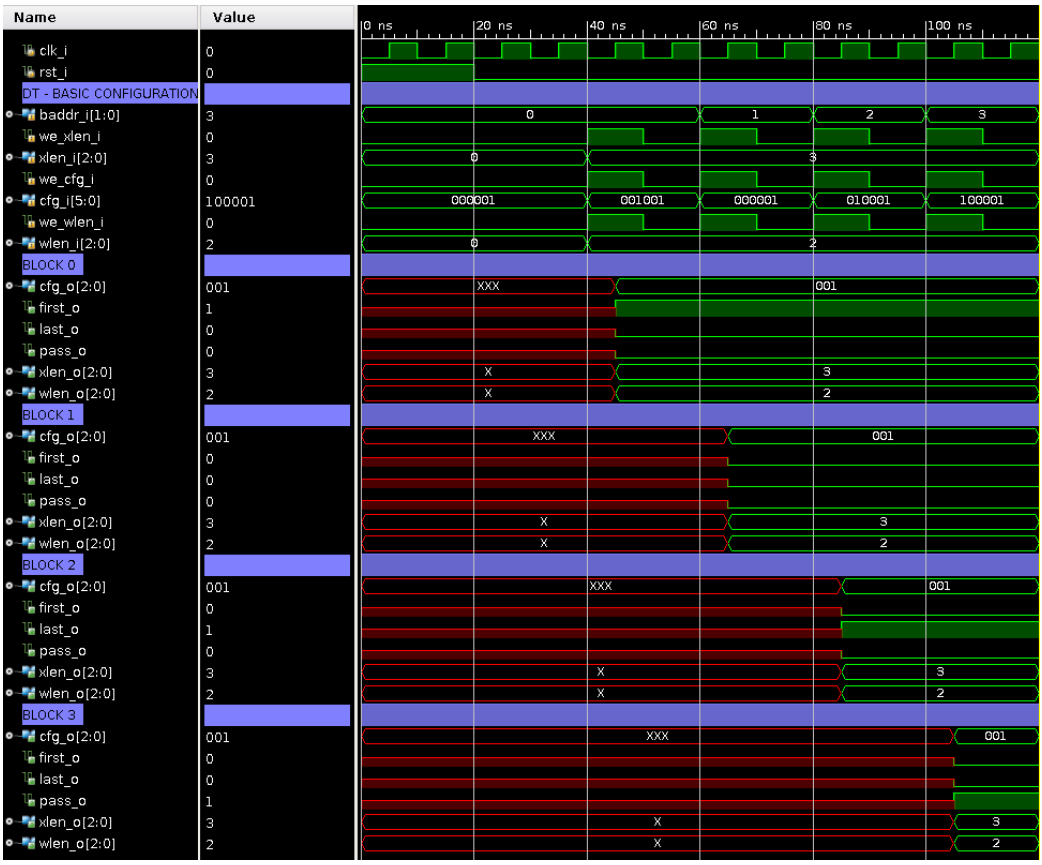


Слика 3.5: Пресликавање стабла на архитектуру

Пре него што се уради основна конфигурација свих блокова, основни регистри свих модула (cfg, xlen, wlen, first, last, pass) су неиницијализовани (Слика 3.6). Сви основни конфигурациони регистри једног блока могу да се иницијализују у једном циклусу. У једном циклусу могуће је уписивати вредности само у један од RB блокова. Вредност на улазу baddr одређује у који блок ће се уписати конфигурација. За овај пример прво је конфигурисан блок 0, потом блок 1, па блок 2 и блок 3. Овај редослед може бити произвољан.

Да би се уписала вредност у регистар xlen потребно је поставити '1' на улазну линију we_xlen. У случају када архитектура RMLC ради као DT класификатор у овај регистар се уписује број атрибута улазне инстанце па због тога на улазној линији стоји број 3.

Омогућавање уписа у регистре cfg, first, last и pass се постиже тако што се стави '1' на улаз we_cfg. На улазној линији cfg поставља се вредност за све ове регистре заједно. Три Least Significant Bit (LSB) бита се уписју у регистар cfg. Ова вредност одређује на који ће начин

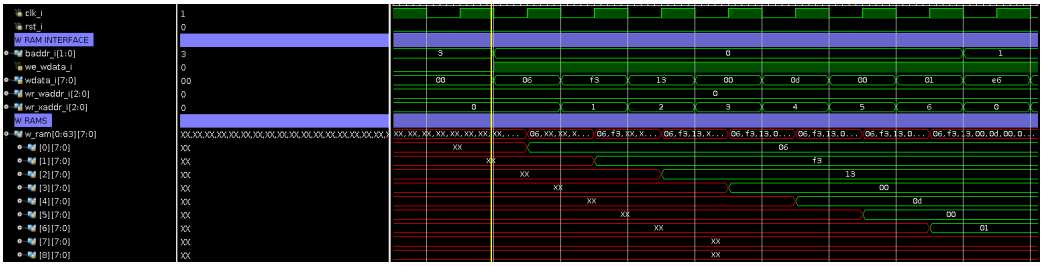


Слика 3.6: Основни конфигурациони интерфејс за DT

класификатор радити. У овом примеру то је вредност “001”, а то значи да класификатор може да ради као неортогонално стабло или као функционално стабло. Четврти бит улазне речи cfg се уписује у основни регистар first. Логично, вредност ’1’ се уписује само у RB блок на адреси 0. Пети бит улазне речи cfg се уписује у основни регистар last. Вредност ’1’ стоји само за RB модул на адреси 2, пошто је то последњи активни модул у овом примеру. Шести бит улазне речи cfg се уписује у регистар pass. Овај бит одређује да ли је RB модул конфигурисан да само пропушта вредности. Овај бит је постављен на ’1’ само за RB модул на адреси 3, пошто се у овај модул не пресликава никаква функционалност стабла.

Две улазне линије we_wlen и wlen контролишу упис у регистар wlen. Када се постави ’1’ на улазну линију we_len тада је омогућен упис у овај регистар и вредност која се уписује се налази на улазу wlen. Када класификатор RMLC ради као стабло вредност у регистру wlen није од значаја. У овом примеру произвољна вредност 2 је уписана у овај регистар.

Пре конфигурисања, меморија W RAM је неиницијализована (Слика 3.7). Меморији W RAM је неопходно више циклуса да би се иницијализовала. Додатно, у једном тренутку могуће, је радити са меморијом W RAM само једног блока RB. Блок чија се меморија конфигурише је, слично као и код основних регистара, контролисан улазним сигналом baddr.



Слика 3.7: Конфигурисање меморија W RAM једног DT класификатора

Упис у W RAM се омогућава постављањем улазног порта we_wdata на вредност ’1’. Вредност која треба да се упише у W RAM се поставља на улазни порт wdata. Локација у коју ће податак да се уписује добија се комбиновањем вредности на улазним портовима wr_waddr и wr_xaddr. Адреса локације унутар W RAM добија се рачунањем вредности:

$$address = 2^{width\_wr\_xaddr} * wr\_waddr + wr\_xaddr \tag{3.1}$$

Ширина адресне линије wr_xaddr у овом примеру је три. У првих седам локација биће уписане важеће вредности за први RB блок: вредности три атрибута, вредност прага, две адресе са који прорачун треба да започне наредни модул као и додатна вредност која се за овај тип стабла не користи. Остале локације остаће неиницијализоване. Други RB блок садржи две формуле које треба рачунати, зато ће у W RAM меморији две врсте бити заузете. Првих седам локација имаће важеће вредности. Осма локација остаће неиницијализована. Затим ће наредних седам локација опет имати вредности са значењем док ће све остале вредности остати неиницијализоване.

За стабло приказано у овом примеру, вредности које треба да се упишу у прве три локације, прве и једине врсте, првог модула RB су 0.1, -0.2 и 0.3 (Слика 3.8). За представљање реалних бројева, у овом примеру, користимо фиксну тачку са два бита у целом делу и шест бита у разломљеном делу. Негативни бројеви су представљени као комплемент двојке. Стога у прве три локације треба да се упишу битови "00000110", "11110011" и "00010011". Све вредности које се уписане у меморију W RAM су добијене одсецањем.

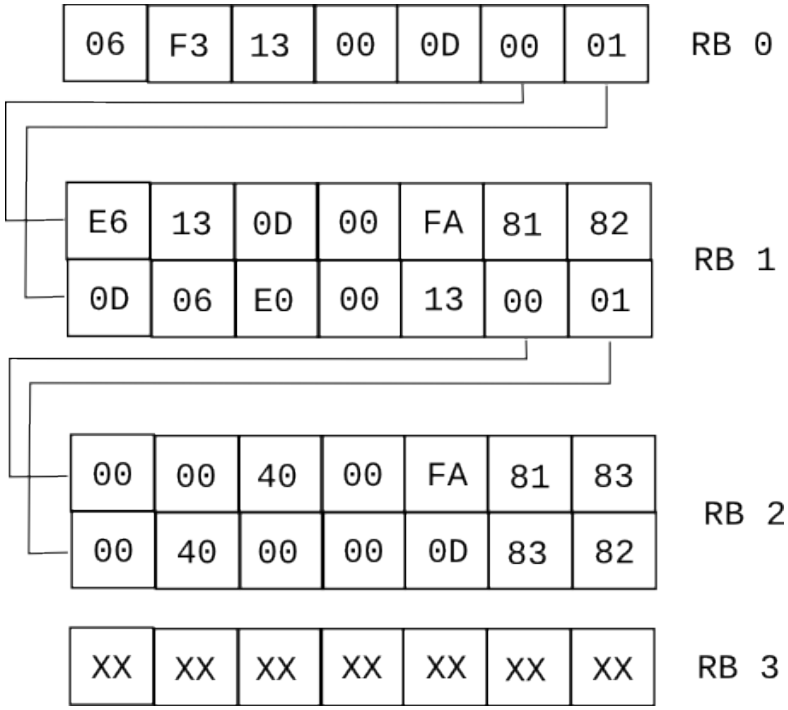
$$00000110 = 00.000110 = 2^{-4} + 2^{-5} = 0.09375 \approx 0.1$$

$$11110011 = 11.110011 = -(00.001101) = \\ -(2^{-3} + 2^{-4} + 2^{-6}) = -0.203125 \approx -0.2$$

$$00010011 = 00.010011 = 2^{-2} + 2^{-5} + 2^{-6} = 0.296875 \approx 0.3$$

Пошто се у примеру ради са неортогоналним стаблом одлуке, у ком се не додаје константа пре рачунања функције, вредност у четвртој локацији је 0. У петој локацији се налази вредност прага 0.2. На сличан начин се рачунају и вредности које су потребне за све остале модуле RB. На основу мапирања чворова по RB модулима, јасно је да ће у модулу RB 0 бити једна врста унутар меморије W RAM, у модулима RB 1 и RB 2 биће две врсте у овој меморији док ће последњи модул, RB 3, остати са потпуно неиницијализованом W RAM меморијом (Слика 3.8).

У последње две локације налазе се адресе врста са којих треба наставити прорачун у наредном модулу RB. Као што се може видети, у случају гране "N" модула RB 0, прорачун се наставља од врсте 0 у модулу RB 1. Ако се иде граном "Y" прорачун би се наставио од врсте 1 у модулу RB 1. Уколико се из чвора иде на лист, уместо адреса у



Слика 3.8: Вредности у W RAM-у за пример стабла

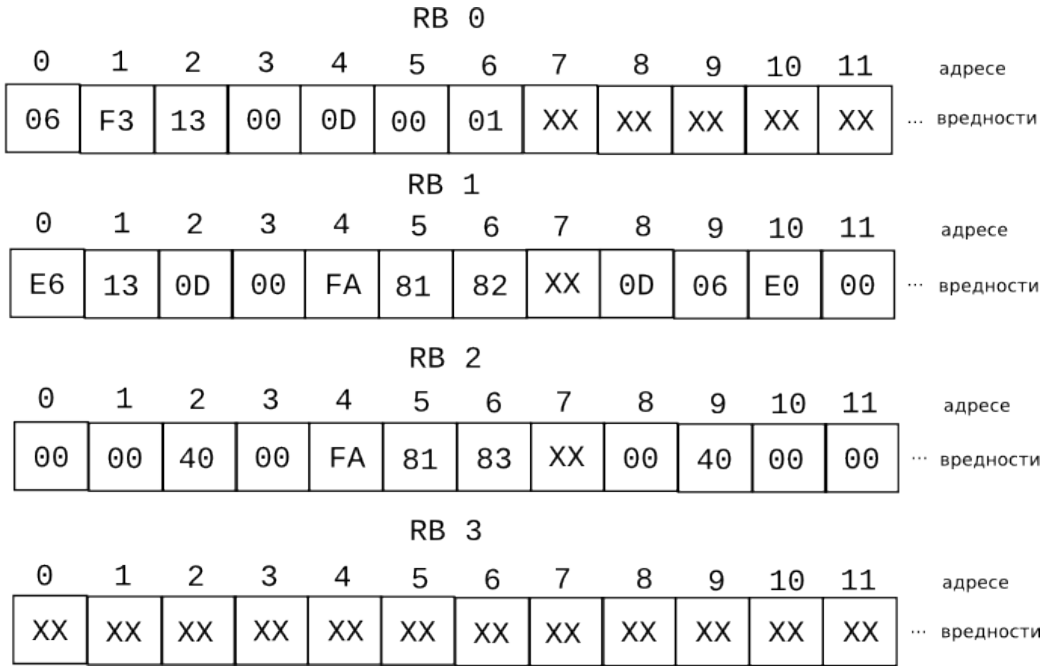
овим локацијама се налазе излазне класе. Класе и адресе се разликују по највишем биту. Уколико је највиши бит '1' тада је у питању класа, у супротом је у питању адреса. То је разлог зашто се у последње две локације свих врста, чвора "С3" налазе вредности 0x81, 0x82 или 0x83. Најзначајнији бит ових вредности означава да су у питању излазне класе док остали битови одређују која класа је у питању: 0x81 = класа 1, 0x82 = класа 2 и 0x83 = класа 3.

Модул 3 садржи два чвора која су ортогонална. У овом примеру се може видети како се једноставно имплементира ортогоналан чвор у RB модулу (Слика 3.8). Само вредност атрибута који се пореди са прагом је различита од нуле и има вредност 1.

Модул 4 је конфигурисан да само пропушта резултат класификације, зато у W RAM меморију овог модула нема потребе уписати било какве вредности.

У меморији W RAM стоје подаци који су концептуално матрице. Меморија W RAM је обична меморија са случајним приступом. Матрична структура је добијена тако што су у меморију ређане врста по врста (Слика 3.9). Улазна адреса за ову меморију добија се спајањем адресних линија wr_waddr и wr_xaddr. То значи да у меморији између

валидних података могу да се налазе неиницијализовани делови, којима архитектура никада неће приступити.



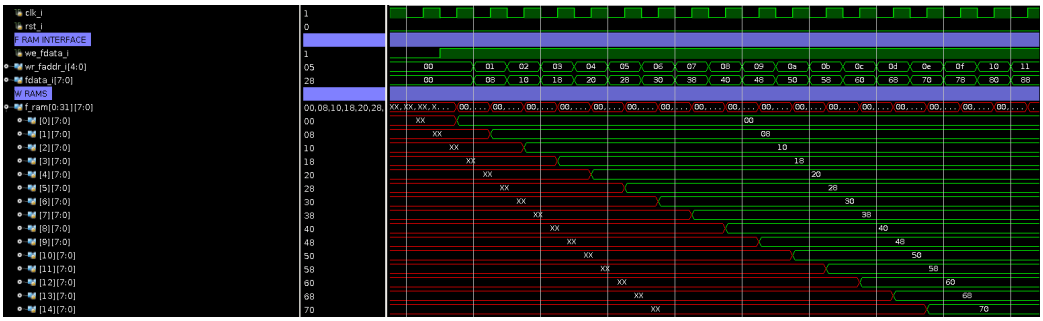
Слика 3.9: Садржај меморије W RAM-а за пример стабла

Пре конфигурисања, меморија F RAM је неиницијализована (Слика 3.10). Меморији F RAM је неопходно више циклуса да би се програмирала. Слично као са меморијом W RAM, у једном тренутку могуће је радити са меморијом F RAM само једног модула RB. Блок чија меморија се конфигурише контролисан је улазним портом baddr.

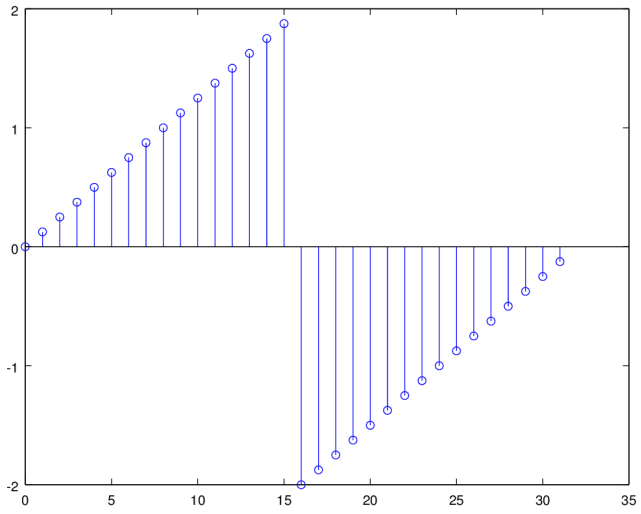
Упис у F RAM омогућава се постављањем улазног порта we_fdata на вредност '1'. Вредност која треба да се упише у F RAM поставља се на улазни порт fdata. Адреса локације у коју ће се податак уписати унутар F RAM-а, поставља се на улазни порт wr_faddr.

У овом примеру, као и у већини случајева, меморија F ADDR за све блокове у једнодимензионом низу има исти садржај. Зато што се ради са неортогоналним стаблом, садржај ове меморије чине одбирци функције идентитета. Пошто је при параметризовању архитектуре узето да је ширина порта wr_faddr 5, у овој меморији ће се налазити 32 одбирка функције.

Улаз у меморију F RAM ће бити нека међувредност током прорачуна. Та вредност ће имати више бита од 5 па ће се мање значајни бити одбацити приликом адресирања ове меморије. Приликом интепретирања



Слика 3.10: Конфигурациони интерфејс за F RAM за функцију идентитета



Слика 3.11: Одбирци функције унутар F RAM-а за стабло из примера

бита као означених бројева представљених у комплементу броја 2, ниже адресе се интепретирају као позитивне вредности а више адресе као негативне. Зато у одбирцима функције идентитета постоји нагли скок између две вредности које представљају прелаз са позитивних величина на негативне (Слика 3.11). Меморија F RAM се у архитектури користи на следећи начин:

- Одреди се улазна вредност функције чији одбирци су смештени у меморији F RAM;
- Та вредност се прошири или скрати на онолико бита колико је широка адресна магистрала F RAM меморије;
- Та вредност се пошаље као адреса у меморију F RAM;
- Наредни циклус на излазној магистрали за податке меморије F RAM налази се вредност тражене функције;

Рад архитектуре биће демонстриран на две улазне инстанце које се класификују једна за другом. Прва улазна инстанца ће бити $(1, 0, 1)$. Друга улазна инстанца ће бити $(-0.4, 0.5, 0.2)$.

Класификација за прву улазну инстанцу започиње прорачуном у чвору стабла - "C1". (Слика 3.12).

$$(0.1, -0.2, 0.3) \cdot (1, 0, 1) = 0.4 > 0.2 \implies Y \tag{3.2}$$

Након прорачуна у чвору "C1" резултат је да се иде низ грану "Y", па је наредни чвор за прорачун "C3".

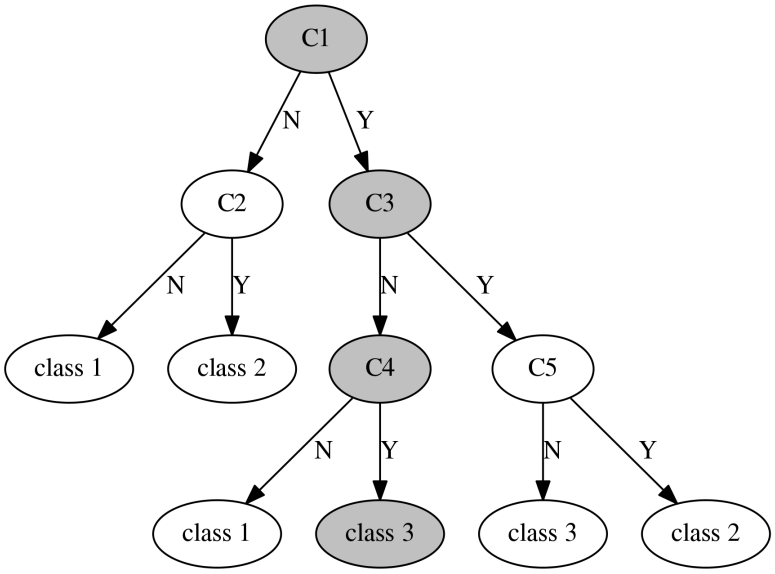
$$(0.2, 0.1, -0.5) \cdot (1, 0, 1) = -0.3 > 0.3 \implies N \tag{3.3}$$

Резултат за прорачун у чвору "C3" је да се иде низ грану "N", тако да је наредни активни чвор "C4". У том чвору се рачуна формула као за ортогонална стабла.

$$(0, 0, 1) \cdot (1, 0, 1) = 1 > -0.1 \implies Y$$

Резултат прорачуна у чвору "C4" је "Y" па је коначан резултат класификовања "класа 3".

Друга улазна инстанца биће на улазним портовима одмах након прве инстанце. То ће максимално оптеретити архитектуру. Наравно, класификација за другу улазну инстанцу започиње прорачуном у чвору стабла - "C1". (Слика 3.13).



Слика 3.12: Путања кроз стабло за прву улазну инстанцу

$$(0.1, -0.2, 0.3) \cdot (-0.4, 0.5, 0.2) = 0.12 > 0.2 \implies N$$

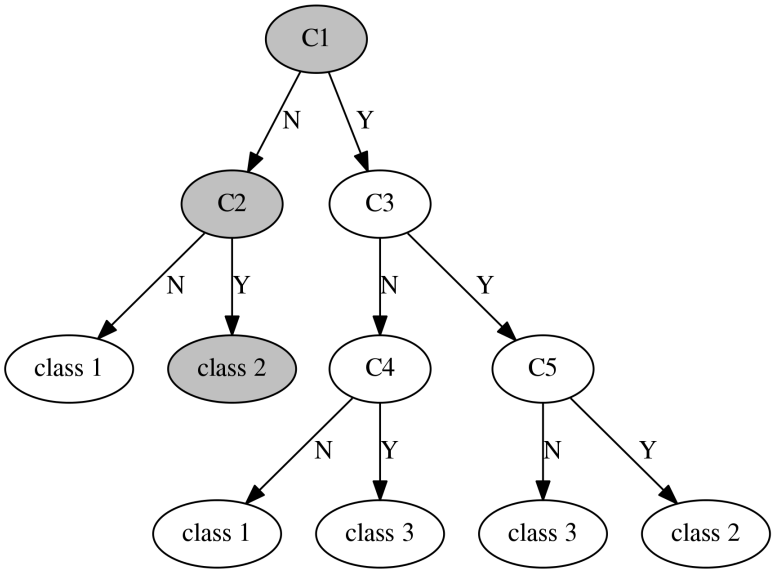
За разлику од прве инстанце, резултат у чвору "C1" је да се иде низ грану "N". На основу структуре стабла, видимо да то значи да ће за ову инстанцу ефективни прорачун бити краћи.

$$(-0.4, 0.3, 0.2) \cdot (-0.4, 0.5, 0.2) = 0.03 > -0.1 \implies Y$$

Резултат прорачуна у чвору "C2" је "Y" па је коначан резултат класификовања "класа 2".

Архитектура RMLC прихвата улазну инстанцу преко интерфејса за податке који сачињавају улазни портови data и vdata, као и излазни порт ready (Слика 3.14). Када је спољни систем спреман да проследи улазну инстанцу у архитектуру, потребно је да се улазни порт vdata постави на '1'. Архитектура ће, када је спремна за класификовање, поставити излазни порт ready на '1'. Трансфер података почеће када су оба сигнала на '1'. Када трансфер почне, не сме да стаје што значи да vdata мора остати на '1' онолико тактова колико постоји атрибута у улазној инстанци.

За овај пример, улазна инстанца има три атрибута па је, од тренутка када почне, за цео трансфер, потребно три циклуса (Слика 3.14). Може се приметити да архитектура може да прими прве две инстанце одмах једну за другом када ради као стабло. Чим први атрибут прве инстанце

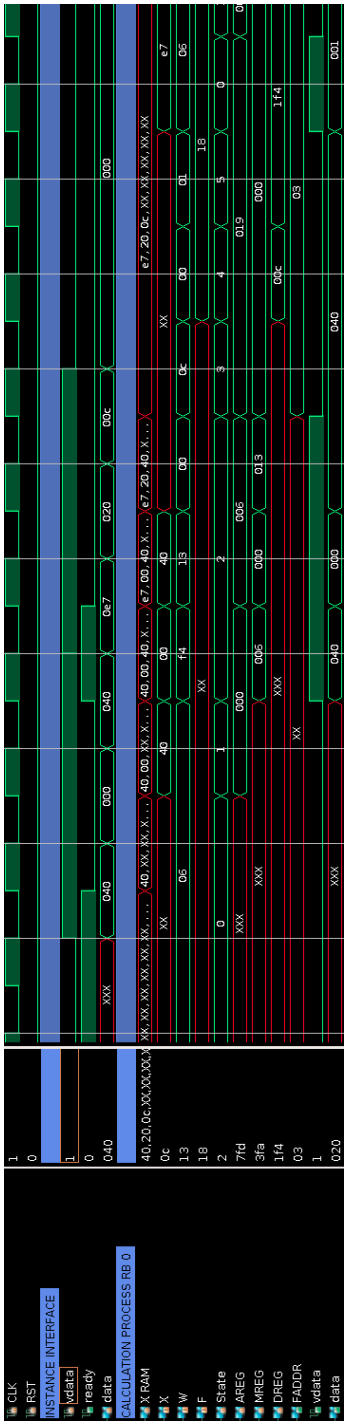


Слика 3.13: Путања кроз стабло за другу улазну инстанцу

заврши у меморији X RAM, модул CALC почиње да ради. То се на таласним облицима може видети по промени сигнала стања система "state" са вредности 0 ("IDLE") на вредност 1 ("STEP_1"). Модул CALC потом почиње да чита вредности из меморије X RAM. Чим се заврши упис атрибута прве инстанце у меморију X RAM, почиње упис атрибута друге инстанце. То је могуће зато што је модул CALC већ обрадио почетне атрибуте прве инстанце. Стања управљачког аутомата унутар CALC модула су кодована бинарно у RTL моделу.

- IDLE = 0
- STEP_1 = 1
- STEP_2 = 2
- STEP_3 = 3
- STEP_4 = 4
- STEP_5 = 5

У стању "IDLE" се након стартовања класификације унутар модула CALC, чита вредност са адресе 0 меморије X RAM и са одговарајуће адресе меморије W RAM. У случају модула RB 0 ова адреса биће исто 0, али у случају осталих модула, ова адреса не мора бити 0 већ зависи

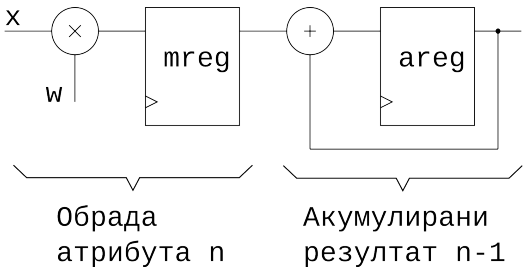


Слика 3.14: Класификовање прве инстанце унутар модула RB 0 за DT

од резултата прорачуна претходних модула. Овај случај ћемо видети за случај прве инстанце и модула RB 1. За читање из меморија је потребан један циклус, стога се модул CALC пребацује у стање “STEP_1”.

Када ради као DT класификатор, модул CALC у стању “STEP_1” има важеће вредности на улазима “X” и “W”. У овом стању модул CALC ће измножити ове вредности и омогућиће регистру MREG да прихвати резултат множења. Уз то ће прочитати вредности из меморија за наредне адресе. Вредност регистра AREG биће постављена на 0. Потом ће се модул пребацити у наредно стање “STEP_2”.

Скаларни производ се рачуна док је модул CALC у стању “STEP_2”. При првом прелазу у ово стање, вредност у регистру MREG је важећа, док је вредност регистра AREG доведена на 0. У овом стању ће модул CALC остати онолико циклуса колико атрибута има улазна инстанца. За наш пример то је три циклуса (Слика 3.14). Док је модул у овом стању активни регистри, реконфигурабилног модула CALC, су регистри MREG и AREG (Слика 3.15). У регистру AREG се акумулира резултат скаларног производа улазне инстанце и тежинског вектора. Улазна инстанца се прослеђује у модул CALC, наравно, преко улаза X, док се тежинска сума добија из меморије W RAM, преко улаза W. Циклус пре него што се аутомат стања модула CALC пребаци у наредно стање “STEP_3”, на улазу W ће се наћи додатна вредност коју треба додати на скаларни производ. Та вредност ће се сачувати у регистру MREG. Уједно док је у стању “STEP_2” модул RB прослеђује атрибуте инстанце која се класификује на излазни порт data и поставља излазни порт vdata на ’1’.



Слика 3.15: Ланац за рачунање у модулу CALC у стању “STEP_2” када класификатор ради као DT.

Када је аутомат у стању “STEP_3”, тада се на вредност скаларног производа додаје додатна вредност сачувана у регистру MREG. Добијена вредност се шаље као улазна адреса меморије F RAM. У наредном циклусу ће се на улазу F, стога, појавити вредност функције. У овом кораку се уради и додавање вредности као и рачунање функције. Док

је машина стања модула CALC у стању “STEP_3” улаз W садржи вредност прага. Ова вредност ће се сачувати у регистру DREG и биће искоришћена у наредном стању “STEP_4”. Аутомат се у овом стању задржава један циклус (Слика 3.14). У овом стању у регистру AREG се налази и срачуната вредност скаларног производа. За прву инстанцу вредност која се налази у регистру AREG је 0x19, што одговара реалном броју 0.390625. То је приближно вредност 0.4 која треба да се добије за овај скаларни производ (Формула 3.2). Додатна вредност у овом примеру је 0 тако да ће вредност која ће бити послата као адреса за меморију F RAM бити 0x19, заокружена на 5 значајних цифара.

Аутомат у стању “STEP_4” упоређује вредност функције и прага и на основу тога доноси одлуку која је врста активна за наредни модул RB или даје резултат класификације. Вредност која је прочитана из F RAM меморије је 0x18 (Слика 3.14). Пошто је као адреса послата вредност 0x19, а у F RAM-у се налази функција идентитета, произилази да се један бит прецизности изгубио. Ово се десило стога што се у F RAM меморији налази 32 одбирка функције јер се ова меморија адресира са адресама ширине 5 бита. Пре адресирања улазна вредност је заокружена на 5 значајних цифара па се добија:

$$\texttt{0x19} = 00011001 = 00.011001 \implies$$
$$addr = 00.011 \implies value = 00.011000 = \texttt{0x18}$$

На улазу W се у стању “STEP_4” налази вредност која ће се проследити даље уколико се прати грана “N”. Уколико би се након одузимања вредности функције од прага добила позитивна вредност, наредно стање аутомата би било стање “IDLE” и вредност са улаза “W” би се проследила на излаз. У овом примеру вредност која се добија је негативна па аутомат прелази у стање “STEP_5” уједно читајући наредну вредност из меморије W RAM.

Када је машина стања модула CALC у стању “STEP_5” тада се вредност на улазу W прослеђује на излаз. Ово је резултат прорачуна модула RB. Наредно стање је стање “IDLE”. Прелаз аутомата из стања “STEP_5” у стање “IDLE” еквивалентан је праћењу гране “Y” у стаблу. За пример прве инстанце коначан резултат прорачуна у чвору “C1” је 0x01 (Слика 3.14). То значи да ће наредни модул у низу RB 1 почети свој прорачун за прву инстанцу од врсте 0x01 у својој меморији W RAM. Модул CALC подиже излазни порт vdata на ’1’ чиме означава да је излазна вредност важећа, а резултат прорачуна, 0x01, ставља на излазни порт data. На таласним облицима (Слика 3.14) се види и како

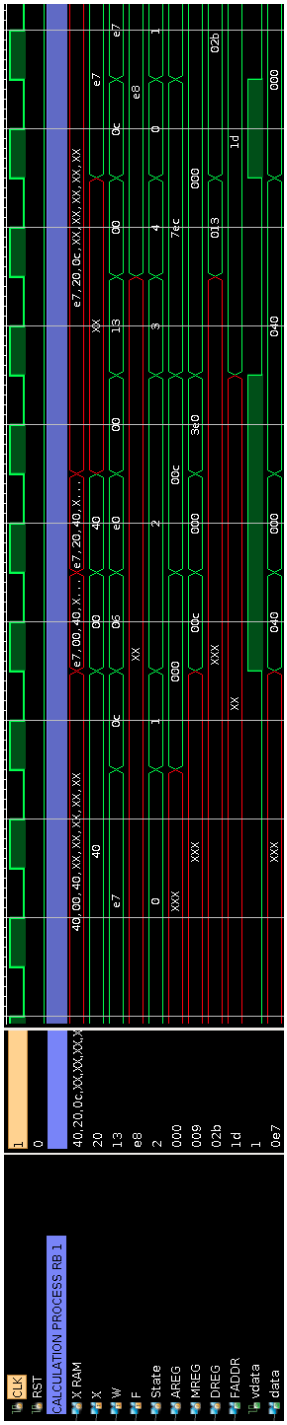
вредности атрибута улазне инстанце одлазе кроз излазне портове до наредног модула RB 1. Након атрибута инстанце на излаз је прослеђен и резултат прорачуна.

У модулу RB 1 прорачун се започиње када кроз улазне портове за податке, vdata и data, стигну атрибути улазне инстанце као и резултат прорачуна претходног модула, RB 1. Модул X на основу конфигурационог бита first има информацију када да започне прорачун у модулу CALC. У случају да је овај бит '1', као у случају модула RB 0, прорачун се започиње након примљеног последњег атрибута. У случају када је овај бит '0' прорачун се започиње тек након примљеног резултата претходног модула RB. Излазни порт ready нема функцију, осим ако је конфигурациони бит first '1'. То значи да ready синхронише трансфер података само између спољних система и архитектуре RMLC, док нема никакав утицај унутар архитектуре. Унутар архитектуре проток информација је такав да размена података на обе стране није неопходна.

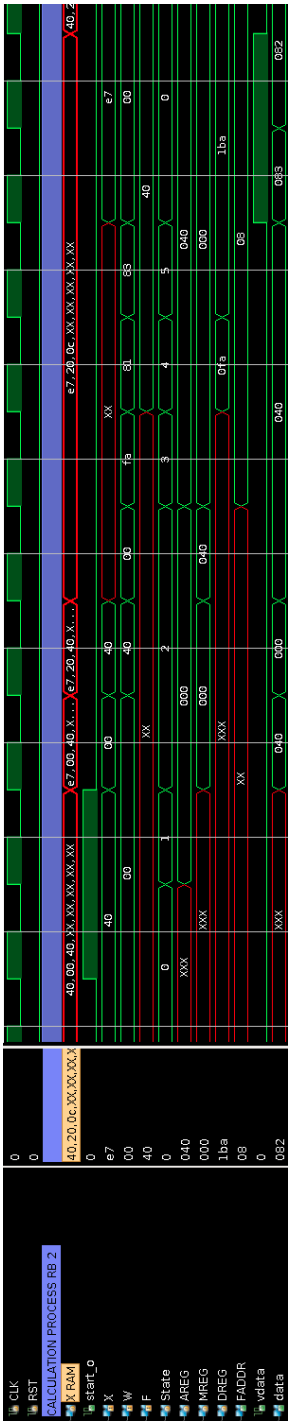
Након стартовања прорачуна унутар модула RB1, рачунање се одвија на већ описани начин (Слика 3.16). Пошто је резултат прорачуна претходног модула 0x01, модул RB 1 ће тежине из меморије W RAM узимати из врсте 1. За овај модул у врсти 0 се налазе тежине за чвор "C2", а у врсти 1 се налазе тежине за чвор "C3". Пошто се читају тежине из врсте 1, то значи да ће модул RB 1 радити рачун за чвор "C3" што и треба да буде, пошто се у току прорачуна за прву инстанцу на другом слоју стабла наилази на чвор "C3".

Након рачунања скаларног производа у овом модулу резултат треба да буде -0.3 (Формула 3.3). У стању "STEP_3" аутомата модула CALC, срачуната вредност је 0x7EC, што одговара реланом броју -0.265625. (Слика 3.16) У овом случају разлика између добијене вредности и тачне вредности (-0.3) је већа него у првом случају али је и даље довољно добра за тачну класификацију. У модулу RB 1 за прву инстанцу се прати грана један тако да машина стања модула CALC неће отићи у стање "STEP_5". Другим речима, прорачун у овом модулу ће трајати за један циклус краће него у модулу RB 0. Резултат прорачуна је вредност 0x0 која представља адресу од које ће модул RB 2 започети свој рачун. Ово се интерпретира као праћење гране "N" у стаблу прорачуна и прелаз са чвора "C3" на чвор "C4" (Слика 3.12).

Модул RB 2 ће радити прорачун за чвор "C4". Почетна адреса која је примљена од претходног модула је 0x0 што значи да у меморији W RAM треба читати вредности из прве врсте а ту се налазе тежине за чвор "C4" (Слика 3.17) . Овај модул ради прорачун као код ортогоналних стабала. Резултат скаларног производа који се добија је 0x40 што



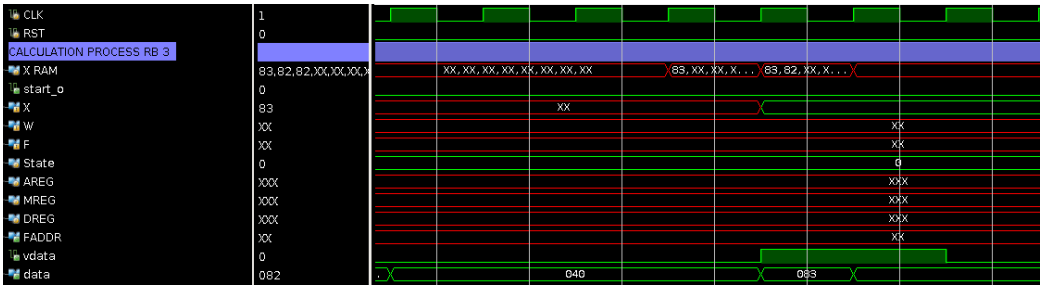
Слика 3.16: Прорачун за прву инстанцу унутар модула RB 1



Слика 3.17: Прорачун за прву инстанцу унутар модула RB 2

представља реалну вредност 1.0 (Формула 3.2) и одговара атрибуту “c” улазне инстанце. Овај модул уједно и завршава класификацију. Као резултат на излазни порт шаље се 0x83. Вредност ’1’ на биту 7 резултата означава да је класификација завршена и да се вредност резултата налази у преосталим битима на нижим позицијама. У овом случају, улазна инстанца је класификована у класу 3, што је и тачан резултат (Слика 3.12).

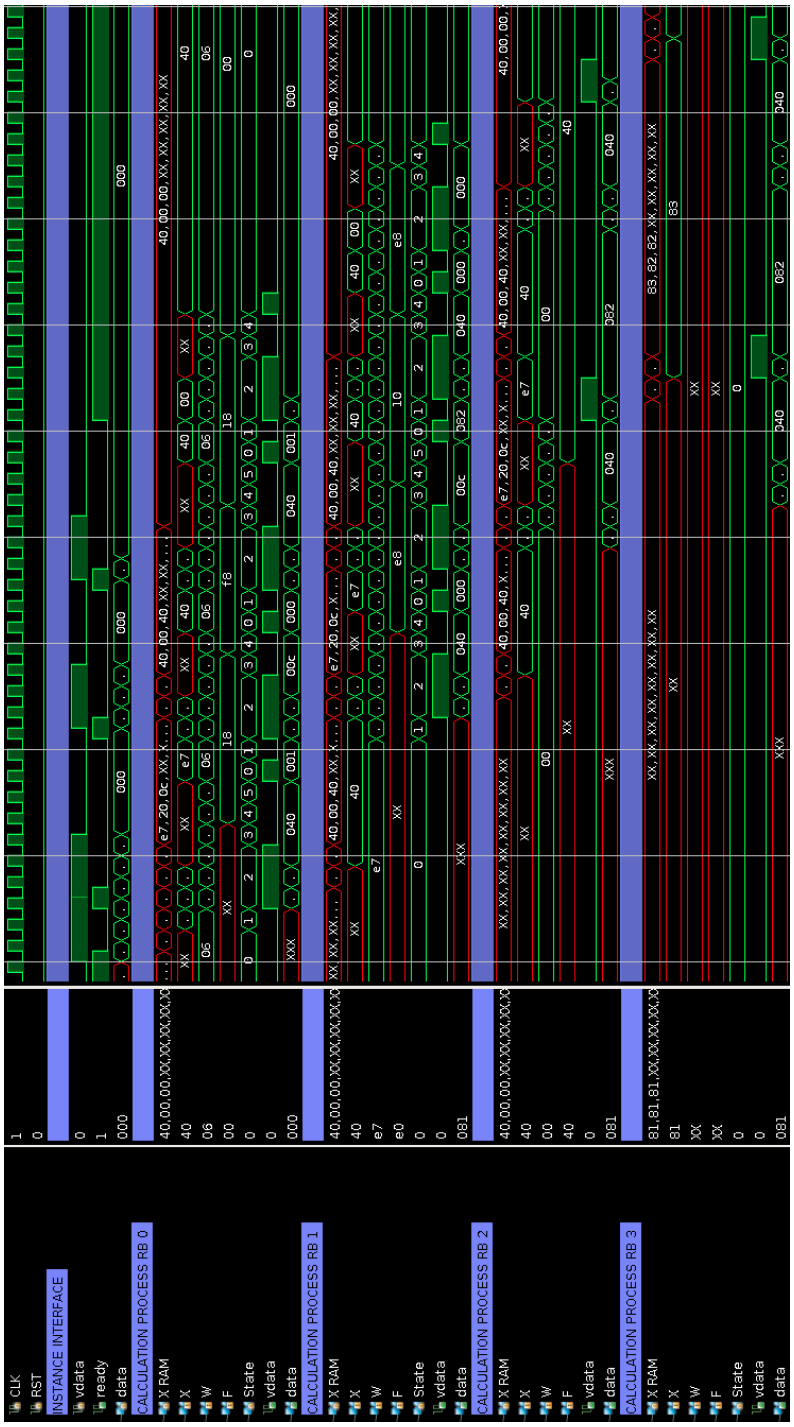
На таласним облицима се може приметити да је, циклус након завршетка класификовања прве инстанце, стигао и резултат класификовања друге инстанце $(-0.4, 0.5, 0.2)$. Пошто је прорачун за другу инстанцу краћи, модул RB 2 неће имати никакав прорачун да ради у случају инстанце 2 па ће само проследити резултат на излазне портове. Као што се може видети, примљени резултат је 0x82. То значи да је другој улазној инстанци додељена класа 2, што је и у овом случају тачан резултат.



Слика 3.18: Прорачун за прву инстанцу унутар модула RB 3

Модул RB 4 је конфигурисан тако да само пропушта добијене резултате зато што је архитектура параметризована са 4 модула RB, док стабло које имплементира има само 3 слоја. Машина стања модула CALC, у овом случају, стално ће бити у стању “IDLE” и пропуштаће резултат са улазних портова за податке ка изланим портовима.

Организација архитектуре је таква да је оптимизована проточност. Захваљујући томе се, и поред извесног кашњења за добијање првог резултата, добија велико убрзање када улазне инстанце иду једна за другом. Уколико се архитектури проследи довољан број улазних инстанци једна за другом, сви RB модули који учествују у прорачуну могу бити активни истовремено (Слика 3.19). Када архитектура ради као DT класификатор, времена класификовања инстанце су непредвидива и зависе од саме инстанце. Ово се може уочити на таласним облицима излазних портова последњег модула у низу. Када



Слика 3.19: Поглед на прорачун свих модула на 4 улазне инстанце DT класификатора

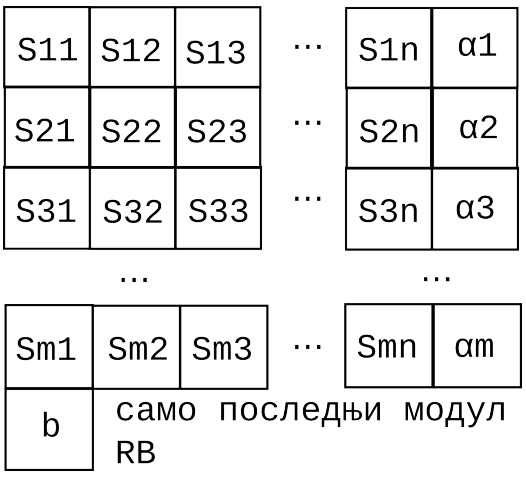
архитектура ради као DT класификатор мора да има бар онолико RB модула колика је дубина стабла. Додавање додатних RB модула у архитектуру неће довести до већег убрзања. Видећемо да за неке класификаторе ово није случај.

Глава 4

Конфигурација универзалне архитектуре као SVM

4.1 Архитектура као SVM

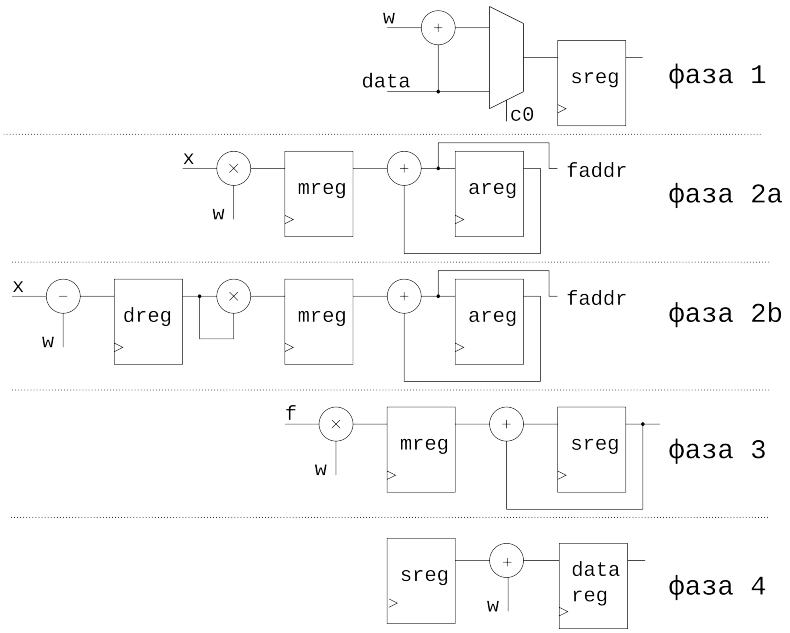
Када RMLC архитектура ради у SVM моду, RB модул срачунава део суме из формуле 1.2 и прослеђује је до наредног RB модула у једнодимензионом низу. Последњи RB модул у низу срачунава коначну суму и одређује коначни резултат класификације. Меморија X RAM садржи вредности атрибута улазне инстанце, **x**. F RAM меморија садржи одбирке нелинеарне кернел функције k у формули 1.2. Сваки W RAM садржи вредности компоненти вектора подршке **s** и Лагранжових множитеља, α. W RAM меморија последњег RB модула у последњој колони садржи додату вредност b (Слика 4.1).



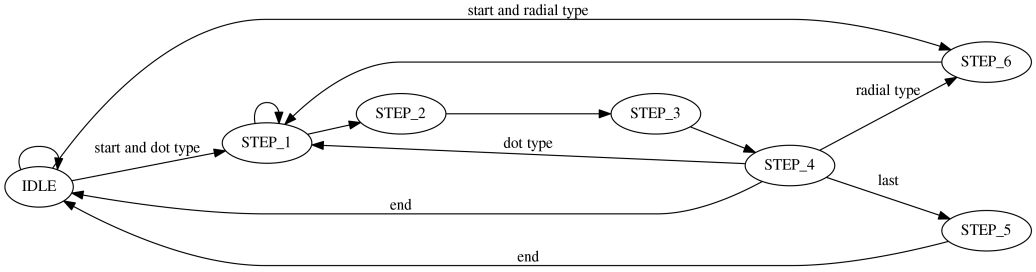
Слика 4.1: Садржај W RAM-а у случају SVM-ова

У случају када модул CALC ради прорачун за SVM класификатор, прорачун пролази кроз 4 фазе. У првој фази регистар SREG се иницијализује. Регистар SREG у првом модулу се иницијализује на нулу. У последњем модулу, регистар SREG се иницијализује сумом излазне вредности претходног модула RB и додатне вредности b из W RAM-а. Сви остали регисти SREG, који припадају модулима RB између првог и последњег, се иницијализују излазним вредностима претходних RB модула. У фази 2, кернел функција се срачунава. У зависности од кернел функције, постоје две могуће варијанте фазе 2. (2a и 2b на слици 4.2). У случају да SVM ради са полиномијалним језгром или са језгром које садржи скаларни производ, само множач и сабирач су неопходни за релизацију (2a на слици 4.2). Када SVM ради са радијалним језгром, сва три елемента за рачунање су неопходна: сабирач, одузимач и множач (2a на слици 4.2). У оба случаја срачуната вредност се прослеђује на излаз faddr. Излаз faddr је адресни улаз за меморију F RAM. Садржај адресиране меморијске локације F RAM-а представља срачунату вредност одговарајућег језгра. У трећој фази, добијена вредност се множи са α и резултат се додаје на садржај регистра SREG. Фазе 2 и 3 се понављају за све векторе подршке смештене у меморију W RAM модула RB. Уколико модул RB није последњи у низу, када се заврши прорачун за последњи вектор подршке, вредност из регистра SREG се прослеђује до регистра DATAREG и прорачун се завршава. Ако је модул последњи у низу, тада се прелази на фазу 4 у којој се додаје додатна вредност b на резултат сумирања. Та вредност се прослеђује на регистар DATAREG. Излаз data је повезан на регистар DATAREG, и тај излаз је везан на улаз data наредног RB модула у једнодимензионом низу.

Када RB модул ради као SVM класификатор, машина стања модула CALC има шест или седам активних стања (Слика 4.3). FSM контролише процес рачунања који је већ описан. Прва фаза прорачуна се завршава у стању "IDLE". У случају када FSM ради са језгрима која садрже скаларни производ, фаза 2 се односи на проласке FSM-а кроз стања "STEP_1", "STEP_2" и "STEP_3". Када машина стања ради са радијалним језгрима фаза два се одвија у проласцима кроз стања "STEP_6", "STEP_1", "STEP_2" и "STEP_3". Фаза 3 се одвија у стању "STEP_4". За последњи модул у низу ради се и фаза 4 у стању "STEP_5".



Слика 4.2: Фазе у прорачуну - SVM



Слика 4.3: Машина стања модула CALC као SVM класификатора

	a	b	c	α
s1	0.5	0.25	0.5	0.25
s2	-0.25	-0.4	-0.3	-0.3
s3	1.0	-0.5	-0.2	-0.5
s4	1.0	0.5	0.2	0.5
s5	-1.0	0	0	-0.6
s6	0	1.0	0	-0.5
s7	-0.5	-0.5	-1	0.4

Табела 4.1: Класификатор SVM за илустративни пример

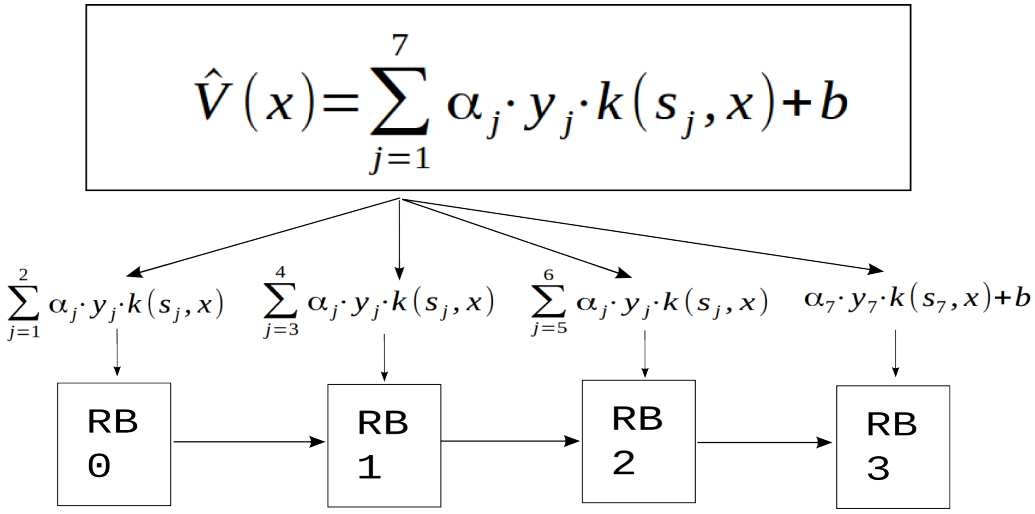
4.2 Пример рада архитектуре као SVM-а

За илустрацију рада архитектуре као SVM класификатора узећемо полиномијални SVM класификатор. Класификатор ће садржати седам вектора подршке (Табела 4.1). У табели, симболом α је обележена вредност Лагранжовог множитеља помноженог са +/-1, у зависности од тога којој класи припада одговарајући вектор подршке. То значи да ће множитељи моћи да имају и негативне вредности. SVM предиктивни модели улазну инстанцу класификују у две излазне класе у зависности од коначно срачунате вредност. Они су бинарни класификатори. За овај пример ћемо узети да за случај када улазна инстанца резултује у коначној суми која је позитивна, тада она припада класи 1 у супротном инстанца припада класи 2.

Вектори подршке се налазе у врстама табеле и означени су словима "SV" уз број који их јединствено идентификује. Атрибути вектора се налазе у колонама a, b и c. Лагранжов множитељ сваког вектора је дат у колони α. Додатна вредност за овај класификатор је -0.2.

Архитектура RMLC биће конфигурисана да има 4 RB блока. У случају када архитектура ради као SVM сви блокови у већини случајева могу бити искоришћени за прорачун. Потребно је само распоредити равномерно све векторе у W RAM меморије. Једини случај када неки блокови не могу бити употребљени и треба да се конфигуришу да пропуштају резултат је када је број вектора подршке мањи од броја RB модула. За овај пример у прва три модула биће распоређена по два вектора подршке док ће последњи модул садржати само један вектор. Додатна вредност биће садржана у последњем RB модулу.

Основни конфигурациони регистри (cfg, xlen, wlen, first, last, pass) су неиницијализовани након ресетовања архитектуре (Слика 4.5). Сви основни конфигурациони регистри за један блок су иницијализовани у једном циклусу. Вредност у регистру baddr одређује који се блок



Слика 4.4: Пресликавање SVM класификатора на архитектуру

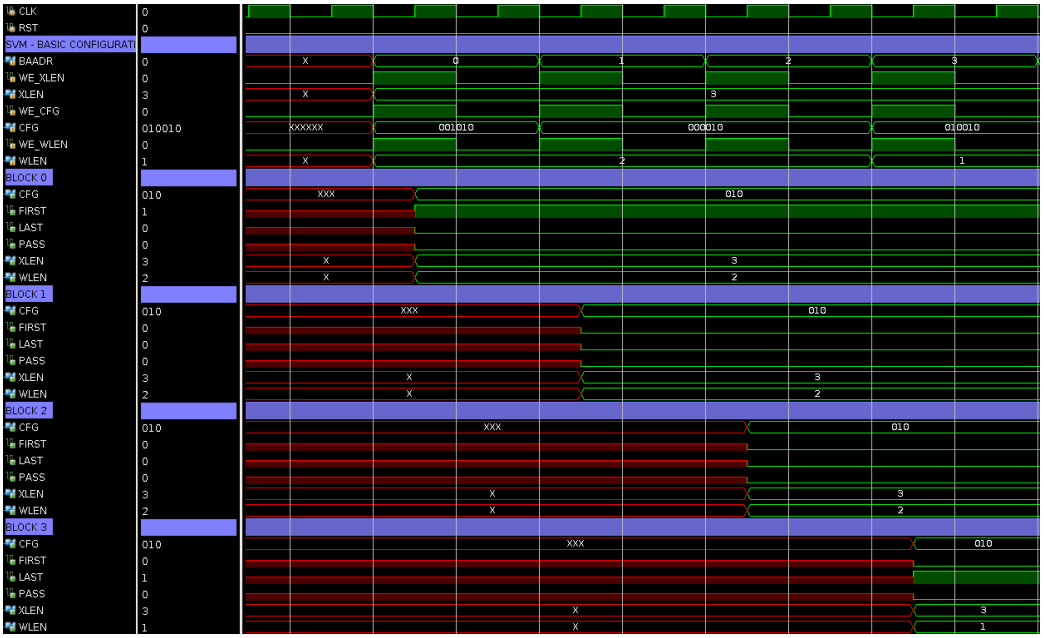
конфигурише, тако да су, слично као у примеру са стаблима, блокови иницијализовани у растућем редоследу.

Регистар xlen када архитектура ради као SVM, означава колико атрибута има улазна инстанца. Овај регистар се, и у овом случају, иницијализује на вредност 3.

Регистар cfg унутар модула CONFIG је постављен на вредност “010” што значи да ће модул радити као SVM класификатор. Само модул RB 0 има вредност бита first постављену на ’1’. То значи да је овај модул први у једнодимензионалном низу и да је једино његов излаз ready од значења. Модул RB 3 једини има регистар last постављен на ’1’. То значи да је овај модул последњи у низу, тај модул садржи додатну вредност и завршава класификовање. Нити један од модула нема постављен регистар pass на један, што значи да ће сви модули бити активни у прорачунавању.

Пошто класификатор ради као SVM садржај регистра wlen је од значаја. Вредност унутар овог регистра означава колико је вектора смештено у меморију W RAM модула CONFIG. Може се уочити да сви модули, осим последњег, имају подешену вредност 2 у овом регистру (Слика 4.5). За све модуле, осим последњег то значи да имају два вектора унутар меморије. За последњи модул, у овом регистру је смештена вредност 1. Ово значи да је у меморију смештен један вектор подршке. Пошто је модул последњи и конфигурисан је као SVM у њему је смештена и додатна вредност.

Као што је већ напоменуто, меморија W RAM је неиницијализована (Слика 4.6) и неопходно јој је више циклуса да би се иницијализовала.

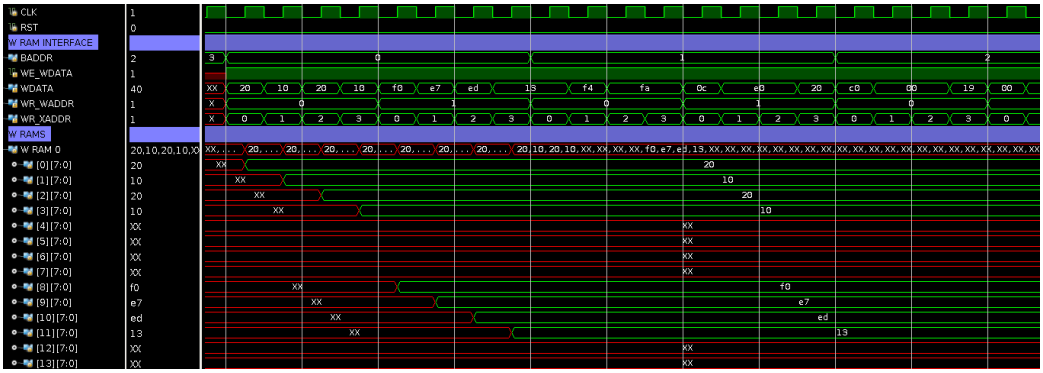


Слика 4.5: Основни конфигурациони интерфејс за SVM

У једном тренутку могуће је радити са меморијом W RAM само једног блока RB а блок чија меморија се конфигурише је контролисан улазним портом baddr.

Улазни портови који се користе за уписивање садржаја у меморију W RAM су we_data, wr_waddr и wr_xaddr. Као што је већ описано, када је порт we_data постављен на вредност '1', омогућен је упис у меморију. Ефективна адреса, у коју се уписује податак, добија се помоћу формуле 3.1.

Када се ради са SVM класификаторима, за прорачун су неопходни вектори подршке као и Лагранжови множитељи. Као што је већ речено за представљање бројева се користи репрезентација са фиксном тачком са два бита у целом делу и шест бита у разломљеном док се за негативне бројеве користи комплемент броја два. Први вектор подршке (Табела 4.1) је (0.5, 0.25, 0.5), а његов множитељ је 0.25. У меморију ће прво бити уписане вредности вектора 0x40, 0x20 и 0x40, а потом и вредност за множитељ 0x20. Прва врста заузима укупно првих осам локација, пошто је адресна магистрала wr_xaddr широка три бита. Из тог разлога наредне четири локације остају неиницијализоване. Потом следе вредности за други вектор подршке (-0.25, -0.4, -0.3) као и његов множитељ 0.3. Те вредности су: 0xf0, 0xe7 и 0xed. На крају се налази и



Слика 4.6: Конфигурисање меморија W RAM једног SVM класификатора

вредност за множитель 0x13. Све вредности које се уписане у меморију W RAM су добијене одсецањем.

$$\texttt{0x20} = 00.100000 = 2^{-1} = 0.5$$

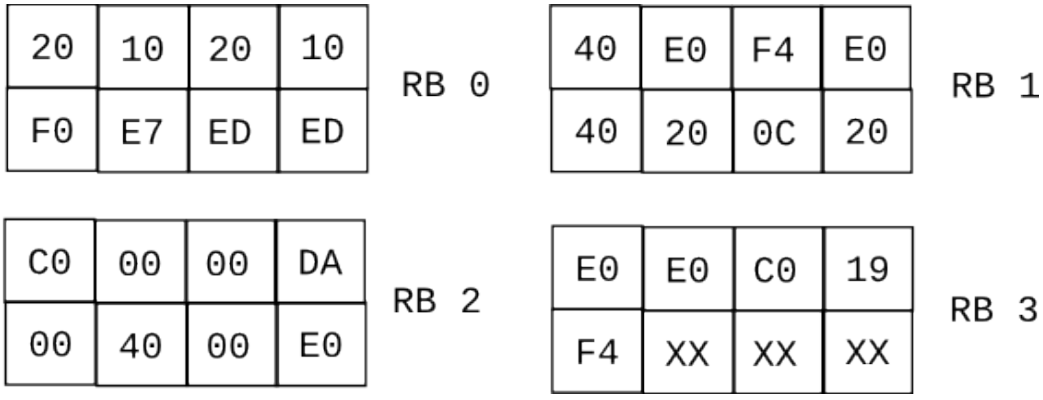
$$\texttt{0x10} = 00.010000 = 2^{-2} = 0.25$$

$$\texttt{0xF0} = -(compl2(11.110000)) = -(00.010000) = -2^{-2} = -0.25$$

$$\texttt{0xE7} = -(compl2(11.100111)) = -(00.011001) = \\ -(2^{-2} + 2^{-3} + 2^{-6}) = -0.390625 \approx -0.4$$

$$\texttt{0x20} = 00.010000 = 2^{-2} = 0.25$$

Слично као у модулу RB 0, у модулима RB 1 и RB 2 су смештени вектори подршке заједно са њиховим Лагранжовим множитељима (Слика 4.7). У сва три модула су смештена по два вектора подршке. Једини модул који нема тачно овакву структуру је последњи модул у низу. То је у овом случају RB 3. У њему се уз векторе подршке у једној



Слика 4.7: Вредности у W RAM-у за пример SVM-а

врсти налази и само додатна вредност. За овај пример у овом модулу се налази један вектор подршке са додатном вредношћу.

Приликом конфигурисања система, да би архитектура имала највећу брзину обраде података, векторе подршке је потребно распоредити што равномерније по модулима RB. Број вектора подршке, распоређених по модулима, треба да иде у опадајућем редоследу. У овом примеру све то је испоштовано па ће архитектура имати максималну брзину обраде улазних инстанци.

Матрична структура унутар меморије W RAM добијена је спајањем адресних линија wr_waddr и wr_xaddr. У меморију су ређане врста по врста (Слика 4.8). Као што је већ напоменуто, ово значи да ће се у меморији између важећих података налазити и локације које нису иницијализоване. Локације које нису иницијализоване имају произвољну вредност означену са XX. То су уједно и делови у меморији којима архитектура неће никада приступити.

Иницијализација меморије F RAM траје онолико тактова колико одбирака функције је узето за прорачун (Слика 4.9). То значи да је у овом случају потребно 32 циклуса за инцијализацију меморије F RAM. Наравно, за одређивање модула, чија меморија F RAM се програмира, користи се улазни порт baddr.

И у случају SVM класификатора, остали сигнали за програмирање меморије F RAM су we_fdata, fdata и wr_faddr. Улазни сигнал we_fdata омогућује упис податка који се налази на порту fdata на адресну локацију која је одређена вредношћу на улазу wr_faddr.

Меморија F ADDR има исти садржај за све модуле RB у низу када класификатор ради као SVM. За овај класификатор је одабрана

RB 0

0	1	2	3	4	5	6	7	8	9	10	11	
20	10	20	10	XX	XX	XX	XX	F0	E7	ED	ED	адресе / вредности

RB 1

0	1	2	3	4	5	6	7	8	9	10	11	
40	E0	F4	E0	XX	XX	XX	XX	40	20	0C	20	адресе / вредности

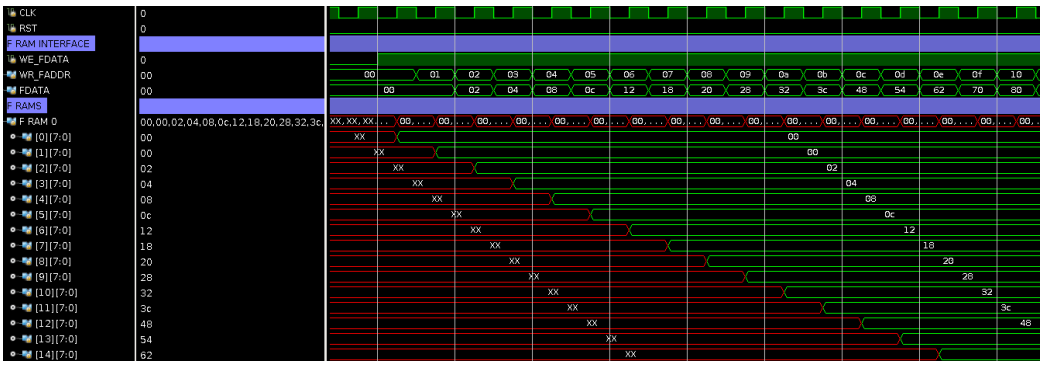
RB 2

0	1	2	3	4	5	6	7	8	9	10	11	
C0	00	00	DA	XX	XX	XX	XX	00	40	00	E0	адресе / вредности

RB 3

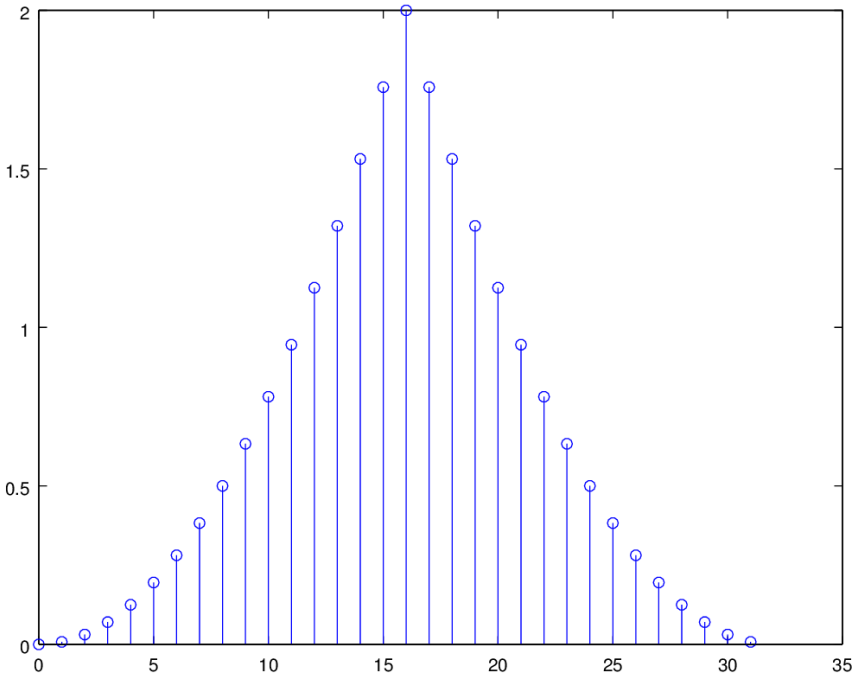
0	1	2	3	4	5	6	7	8	9	10	11	
E0	E0	C0	19	XX	XX	XX	XX	F4	XX	XX	XX	адресе / вредности

Слика 4.8: Садржај меморије W RAM-а за пример SVM-а



Слика 4.9: Конфигурациони интерфејс за F RAM за квадратну функцију

полиномијална функција $1/2x^2$ и пошто је ширина адресног порта параметризована бројем 5, ова функција ће бити узоркована у 32 тачке.



Слика 4.10: Одбирци функције унутар F RAM-а за SVM класификатор из примера

Адресни улаз у меморију F RAM ће бити вредност скаларног производа улазне инстанце и вектора подршке. Та вредност ће у овом случају имати више битова од 5, колико је потребно за адресирање па ће се вишак бита одбацити. Ова вредност је означен број представљен као комплемент броја 2, па као што је већ описано код примера за DT, ниже адресе се интерпретирају као позитивни бројеви а више као негативни, тако да ће квадратна функија имати шпиц на прелазу са улазних позитивних вредности на негативне (Слика 4.10).

Рад архитектуре биће демонстриран на две улазне инстанце које се класификују једна за другом. Прва улазна инстанце ће бити $(1, 0, 1)$. Друга улазна инстанца ће бити $(-0.4, 0.5, 0.2)$.

Класификација за сваку инстанцу започиње у модулу RB 0. У модулу RB 0, рачуна се формула 1.2 за прва два вектора подршке. За овај модул

укупна срачуната сума се поставља на 0. Потом се, за сваки вектор подршке, раде четири корака. Први је рачунање скаларног производа улазне инстанце и вектора подршке. Други је рачунање нелинеарне функције за срачунату вредност скаларног производа. Трећи корак је множење са Лагранжовим множитељем. И на крају, четврти корак је додавање резултата множњена на суму. Након обраде првог вектора подршке резултат суме у модулу 0 биће тачно 0.125 (Прорачун 4.1). Након срачунавања за други вектор подршке резултат би требао да је приближно једнак 0.079625.

$$
\begin{aligned}
& & sum &= 0 \\
& \text{s1 1.} & (1,0,1)\cdot(0.5,0.25,0.5) &= 1 \\
& \text{s1 2.} & \frac{1}{2}1^2 &= 0.5 \\
& \text{s1 3.} & 0.5\cdot 0.25 &= 0.125 \\
& \text{s1 4.} & sum &= 0.125 \\
& \text{s2 1.} & (1,0,1)\cdot(-0.25,-0.4,-0.3) &= -0.55 \\
& \text{s2 2.} & \frac{1}{2}(-0.55)^2 &= 0.15125 \\
& \text{s2 3.} & 0.15125\cdot -0.3 &= -0.045375 \\
& \text{s2 4.} & sum = 0.125 - 0.045375 &= 0.079625
\end{aligned}
\tag{4.1}
$$

Прорачуни унутар модула RB 1 и RB 2 биће слични као прорачун унутар модула RB 0. Разлика ће бити у почетној суми. Пошто је RB 0 модул први у низу, његова почетна сума је једнака 0. Модули RB 1 и RB 2 добијају своју почетну суму од претходних модула у низу: модул RB 1 од модула RB 0, а модул RB 2 од модула RB 1. Модул RB 1 ће стога проследити вредност приближну 0.27962 до модула RB 2 (Прорачун 4.2), док ће модул RB 2 проследити вредност приближну -0.020375 до модула RB 3 (Прорачун 4.3).

$$
\begin{aligned}
& & sum &= 0.079625 \\
& \text{s3 1.} & (1,0,1)\cdot(1,-0.5,-0.2) &= -0.8 \\
& \text{s3 2.} & \frac{1}{2}(-0.8)^2 &= 0.32 \\
& \text{s3 3.} & 0.32\cdot -0.5 &= -0.16 \\
& \text{s3 4.} & sum &= -0.16 + 0.079625 = -0.080375 \\
& \text{s4 1.} & (1,0,1)\cdot(1.0,0.5,0.2) &= 1.2 \\
& \text{s4 2.} & \frac{1}{2}1.2^2 &= 0.72 \\
& \text{s4 3.} & 0.72\cdot 0.5 &= 0.36 \\
& \text{s4 4.} & sum &= 0.36 - 0.080375 = 0.27962 \\
& & sum &= 0.27962 \\
& \text{s5 1.} & (1,0,1)\cdot(-1,0,0) &= -1 \\
& \text{s5 2.} & \frac{1}{2} - 1^2 &= 0.5 \\
& \text{s5 3.} & 0.5\cdot -0.6 &= -0.3 \\
& \text{s5 4.} & sum &= -0.3 + 0.27962 = -0.020375 \\
& \text{s6 1.} & (1,0,1)\cdot(0,1,0) &= 0 \\
& \text{s6 2.} & \frac{1}{2}0^2 &= 0 \\
& \text{s6 3.} & 0\cdot -0.5 &= 0 \\
& \text{s6 4.} & sum &= 0 + -0.020375 = -0.020375
\end{aligned}
\tag{4.2}
\tag{4.3}
$$

Прорачун унутар модула RB 3 је нешто краћи него у остала три модула, јер у њему стоји само један вектор подршке и додатна вредност. Почетну вреност суме модул RB 3 добија од модула RB2. Након што заврши већ наведене кораке, овај модул још сабира додатну вредност са тренутном укупном сумом. Тиме се завршава срачунавање формуле за SVM класификатор. Као коначан резултат на излаз се прослеђује завршна сума. На основу знака резултата, зна се којој класи је придружена улазна инстанца. Коначан резултат прорачуна је 0.22962 (Прорачун 4.4), стога, првој улазној инстанци класификатор треба да додели као резултат класу 1.

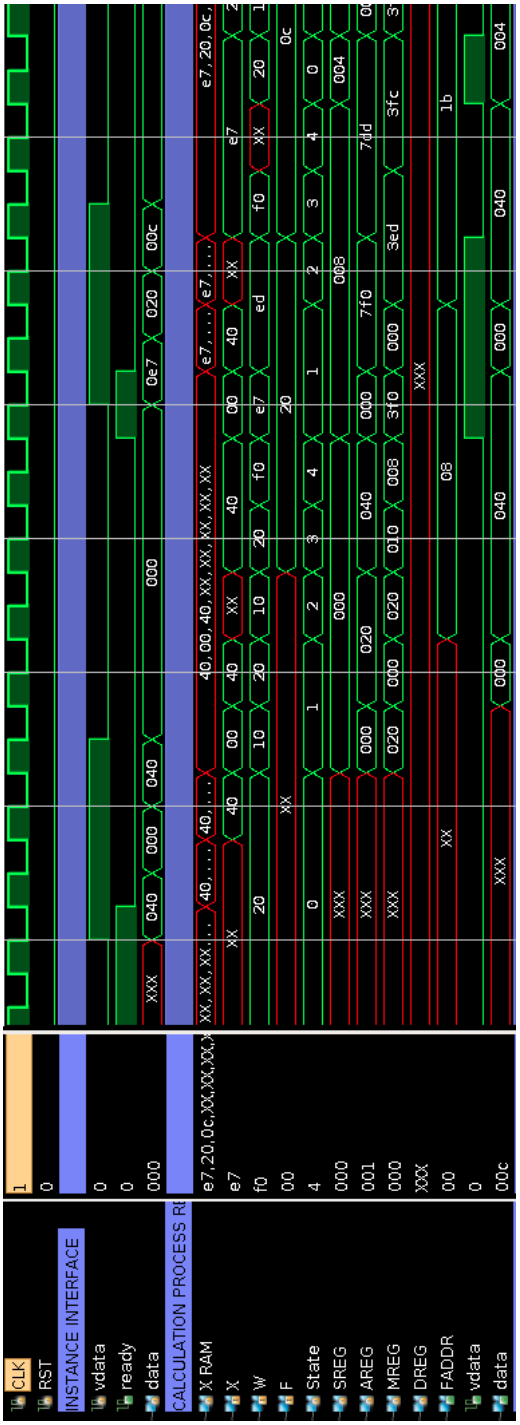
$$
\begin{aligned}
& & sum &= -0.020375 \\
\text{s7 1.} & & (1,0,1)\cdot(-0.5,0.5,-1) &= -1.5 \\
\text{s7 2.} & & \frac{1}{2} - 1.5^2 &= 1.125 \\
\text{s7 3.} & & 1.125 \cdot 0.4 &= 0.45 \\
\text{s7 4.} & & sum &= 0.45 - 0.020375 = 0.42962 \\
& & out &= sum - 0.2 = 0.22962
\end{aligned}
\tag{4.4}
$$

Сличним прорачуном за другу улазну инстанцу се добија коначна вредност суме од -0.41776. У прорачуну 4.5 су приказани резултати суме након сваког појединог вектора подршке. Пошто је за ову инстанцу коначна вредност негативна, њој се, као коначан резултат класификовања, додељује класа 2.

$$
\begin{aligned}
\text{s1} \quad & sum = 7.8125e-05 \\
\text{s2} \quad & sum = -0.0037619 \\
\text{s3} \quad & sum = -0.12279 \\
\text{s4} \quad & sum = -0.11976 \\
\text{s5} \quad & sum = -0.16776 \\
\text{s6} \quad & sum = -0.23026 \\
\text{s7} \quad & sum = -0.21776 \\
& sum = -0.41776
\end{aligned}
\tag{4.5}
$$

Архитектура прихвата улазну инстанцу преко улазних портова vdata и data. Уз помоћ излазног порта ready, може се одложити обрада улазне инстанце до тренутка када је архитектура спремна за прорачун (Слика 4.11). Трансфер података започиње када су оба порта vdata и ready постављени на '1'. Спољни систем не сме да стаје са прослеђивањем улазне инстанце једном када је трансфер започео.

Пошто улазна инстанца има три атрибута, цео трансфер инстанце траје три циклуса. За разлику од DT-ова архитектура не може да прими две инстанце једну за другом (Слике 4.11 и 3.14). То није могуће зато што обрада једне инстанце унутар модула RB може да траје знатно дуже за SVM-ове него за DT-ове. Трајање класификације инстанце, унутар једног модула RB, је у случају када архитектура ради као SVM пропорционално, осим броју атрибута, и броју вектора подршке унутар модула RB. Када архитектура ради као DT трајање класификације унутар једног RB модула пропорционално је само броју атрибута. Модул RB започиње класификацију чим се први атрибут прве инстанце смести



Слика 4.11: Класификовање прве инстанце унутар модула RB 0 за SVM

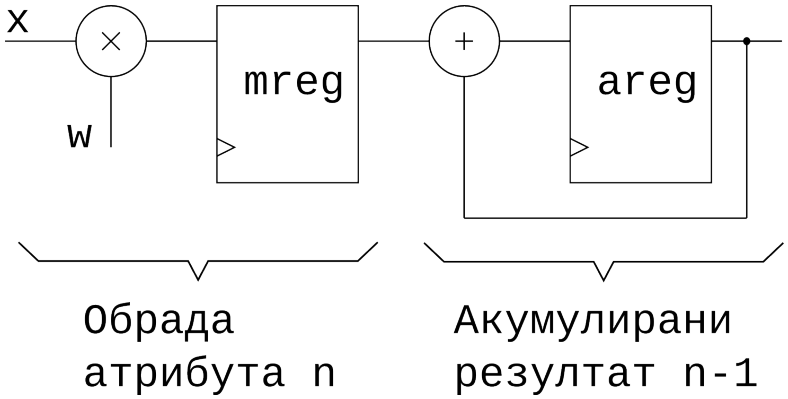
у меморију X RAM. Почетак класификовања унутар модула CALC може се уочити променом тренутног стања аутомата унутар модула. Промена вредности стања са 0 (“IDLE”) на 1 (“STEP_1”) значи да је модул почео класификацију инстанце која се тренутно смешта у меморију X RAM тренутног модула RB. Стања коначног аутомата, унутар модула CALC, су кодована бинарно у RTL моделу.

- IDLE = 0
- STEP_1 = 1
- STEP_2 = 2
- STEP_3 = 3
- STEP_4 = 4
- STEP_5 = 5

Док је аутомат унутар модула CALC у стању “IDLE”, он чека да пристигне први атрибут улазне инстанце. Чим се то деси аутомат прелази у стање “STEP_1”. Прорачун се, када класификатор ради као SVM, увек започиње од адресе 0 и за меморију W RAM као и за меморију X RAM. Након стања “IDLE” у ланцу за обраду података, важећа вредност ће се налазити у регистру MREG. Вредност регистра SREG се за овај модул, RB 0, пошто је он први у низу, иницијализује на 0.

Када је машина стања унутар модула CALC прешла у стање “STEP_1”, прорачун скаларног производа већ је започео. Производ вредности првог атрибута улазне инстанце и прве вредности првог вектора подршке већ је срачунат и резултат се налази у регистру MREG. Први ниво ланца за обраду већ је активан (Слика 4.12). У овом стању ће модул CALC остати онолико циклуса колико улазна инстанца има атрибута, умањено за вредност 1. У овом примеру то значи да ће аутомат остати у овом стању 2 циклуса (Слика 4.11). Док се налази у овом стању, оба нивоа ланца обраде су активна. Први ниво ланца обраде који се завршава са регистром MREG обрађује n-ти атрибут улазне инстанце и n-ти скалар вектора подршке. Други ниво ланца обраде, који се завршава са регистром AREG акумулира n-1 резултат множења. Треба напоменути да, када класификатор ради као SVM у зависности од кернела класификатора, овај ланац може бити и другачије конфигурисан и пролаз кроз стања аутомата може бити другачији. За пример рада класификатора као SVM узет је полиномијални кернел па

ланац за обраду има илустровани изглед. Уколико би SVM радио са, рецимо, радијалним кернелом ланац би имао три нивоа. Овакав ланац за обраду ће бити илустрован код ANN класификатора. Машина стања модула CALC ће прећи у наредно стање "STEP_2" у тренутку када се на улазу X појави последњи атрибут улазне инстанце, а на улазу W се појави последња вредност тренутног вектора подршке. У овом случају ће на улазу X бити вредност 1 (0x40), а на улазу W ће бити вредност 0.5 (0x20). Ово значи да се у стању "STEP_1" рачунање скаларног производа још није завршило. При уласку у стање "STEP_2" у регистру MREG ће се налазити помножена вредност улаза X и W.



Слика 4.12: Ланац за рачунање у модулу CALC у стању "STEP_1" када класификатор ради као SVM.

У стању "STEP_2" завршиће се рачунање скаларног производа. Та вредност се може видети, тек у наредном циклусу, на регистру AREG (Слика 4.11). Да би се прорачун убрзао, вредност са сабирача се шаље на излаз FADDR, тако да ће се наредни циклус на улазу F појавити вредност израчунате нелинеарне функције. У овом стању се и завршава рачун скаларног производа и рачун нелинеарне функције. Обе вредности ће бити важеће тек у наредном стању "STEP_3". Уједно, у регистру MREG се чува вредност Лагранжовог множитеља који је преузет из меморије W RAM и налази се на улазу W док је аутомат у стању "STEP_2". Активна врста у меморији W RAM се мења на наредну врсту.

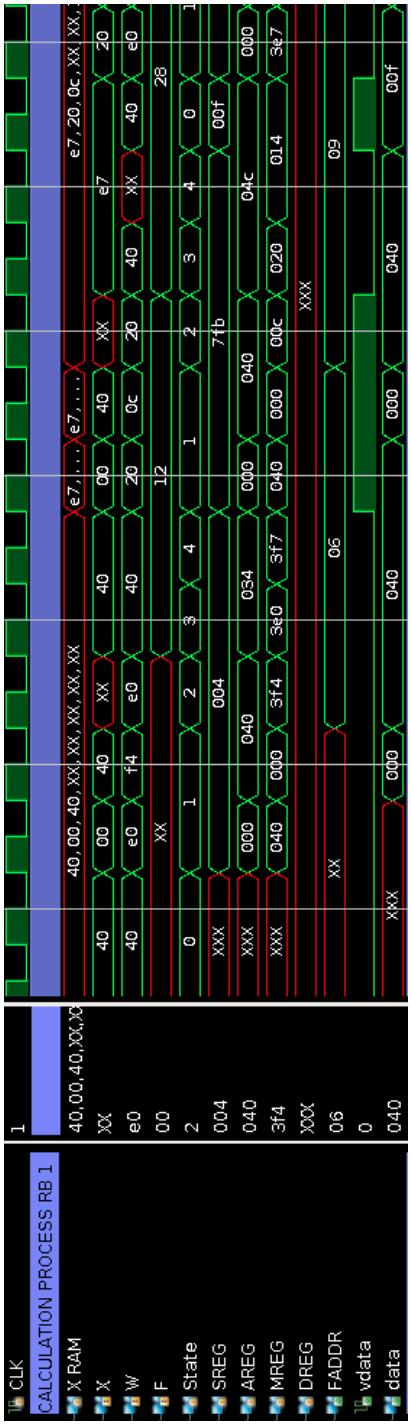
Док је аутомат у стању "STEP_3" у регистру AREG се налази вредност скаларног производа. У нашем примеру та вредност је 1 (0x40) (Слика 4.11). Вредност срачунате нелинеарне функције се налази на улазу F, што је у овом примеру 0.5 (0x20). Вредност Лагранжовог множитеља је смештена у регистар MREG и за наш пример износи 0.25 (0x10). У овом стању аутомата множи се вредност резултата нелинеарне функције и Лагранжовог множитеља. Резултат ће бити валидан наредни

циклус и налазиће се у регистру MREG. Аутомат остаје у овом стању само један циклус и пребацује се у стање “STEP_4”.

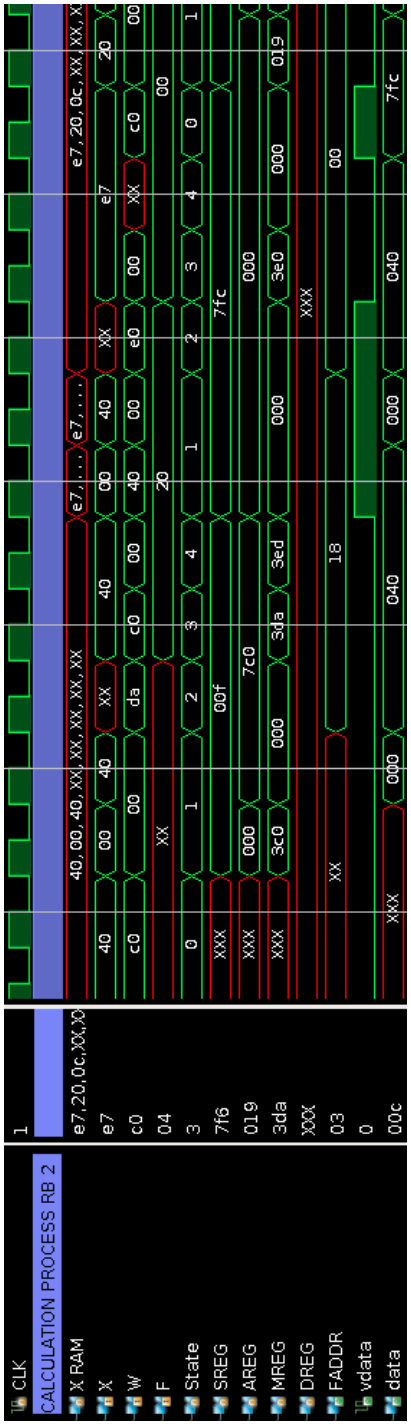
У стању “STEP_4”, аутомат додаје вредност из регистра MREG на тренутну вредност суме које се налази у регистру SREG. Вредност се опет чува у регистру SREG. Вредност која је сачувана у регистру SREG се види у наредном циклусу. За овај пример она износи 0.125 (0x08) (Слика 4.11). За први вектор подршке израчуната вредност не одступа од тачне вредности (Прорачун 4.1). Пошто у модулу RB 0, постоји још један вектор подршке прелази се на већ описану процедуру почевши од стања “IDLE”. У општем случају машина стања модула CALC ће циклично пролазити кроз стања “STEP_1”, “STEP_2”, “STEP_3” и “STEP_4”. Због ове чињенице, јасно је да је трајање прорачуна директно пропорционално броју вектора подршке. Аутомат се у стању “STEP_1” задржава пропорционално броју атрибута улазне инстанце. На основу рада аутомата, јасно је да је трајање прорачуна у случају када класификатор ради као SVM директно пропорционално и броју вектора подршке као и броју атрибута улазне инстанце. Крај прорачуна унутар једног модула, који није последњи у низу, када архитектура ради као SVM, може се одредити на основу преласка аутомата унутар модула CALC са стања “STEP_4” на стање “IDLE”.

При другом пролазу кроз стања “STEP_1” и “STEP_2”, аутомат ће проследити вредности атрибута улазне инстанце на излазни порт data и учиниће вредност порта важећом, постављањем порта vdata на ’1’ (Слика 4.11). Аутомат ово у општем случају ради при последњем пролазу аутомата кроз ова два стања приликом класификовања неке инстанце. На тај начин, наредни RB модул добија атрибуте улазне инстанце. Последњи пролаз аутомата кроз ова стања догађа се када се ради прорачун за последњи вектор подршке смештен у тренутном модулу RB. Када обради све векторе подршке и заврши у стању “STEP_4”, аутомат прослеђује збир тренутне суме и резултата обраде улазне инстанце и последњег вектора подршке на излазни порт data. Притом се, наравно, овај порт начини важећим. За наш пример, модул RB 0, је послао резултат 0.0625 (0x04) до наредног модула RB 1. Резултат прорачуна се након обраде другог вектора подршке разликује од тачног резултата 0.079625. До овога је дошло, наравно, због ограничене прецизности са којом архитектура ради.

Трајање класификације архитектуре када ради као SVM не зависи од конкретне улазне инстанце, као што је то био случај када архитектура ради као DT. Сви кораци су исти и нема специјалног кодовања унутар меморије, као што је то био случај када је архитектура радила као DT. Стога, сви већ описани кораци су потпуно исти за прву инстанцу



Слика 4.13: Класификовање прве инстанце унутар модула RB 1 за SVM



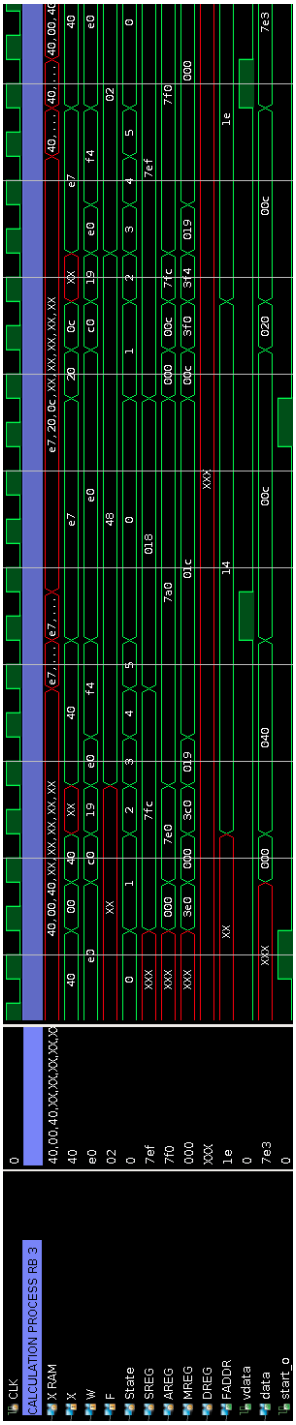
Слика 4.14: Класификовање прве инстанце унутар модула RB 2 за SVM

и унутар модула RB 1 и RB 2 (Слике 4.13 и 4.14). Једина разлика је што је у модулу RB 0 регистар SREG био иницијализован на вредност 0, а у случају модула RB 1 и RB 2 овај регистар се поставља, на почетку прорачуна, на вредност која је добијена од предходног модула. Након модула RB 1 резултат класификовања је 0.23438 (0x0F). Према прорачуну 4.3 тачан резултат је 0.27962. Након модула RB 2 резултат је -0.062500 (0xFC) док је према прорачуну 4.4 тачан резултат -0.020375. Као што се види прорачун унутар архитектуре прати тачан прорачун уз извесно одступање, које зависи од броја бита који је узет за представљање бројева. Више о анализи потребног броја бита за представљање бројева може се наћи у поглављу 7.

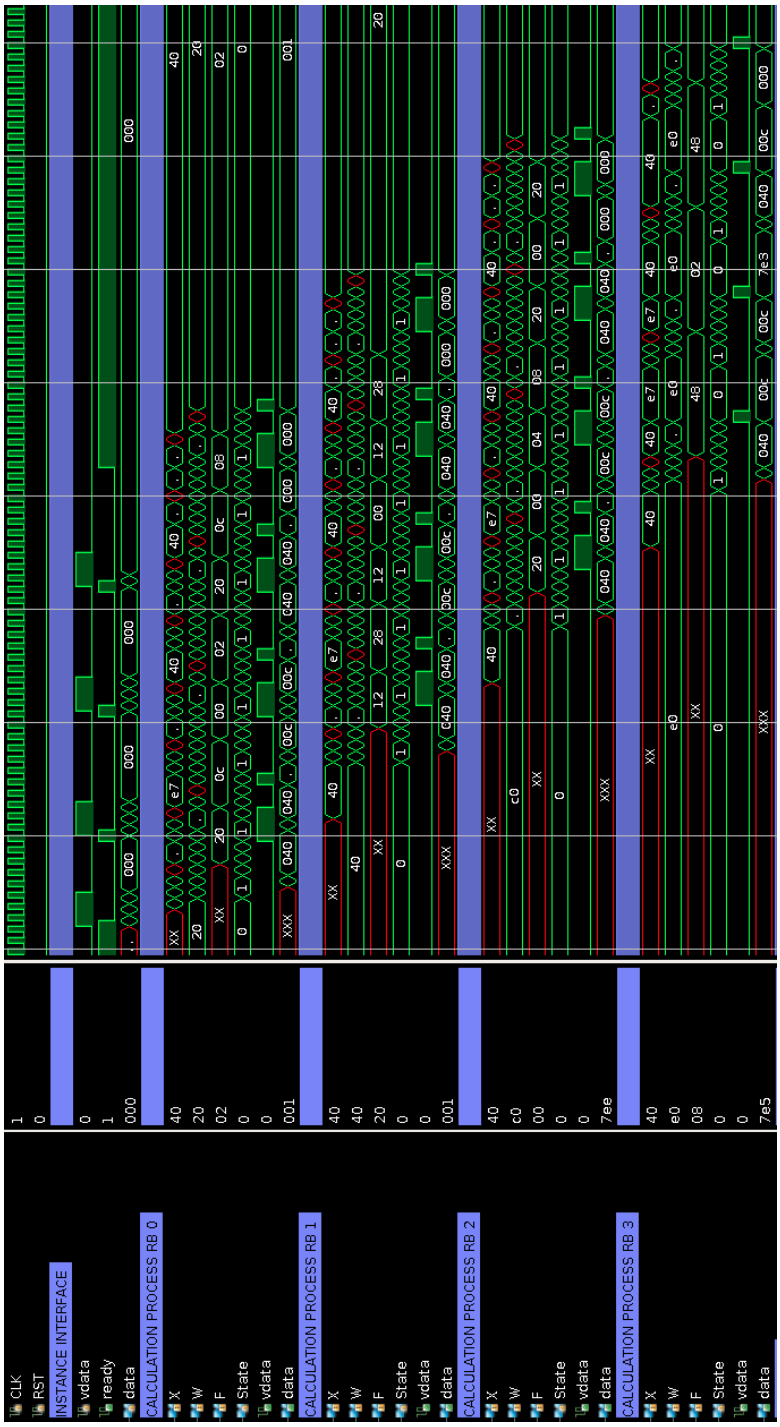
Прорачун унутар модула RB 3 је нешто краћи него у остала три модула. Наравно, то је последица тога што се у овом модулу налази само један вектор подршке. Овај модул, пошто је последњи у низу, не поставља излазни порт data на вредности улазне инстанце (Слика 4.15). Након њега се налази спољни систем, стога је на излазне портове потребно поставити само коначну вредност рачунања. Овај модул ради исто као и претходни све док ради прорачун до последњег вектора подршке. Овај модул у меморији W RAM након последњег вектора подршке, садржи додатну вредност. Стога у цикусу, када на крају прорачуна аутомат модула CALC дође до стања "STEP_4", прелази у стање "STEP_5", у ком сабира вредност регистра SREG са додатном вредношћу и на тај начин добија коначну вредност класификовања. Та вредност се пребацује на излазни порт data уз постављање порта vdata на вредност '1'. Другим речима стање "STEP_5" је достижно само за последњи модул RB у низу, када архитектура ради као SVM. Након прелаза аутомата модула CALC из стања "STEP_5" у стање "IDLE" архитектура је завршила класификовање једне инстанце. За прву инстанцу резултат класификовања је 0.18750 (0x0C) што се разликује од тачно израчунате вредности 0.22962. Но, пошто SVM упоређује ову вредност са 0 и придружује улазној инстанци класу на основу знака резултата, у оба случаја ће улазној инстанци бити додељена класа 1, као резултат предикције.

За другу улазну инстанцу резултат класификовања је -0.45312 (0xE3) (Слика 4.15). Као тачан резултат класификације треба да се добије коначна сума -0.41776. Опет се резултати разликују али је резултат класификовања идентичан, пошто ће у оба случаја улазној инстанци бити додељена класа 2.

У досадашњем разматрању се говорило о детаљима прорачуна унутар појединих модула архитектуре. Оно на шта треба посебно скренути пажњу је, да је организација архитектуре таква да је оптимизована



Слика 4.15: Класификовање прве инстанце унутар модула RB 3 за SVM



Слика 4.16: Поглед на прорачун свих модула на 4 улазне инстанце SVM класификатора

проточност. Захваљујући томе се, и поред великог кашњења за добијање првог резултата, добија велико убрзање када улазне инстанце класификују једна за другом. То значи да, док модул RB 3 ради прорачун за прву инстанцу, истовремено, модул RB 2 ради прорачун за другу инстанцу, модул RB 1 обрађује трећу инстанцу док модул RB 0 прорачунава за четврту инстанцу (Слика 4.16). Оно што се може уочити када архитектура ради као SVM, за разлику од DT-ова, јесте предвидљивост са којом долазе резултати прорачуна на излазне портове. Може се уочити и да прорачун за SVM-ове траје доста дуже, пошто је за SVM класификовање потребно урадити знатно више операција. Треба нагласити и да архитектура, када ради као SVM класификатор, готово увек има добитак у убрзању ако се у низ додају нови RB модули. Што је више RB модула, то се мање вектора подршке налази у свакоме од њих појединачно, па сваки модул рачуна краће, што резултује у већем убрзању.

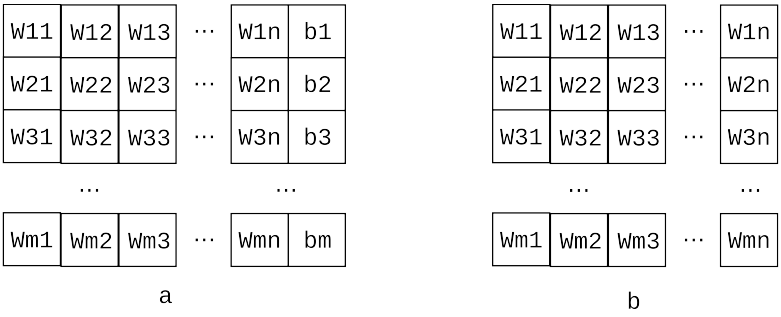
Глава 5

Конфигурација универзалне архитектуре као ANN

5.1 Архитектура као ANN

Када RMLC архитектура ради као ANN класификатор, активни RB модули рачунају излазне вредности свих неурона у једном слоју. У случају да RB модул није први у једнодимензионом низу, X RAM тада садржи излазне вредности добијене од претходног RB модула. Меморија X RAM првог RB модула садржи вредности атрибута улазне инстанце. Меморија F RAM садржи одбирке нелинеарне функције коју користи ANN класификатор. У случају када се ради о излазном слоју радијалног ANN класификатора, ова функција је функција идентитета. Пошто RMLC архитектура може да ради као две врсте ANN класификатора, постоје две могућности за распоред вредности унутар W RAM-а. Када RMLC архитектура ради као MLP ANN, тада W RAM меморија садржи тежински вектор **w** и додату вредност b, за сваки неурон тренутног слоја (Слика 5.1 a). Број врста унутар меморије W RAM зависи од броја неурона у тренутном слоју ANN класификатора. У случају да RMLC архитектура ради као радијални ANN класификатор, меморија W RAM за неуроне из скривеног слоја садржи само централни вектор **w** (Слика 5.1 b) из формуле 1.5. За неуроне у излазном слоју, меморија W RAM садржи само тежински вектор **w** из формуле 1.7.

Приликом прорачуна модул CALC пролази кроз три фазе. У зависности од типа ANN класификатора постоје две могућности у фази 1 (Слика 5.2 1a, 1b) и фази 2. (Слика 5.2 2a, 2b). У фази 1 се, у случају када RMLC архитектура ради као класификатор MLP ANN, рачуна скаларни производ између садржаја меморије X RAM и тежинског



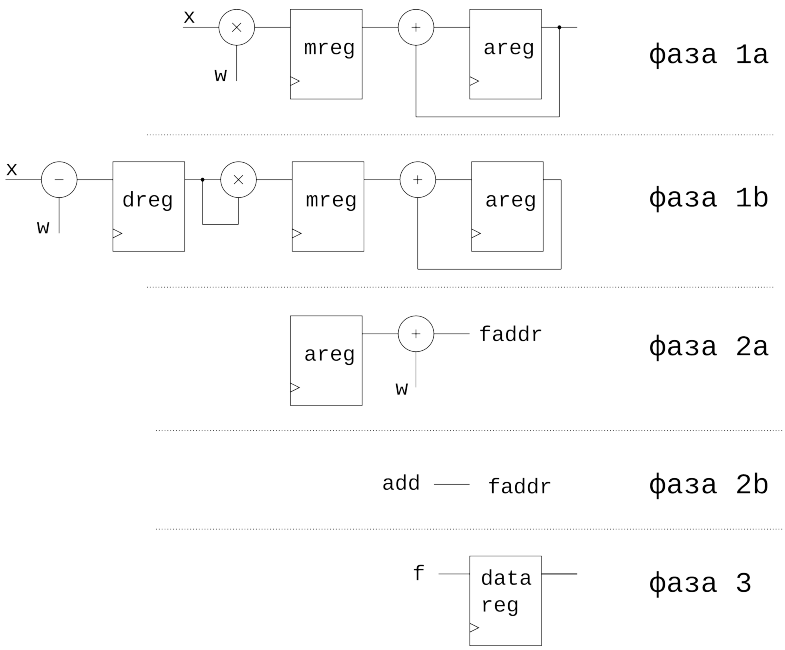
Слика 5.1: Садржај W RAM-а у случају ANN-ова

вектора **w** (Слика 5.2 1a). У случају када RMLC архитектура ради као радијални ANN класификатор прво се рачуна разлика вектора унутар меморије X RAM и централног вектора **w**, потом квадрирање па акумулација (Слика 5.2 1b). У фази 2, у случају MLP ANN конфигурације, додатна вредност b се сумира са резултатом скаларног производа па се резултат прослеђује на излаз faddr (Слика 5.2 2a). Када је архитектура у конфигурацији радијалног ANN класификатора, сума квадрираних вредности се само прослеђује на излаз faddr. (Слика 5.2 2b). Излаз faddr одређује адресу меморијске локације F RAM-а чији садржај ће се преузети. Вредност прочитана из F RAM меморије представља срачунату вредност одговарајуће активационе функције. У фази 3 се садржај прочитан из меморије F RAM смешта у регистар DATAREG. Да би се израчунала вредност једног неурона, потребан је један пролазак кроз фазе 1, 2 и 3. За сваки неурон у слоју мреже понављају се ове три фазе.

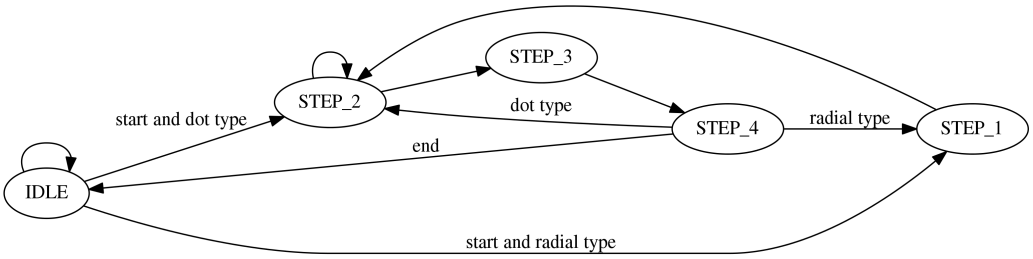
Машина стања модула CALC има 4 или 5 стања (Слика 5.3). Када дигитална архитектура RMLC ради као класификатор MLP ANN тада се фазе 1 и 2 реализује у стањима "STEP_2" и "STEP_3" док се у случају радијалних ANN класификатора фаза 1 имплементира у стањима "STEP_1", "STEP_2" и "STEP_3". Фаза 3 је имплементирана у стању "STEP_4".

5.2 Пример рада архитектуре као ANN-а

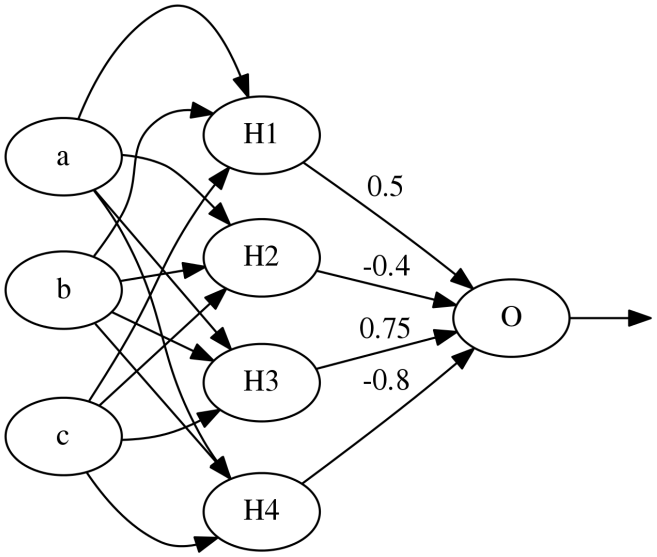
Рад архитектуре RMLC као ANN класификатора илустроваћемо на примеру ANN мреже са радијалним кернелом (Слика 5.4). Радијална функција коју ће архитектура користити биће $e^{-(\|\mathbf{x}-\mathbf{c}\|)^2}$. У радијалној функцији улазна инстанца је означена са **x**, а централни вектор са **c**.



Слика 5.2: Фазе у прорачуну - ANN



Слика 5.3: Машина стања модула CALC као ANN класификатора



Слика 5.4: Неуронска мрежа за илустративни пример

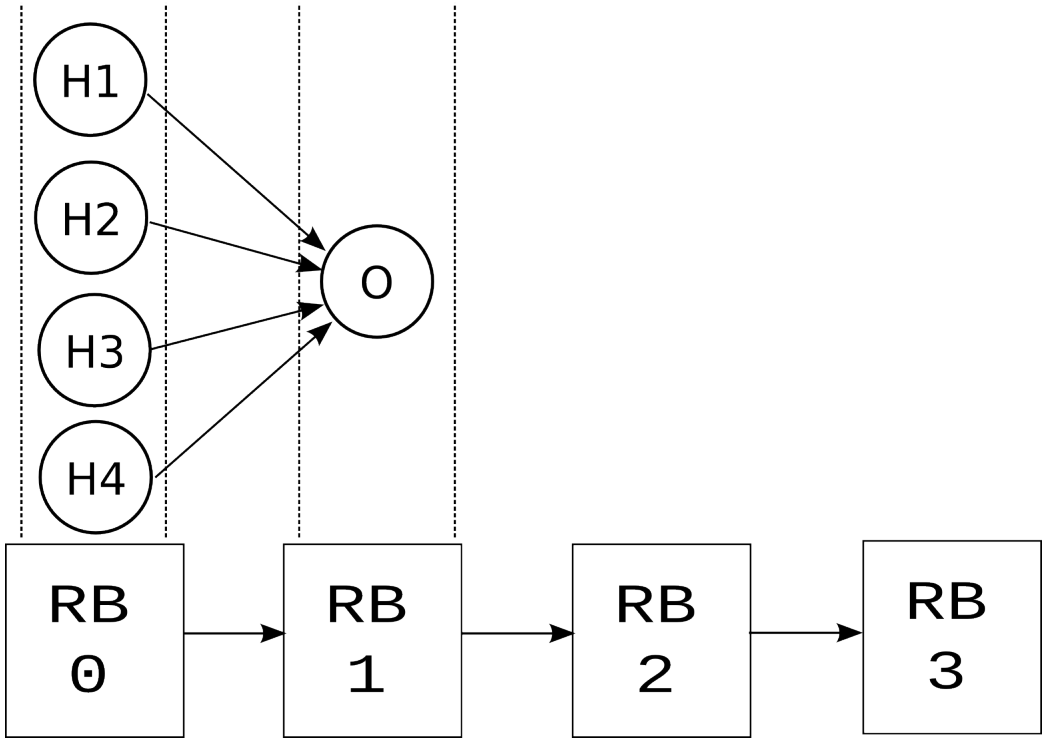
Неуралне мреже са радијалним кернелом имају три слоја: улазни, скривени и излазни. Као што је већ напоменуто, улазни слој представљају улазни атрибути. Они су означени словима a, b и c. Улазни атрибути се прослеђују до свих неурона скривеног слоја. Број неурона у скривеном слоју за овај илустративни пример је 4. Неурони у скривеном слоју су означени почетним словом "H" и бројем који их јединствено идентификује. На пример, "H1" означава неурон у скривеном слоју са идентификационим бројем 1. Излазни слој у овом примеру садржи неурон који је означен словом "O".

За ову конкретну мрежу, централни вектори са којима се ради прорачун у неуронима су:

- H1: (0.5 0 0.5)
- H2: (0.3 -0.2 0.3)
- H3: (0.4 -0.1 0.25)
- H4: (0.0 -0.15 0.6)

Као и у преходним примерима, архитектура RMLC ће бити конфигурисана са 4 RB модула (Слика 5.5). Када архитектура ради као неуронска мрежа, цели слојеви мреже се пресликавају у један RB модул. Улазни слој неуронске мреже представља улазне атрибуте па се он не имплементира унутар архитектуре. Слојеви се пресликавају на

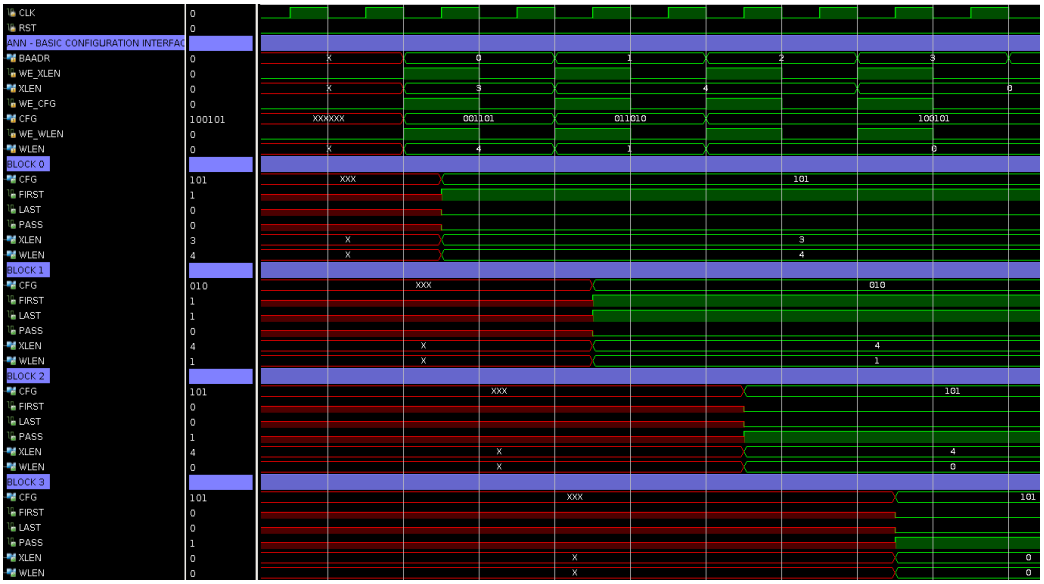
модуле архитектуре почевши од првог скривеног слоја. То значи да ће се сви неурони јединог скривеног слоја мреже из примера: “H1”, “H2”, “H3” и “H4” пресликати у модул RB 0. Неурон “O” из излазног слоја ће се пресликати у модул RB 1”. За ову мрежу, само прва два модула RB ће ефективно учествовати у прорачуну резултата, док ће последња два само пропуштати резултате прорачуна.



Слика 5.5: Пресликавање неурноске мреже на архитектуру

Основни конфигурациони регистри (cfg, xlen, wlen, first, last, pass) су, слично као и у претходним примерима иницијализовани сви у једном циклусу за један RB модул (Слика 5.6). Вредност улазног порта baddr одређује који се блок конфигурише, па се види да су блокови програмирани у растућем редоследу.

Регистар xlen има другачије значење када архитектура ради као неуронска мрежа за разлику од случајева када архитектура ради као стабло или SVM. За слој који је пресликан у први RB модул, вредност регистра xlen има исто значење као и у претходним примерима. Ова вредност означава колико атрибута има улазна инстанца. Но, за остале слојеве неуронске мреже вредност овог регистра означава број веза неурона у тренутном слоју са неуронима у претходном слоју.



Слика 5.6: Основни конфигурациони интерфејс за ANN

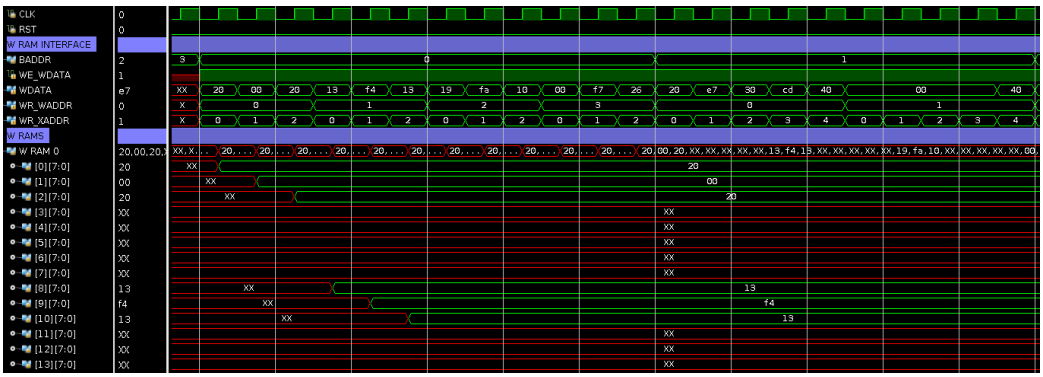
Вредност овог регистра може бити различита у свим модулима који имплементирају неки слој мреже и вредност тог броја зависи од структуре мреже. Стога, за овај пример, вредност уписана у регистар xlen модула RB 0 биће једнака 3, пошто слој мреже пресликан у овај модул ради директно са атрибутима улазне инстанце. Вредност уписана у регистар xlen унутар модула RB 1 је 4. Једини неурон, "О", унутар излазног слоја, који је пресликан у овај модул повезан је са 4 неурона из претходног слоја мреже, зато је вредност 4 уписана у овај регистар. За последња два модула архитектуре вредност унутар регистра xlen није од значаја пошто су ова два модула конфигурисана да само пропуштају резултат прорачуна.

Када архитектура ради као неуронска мрежа и регистар cfg може имати различите вредности у зависности од модула до модула. За модул RB 0 из примера вредност у регистру cfg је "101". То значи да тај модул ради као неуронска мрежа са радијалним кернелом (Слика 5.6). Интересантна је вредност регистра cfg за модул RB 1. Вредност у том регистру је "010". Ова вредност означава да модул ради као SVM. Ако се пажљиво погледа какав је прорачун унутар различитих неурона неуронске мреже са радијалним кернелом, може се приметити да излазни неурон има другачији прорачун него остали неурони у мрежи. Овај прорачун је исти као прорачун модула код SVM класификатора па је овај модул тако и конфигурисан. За модул RB 0, вредност у регистирма

first, last и pass су исто очекиване. Вредност унутар регистра first је ’1’ пошто је овај модул први у низу. Вредност у друга два регистра су ’0’ пошто овај модул није последњи и не пропушта резултат. Но вредност унутар регистра first за модул RB 1 је исто ’1’ што није очекивана вредност. Ова вредност је постављена на ’1’ зато што се само у том случају вредност суме поставља на 0 приликом прорачуна модула који ради као SVM. О овоме ће још бити речи када се буде описивао сам прорачун. Вредност унутар регистра last за модул RB 1 је ’1’ пошто је овај модул последњи активан модул архитектуре када се имплементира оваква мрежа. Последња два модула имају уписану вредност ’1’ унутар регистра pass, тако да ће они само пропуштати добијене вредности до наредног модула, односно до излаза мреже.

Регистар wlen означава број неурона у слоју мреже који имплементира модул. Четири неурона се налазе унутар модула RB 0 па је зато уписана вредност 4 унутар регистра wlen овог модула (Слика 5.6). Само један неурон се налази унутар модула RB 1 па је уписана вредност 1 у регистар wlen модула RB 1. Вредности у овом регистру у последња два модула нису од значаја, пошто они не раде прорачун.

Меморија W RAM се програмира модул по модул и у зависности од броја неурона који су смештени унутар модула потребан је одговарајући број циклуса да би се ова меморија иницијализовала (Слика 5.7).



Слика 5.7: Конфигурисање меморије WRAM једног ANN класификатора

Улазни портрови за конфигурисање меморије W RAM су we_data, wr_waddr и wr_xaddr и контролишу се на већ описани начин, као код класификатора DT-ова и SVM-ова.

Када се ради са ANN класификаторима са радијалним кернелом за прорачун у скривеном слоју мреже, потребан је централни вектори а за прорачун у излазном слоју потребни су коефицијенти са којима се

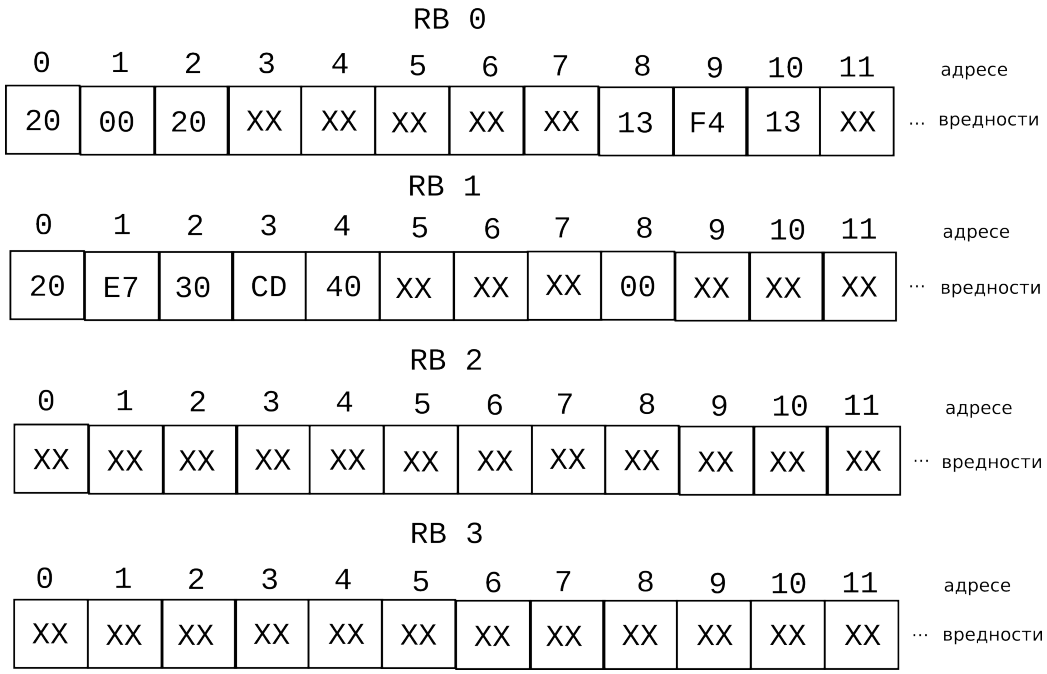
множе резултати из скривеног слоја. За представљање бројева се, као и до сада, користи репрезентација са фиксном тачком са два бита у целом делу броја и шест бита у разломљеном делу. Негативни бројеви су представљени као комплемент броја два. Скалари централних вектора ће бити пребачени у репрезентацију фиксном тачком коришћењем одсецања. Репрезентација за неке конкретне вредности је приказана у прорачуну 5.1. Уместо одсецања за већу прецизност могло је бити коришћено заокруживање. За ове примере, за коначан резултат класификације и ово ће бити довољно прецизно а само пребацивање је једноставније. За први неурон у скривеном слоју, "H1" , централни вектор је (0.5, 0, 0.5). У меморију W RAM ће бити уписане вредности 0x20, 0, 0x20 (Слика 5.7). То је све што се уписује у меморију за неуроне скривеног слоја. Пошто је адресна магистрала wr_addr широка три бита, информације о првом неурону се смештају у првих осам локација. Пошто је за смештање централног вектора потребно само три локације, осталих пет ће остати неиницијализоване. Потом се у меморију смешта централни вектор другог неурона, "H2" у скривеном слоју, (0.3 -0.2 0.3). Те вредности ће бити смештене на адресама 8, 9 и 10, а оне износе 0x13, 0xf4, 0x13. За преостала два неурона уписују се њихови централни вектори, (0.4 -0.1 0.25) и (0.0 -0.15 0.6) представљени у одговарајућој репрезентацији: 0x19, 0xfa, 0x10 и 0, 0xf7 и 0x26.

$$
\begin{array}{lll}
0.5 & = 00.100000 = 0\text{x}20 & \\
0 & = 00.000000 = 0 & \\
0.3 & \approx 00.010011 = 0\text{x}13 & = 0.29688 \\
\text{-}0.2 & \approx 11.110100 = 0\text{xF}4 & = -0.18750 \\
0.4 & \approx 00.011001 = 0\text{x}19 & = 0.39062 \\
\text{-}0.1 & \approx 11.111010 = 0\text{xFA} & = -0.093750 \\
0.25 & = 00.010000 = 0\text{x}10 & \\
\text{-}0.15 & \approx 11.111110 = 0\text{xF}7 & = 0.14062 \\
0.6 & \approx 00.100110 = 0\text{x}26 & = 0.59375 \\
\text{-}0.4 & \approx 11.100111 = 0\text{xE}7 & = -0.39062 \\
0.75 & = 00.110000 = 0\text{x}30 & \\
\text{-}0.8 & \approx 11.001101 = 0\text{xCD} & = -0.79688
\end{array}
\tag{5.1}
$$

За излазни слој, за неурон "O", у меморију W RAM потребно је сместити све коефицијенте са којима је потребно помножити излазе неурона из скривеног слоја. Као што се може видети (Слика 5.4), ти коефицијенти су 0.5, -0.4, 0.75, -0.8. У меморију W RAM модула RB 1 у

прву врсту од адресе 0 ће бити уписане репрезентације фиксном тачком ових реалних бројева: 0x20, 0xe7, 0x30, 0xcd (Слика 5.7). Пошто је овај модул програмиран као да ради прорачун за SVM на наредну локацију у меморији је потребно ставити неку вредност за Лагранжов множитељ која неће утицати на прорачун. Наравно, за ту вредност је одабран број 1. Пошто је ово уједно и последњи модул, а он ради као да је SVM класификатор у питању, потребно је ставити неку додатну вредност која неће утицати на резултат. За ту вредност је наравно одабрана 0. Она је смештена у другу врсту унутар модула RB 1.

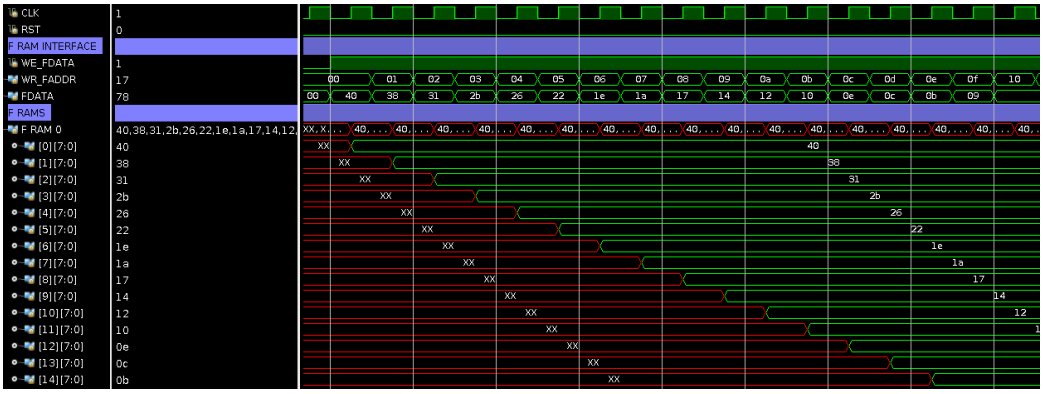
У меморији W RAM која је обична секвенцијална меморија са случајним приступом матрична структура је добијена тако што су ређане врста по врста у меморију (Слика 5.8). Као што је већ напоменуто, известан број локација ће остати са садржајем који може имати произвољну вредност. Вредност локација које нису иницијализоване означена је са XX. Садржају тих локација архитектура никада неће приступити.



Слика 5.8: Садржај меморије W RAM за пример ANN-а

Меморија F RAM садржи одбирке функције потребне за прорачун кернела. Архитектура смешта 32 одбирка функције па је исто толико циклуса потребно да би се ова меморија иницијализовала (Слика 5.9).

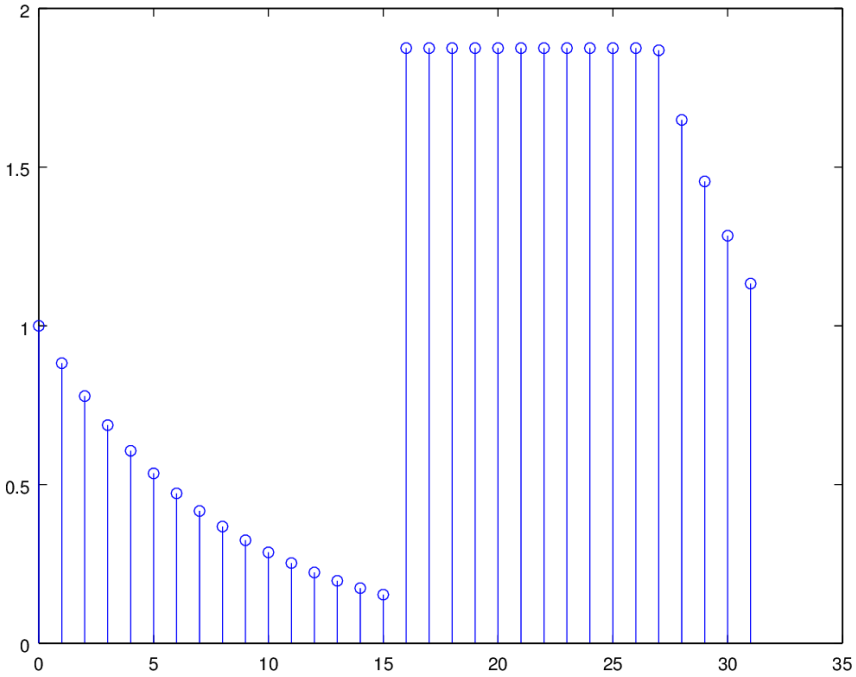
Као и у претходним примерима за конфигурисање меморије F RAM користе се улазни портови baddr, we_fdata, fdata и wr_faddr, на начин који је већ описан.



Слика 5.9: Конфигурациони интерфејс за F RAM за функције коришћене за ANN мрежу из примера

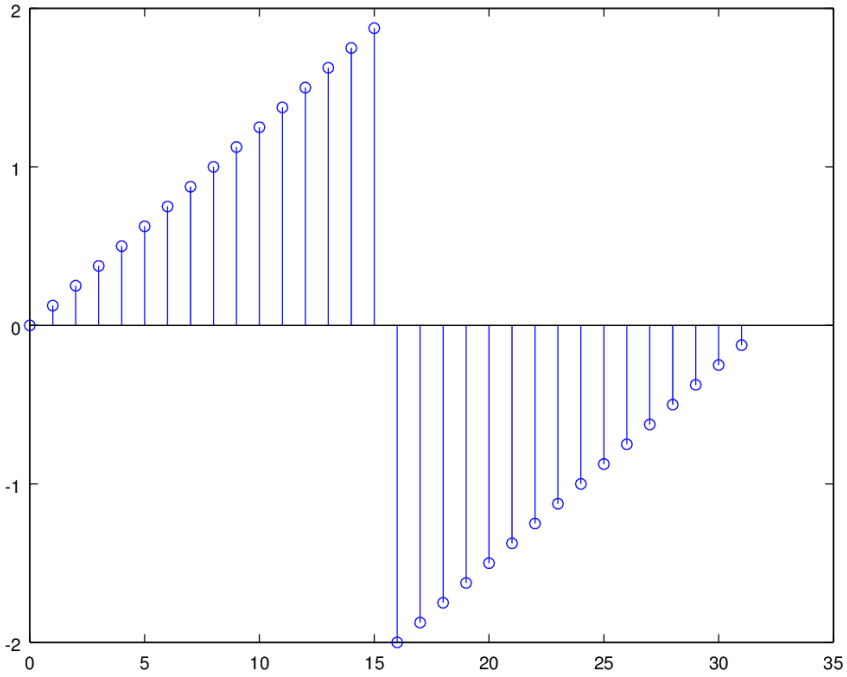
Као што је већ речено радијални кернел који мрежа у примеру користи је $e^{-(\|\mathbf{x}-\mathbf{c}\|)^2}$. Овај кернел се рачуна само у скривеном слоју неуралне мреже. Да би архитектура могла да срачуна вредност овог кернела у меморији F RAM, модула RB0, треба сместити одбирке функције e^{-t}. У модулу RB 1 се множе вредности неурона скривеног слоја са одговарајућим коефицијентима и рачуна се њихова сума. Стога у овом модулу треба сместити функцију која неће мењати вредност улаза. Та функција је функција идентитета, $f(x) = x$. У модулима RB 2 и RB 3 није битно какве ће се вредности налазити у меморији F RAM. У овом примеру је приказан случај који се разликује од претходна два примера по томе што се функције које се налазе у меморији F RAM разликују.

У модулу RB 0, меморија F RAM ће бити адресирана вредношћу која се добије рачунањем $(\|\mathbf{x} - \mathbf{c}\|)^2$. Ову вредност ће срачунати архитектура, заокружиће је на 5 бита и потом ће послати као адресу за меморију F RAM. Пошто је улазна вредност функције смештене у F RAM увек позитивна није битно које ће вредности бити смештене за негативне улазне вредности (Слика 5.10). Негативне улазне вредности се налазе у делу меморије који се адресира са адресама које у свом MSB биту имају вредност '1'. Из тога следи да је неважно какви су одбирци од 16-тог до 31-ог. На адресама од 0 до 15 се налазе одбирци функције e^{-t} за позитивне вредности. У модулу RB 1 се налазе одбирци функције



Слика 5.10: Одбирци функције унутар F RAM-а за ANN класификатор из примера у модулу RB 0

идентитета и то је исти случај као у примеру код стабала одлука (Слика 5.11).



Слика 5.11: Одмерци функције унутар F RAM-а за ANN класификатор из примера у модулу RB 1

Рад архитектуре, као и у претходним примерима, ће бити демонстриран на две улазне инстанце које се класификују једна за другом. Прва улазна инстанца ће бити $(1, 0, 1)$, док ће друга улазна инстанца бити $(-0.4, 0.5, 0.2)$.

Одређивање класе улазне инстанце почиње у модулу RB 0, који садржи скривени слој неуронске мреже. За сваки неурон потребно је срачунати радијалну функцију и вредност је потребно проследити до следећег модула у низу. Да би се срачунала вредност радијалне функције потребно је проћи кроз неколико корака. Први корак је да се израчуна разлика улазне инстанце и централног вектора. На тај начин се добија вектор разлике. Други корак је да се израчуна сума квадрата скалара вектора разлике. Трећи корак је израчунавање природног експонента негативне вредности израчунате суме. Прорачун унутар модула RB 1 је

приказан у прорачуну 5.2. У прорачуну је са леве стране приказано име неурона и корак прорачуна. Након срачунатих кернела за сваки неурон, модул RB 0, за прву инстанцу, ће проследити модулу RB 1, приближне вредности тачним вредностима 0.60653, 0.36059, 0.39357 и 0.30651.

$$
\begin{aligned}
&\text{H1 1.} && (1,0,1)-(0.5,0,0.5)=(0.5,0,0.5)\\
&\text{H1 2.} && (\|(0.5,0,0.5)\|)^2=0.5\\
&\text{H1 3.} && e^{-0.5}=0.60653\\
&\text{H2 1.} && (1,0,1)-(0.3,-0.2,0.3)=(0.7,0.2,0.7)\\
&\text{H2 2.} && (\|(0.7,0.2,0.7)\|)^2=1.02\\
&\text{H2 3.} && e^{-1.02}=0.36059\\
&\text{H3 1.} && (1,0,1)-(0.4,-0.1,0.25)=(0.6,0.1,0.75)\\
&\text{H3 2.} && (\|(0.6,0.1,0.75)\|)^2=0.9325\\
&\text{H3 3.} && e^{-0.9325}=0.39357\\
&\text{H4 1.} && (1,0,1)-(0,-0.15,0.6)=(1,0.1,0.4)\\
&\text{H4 2.} && (\|(1,0.1,0.4)\|)^2=1.1825\\
&\text{H4 3.} && e^{-1.1825}=0.30651
\end{aligned}
\tag{5.2}
$$

Након што су резултати послати до излазног слоја, креће прорачун у модулу RB 1. У том модулу се рачуна скаларни производ резултата добијених од скривеног слоја и коефицијената у излазном слоју. У прорачуну 5.3 види се да се као коначан резултат добија вредност 0.20900. На основу знака резултата ANN класификатор из примера додељује првој инстанци као резултат класификације класу 1.

$$
\text{O}\quad (0.60653,0.36059,0.39357,0.30651)\cdot(0.5,-0.4,0.75,-0.8)=0.20900 \tag{5.3}
$$

За другу инстанцу на основу прорачуна 5.4 и 5.5 као коначан резултат рачунања се добија вредност -0.09583. Пошто је резултат прорачуна вредност мања од 0, инстанци 1 се додељује класа 2 као коначна вредност класификације.

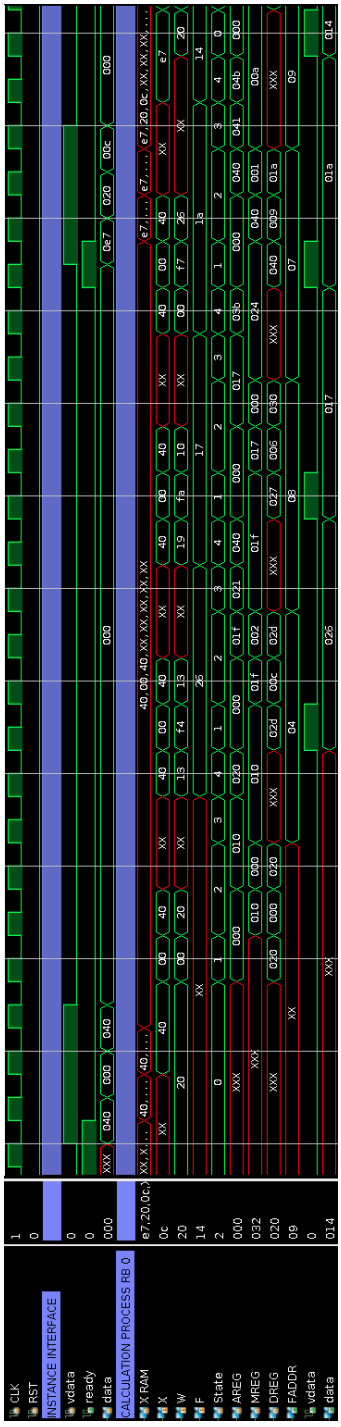
$$
\begin{aligned}
&\text{H1 1.} && (-0.4, 0.5, 0.2) - (0.5, 0, 0.5) = (-0.9, 0.5, -0.3)\\
&\text{H1 2.} && (\|(-0.9, 0.5, -0.3)\|)^2 = 1.15\\
&\text{H1 3.} && e^{-1.15} = 0.31664\\
&\text{H2 1.} && (-0.4, 0.5, 0.2) - (0.3, -0.2, 0.3) = (-0.7, 0.7, -0.1)\\
&\text{H2 2.} && (\|(-0.7, 0.7, -0.1)\|)^2 = 0.99\\
&\text{H2 3.} && e^{-0.99} = 0.37158\\
&\text{H3 1.} && (-0.4, 0.5, 0.2) - (0.4, -0.1, 0.25) = (-0.8, 0.6, -0.05)\\
&\text{H3 2.} && (\|(-0.8, 0.6, -0.05)\|)^2 = 1.0025\\
&\text{H3 3.} && e^{-1.0025} = 0.36696\\
&\text{H4 1.} && (-0.4, 0.5, 0.2) - (0, -0.15, 0.6) = (-0.4, 0.65, -0.4)\\
&\text{H4 2.} && (\|(-0.4, 0.65, -0.4)\|)^2 = 0.7425\\
&\text{H4 3.} && e^{-0.7425} = 0.47592
\end{aligned}
\tag{5.4}
$$

$$
\text{O} \quad (0.31664, 0.37158, 0.36696, 0.47592) \cdot (0.5, -0.4, 0.75, -0.8) = -0.09583 \tag{5.5}
$$

Атрибути улазне инстанце се прихватају преко улазних портова vdata и data док се уз помоћ излазног порта ready може одложити прихватање улазне инстанце потребан број циклуса, док архитектура није спремна за прорачун (Слика 5.12). Трансфер који је започео постављањем порта vdata на '1' се не сме прекидати.

Улазна инстанца се пребацује у архитектуру током три циклуса, пошто толико атрибута има улазна инстанца. Трајање прорачуна унутар једног RB модула, када архитектура ради као ANN класификатор, може доста да се разликује од модула до модула. Време прорачуна за једну инстанцу унутар модула пропорционално је броју неурона унутар слоја који је смештен у модулу. Углавном, када архитектура ради као ANN, мора да се направи известан размак између трансфера и класификовања две инстанце. Тај размак се уочава по томе што архитектура држи излазни порт ready на '0' све време за које не може да прими наредну инстанцу због класификовања тренутне инстанце (Слика 5.12). Чим се прими последњи атрибут улазне инстанце, модул RB започиње прорачун. Почетак прорачуна је циклус у коме архитектура прелази из стања 0 ("IDLE") у стање 1 ("STEP_1"). Када архитектура започне прорачун, инстанца која је тренутно смештена у меморији X RAM почиње да се обрађује. Стања аутомата модула RB су кодована бинарно.

- IDLE = 0



Слика 5.12: Класификовање прве инстанце унутар модула RB 0 за ANN

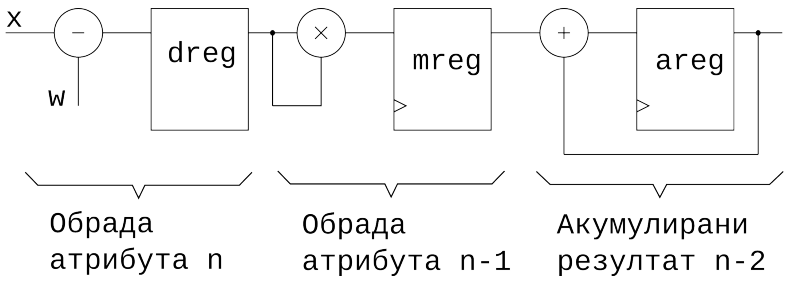
- STEP_1 = 1
- STEP_2 = 2
- STEP_3 = 3
- STEP_4 = 4
- STEP_5 = 5

Аутомат у модулу CALC, док је у стању "IDLE", чека да пристигне последњи атрибут улазне инстанце или последњи резултат претходног слоја неуронске мреже пре него што започне прорачун. Ово се разликује од понашања аутомата када он ради као DT или SVM где је аутомат почео прорачун чим је пристигао први атрибут. Ова разлика постоји стога што када модул CALC ради као неуронска мрежа, он не разликује да ли прима улазне атрибуте или резултате прорачуна претходног слоја мреже. Резултати претходног слоја мреже не долазе један за другим, већ су размакнути известан број циклуса. Стога је прорачун могуће започети тек када пристигне последњи атрибут инстанце, односно резултат прорачуна претходних слојева. Када је пристигла последња вредност неопходна за прорачун аутомат иницијализује први ниво у ланцу прорачуна, који је у овом случају регистар иза одузимача, DREG (Слика 5.13). Уједно аутомат се пребацује у стање "STEP_1".

Аутомат модула CALC у стању "STEP_1" остаје само један циклус. У овом стању прорачун разлике централног вектора и улазне инстанце је почео и резултат за први атрибут улазне инстанце и први скалар централног вектора се налази у регистру DREG. Може се видети да је резултат у овом регистру 0.5 (0x20) (Слика 5.13). У овом стању други ниво ланца прорачуна се активира. Након једног циклуса аутомат модула CALC се пребацује у стање "STEP_2".

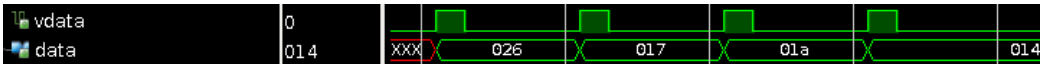
Аутомат у стању "STEP_2" рачуна радијалну кернел функцију и све операције осим рачунања експонента се извршавају унутар модула CALC. Сва три нивоа ланца прорачуна су активна. У регистру DREG се налазе међурезултати разлике два вектора за n-ти атрибут инстанце и скалар вектора. У регистру MREG је смештен квадрат разлике n-1 атрибута инстанце и скалара централног вектора. Ови квадрати се на крају сумирају у регистру AREG, трећем нивоу ланца прорачуна (Слика 5.13). Аутомат остаје у стању "STEP_2" онолико циклуса колико има атрибута улазна инстанца умањеном за 1. За овај пример то значи да ће аутомат остати 2 циклуса у овом стању (Слика 5.12). У случају када се не би рачунао кернел који је радијалног типа, већ

када би се радило са неуронским мрежама типа MLP, ланац прорачуна би имао 2 нивоа и био би као у примеру рада архитектуре као SVM класификатора. Аутомат унутар модула CALC ће променити тренутно стање на наредно, "STEP_3" у тренутку када се у регистру DREG појави разлика последњег атрибута инстанце и последњег скалара вектора разлике. У примеру се може видети да је у циклусу, када је аутомат још увек у стању "STEP_2", у регистру DREG је вредност 0.5 (0x20) што тачно одговара последњој разлици. У регистру DREG се током три циклуса може видети вектор разлике, у овом случају (0.5, 0, 0.5) - (0x20, 0x00, 0x20).



Слика 5.13: Ланац за рачунање у модулу CALC у стању "STEP_2" када класификатор ради као ANN.

У стању "STEP_3" аутомата завршава се рачунање вредности која је потребна да би се проследила као улаз за нелинеарну функцију. У овом стању у регистру MREG се налази последњи квадрат разлике. Адресном улазу меморије F RAM се прослеђује срачуната сума са сабирача, али само пет значајних бита. У примеру се може видети да је меморији FRAM прослеђена вредност 0.5 (0x04) (Слика 5.12). Вредност која се прослеђује као адреса меморији FRAM се заправо интерпретира као реалан број представљен у репрезентацији фиксном тачком са 2 бита у целом делу и 3 бита у разломљеном. Вредност која је прочитана из меморије ће се наредни циклус видети на улазу F аутомата. Аутомат остаје у стању "STEP_3" један циклус и потом се пребацује у наредно стање, "STEP_4".



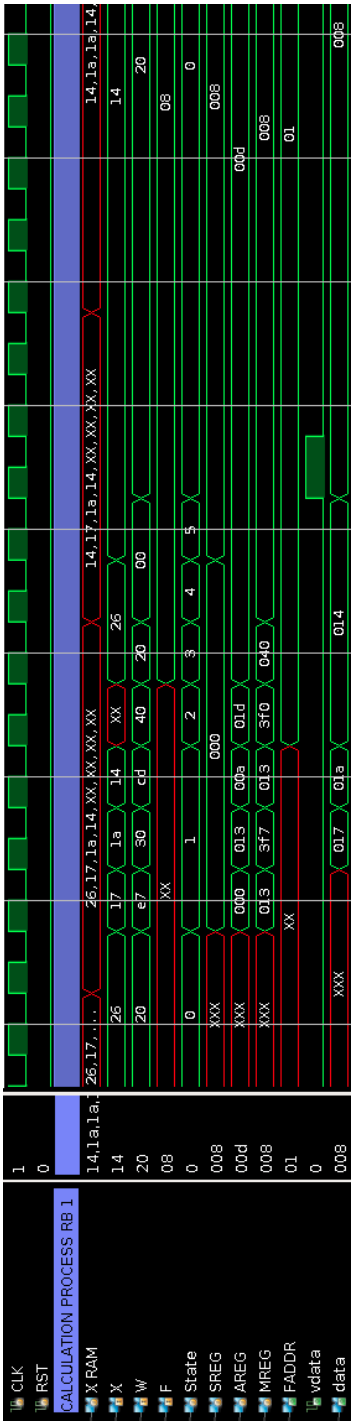
Слика 5.14: Излазне вредности за прву инстанцу унутар модула 0, када класификатор ради као ANN

Када је аутомат у стању "STEP_4" добијена вредност из меморије F RAM се прослеђује на излаз data уз потврђивање да је вредност излаза важећа. Може се видети да је на излаз прослеђена вредност 0.59375 (0x26) што је врло приближно тачно срачунатој вредности 0.60653. Избацивање важеће вредности на излаз data представља крај прорачуна за први неурон. Пошто у слоју неуронске мреже која се налази у модулу RB 0 има још неурона, аутомат наставља прорачун од стања "STEP_1". Док не заврши прорачун за све неуроне у слоју, аутомат ће пролазити циклично кроз стања "STEP_1", "STEP_2", "STEP_3" и "STEP_4". Када дође до последњег неурона у слоју и налази се у стању "STEP_4", аутомат прослеђује срачунату вредност на излаз али у том случају прелази у стање "IDLE". Из овог разматрања је јасно да је време прорачуна модула RB, када ради као ANN калсификатор, сразмерно броју неурона у слоју, као и броју улазних вредности које неурон прима.

Још једна значајна разлика у раду аутомата CALC као ANN класификатора, у односу на случајеве када ради као DT и SVM класификатор, је да се улазни атрибути не прослеђују до наредног модула RB, пошто они нису потребни за прорачун. Уместо тога се прослеђују само резултати прорачуна неурона унутар слоја (Слика 5.14). Ово значи да RB модули када раде као неуронске мреже на улазе могу добити или атрибуте или резултате прорачуна претходних слојева и они се употребљавају на исти начин, без разлике. Јасно је и зашто модул CALC започиње прорачун тек када добије последњу улазну вредност. Пошто улазне вредности долазе раздвојене, прорачун и није могуће почети док све вредности не стигну. Код рада модула RB као DT или SVM класификатора, сви улазни атрибути су долазили до свих RB модула један за другим па је у том случају могуће започети прорачун чим стигне прва вредност. Вредности срачунате у неуронима су: 0.59375 (0x26), 0.35938 (0x17), 0.40625 (0x1a) и 0.3125 (0x14). Ове вредности се прослеђују до наредног модула и врло су приближне тачним вредностима: 0.60653, 0.36059, 0.39357 и 0.30651.

Модул RB 1 ради другачије од модула RB 0. Пошто цео класификатор ради као неуронска мрежа са радијалним кернелом овај модул треба да направи тежинску суму резултата добијених од претходног модула. Треба нагласити да би у случају да класификатор ради као неуронска мрежа типа MLP и овај модул радио исто као и претходни. У зависности колико слојева има MLP класификатор, толико модула RB би било активно и сваки од њих би имплементирао један слој MLP неуронске мреже.

Као што је већ речено, модул RB 1 ради као да се рачуна прорачун за SVM класификатор са линеарним кернелом, при чему је овај модул

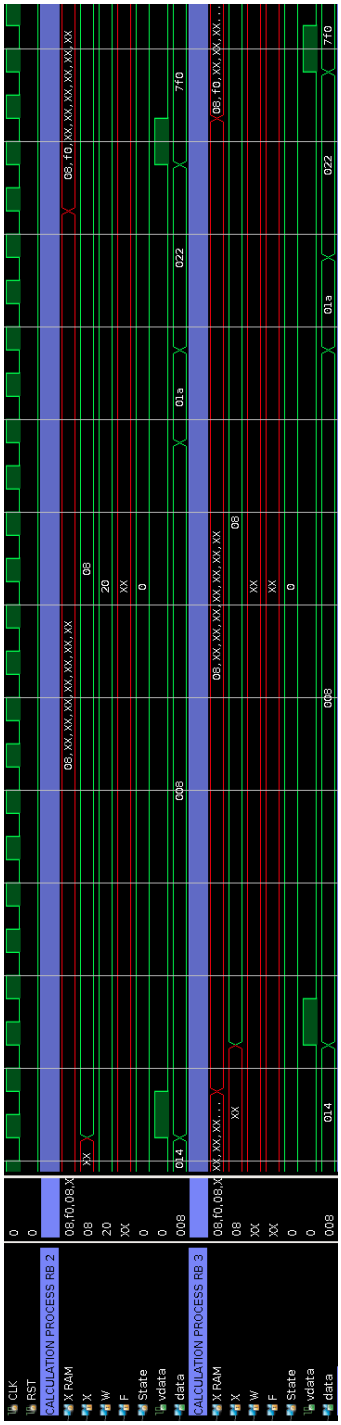


Слика 5.15: Класификовање прве инстанце унутар модула RB 1 за ANN

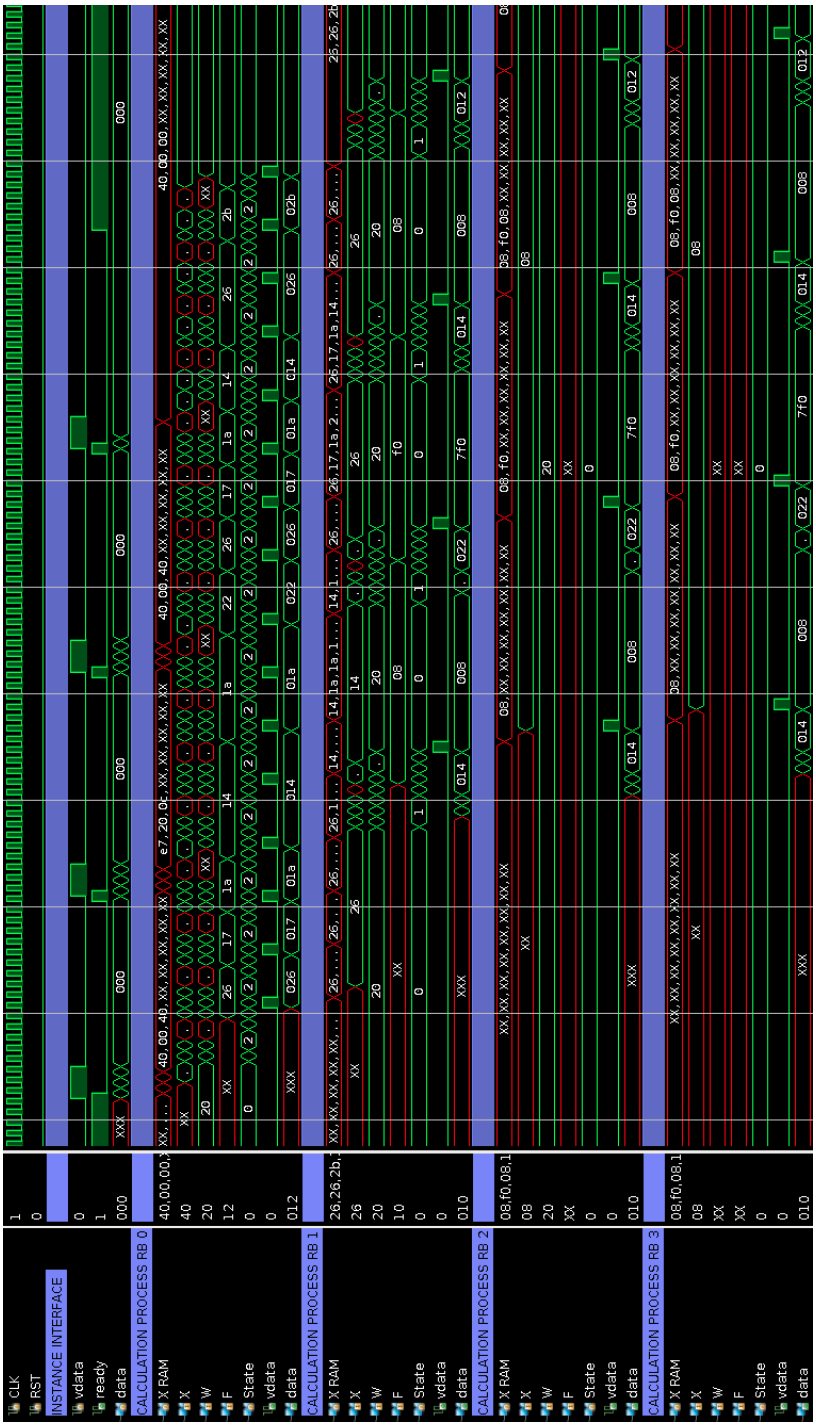
и први и последњи. Модул се конфигурише као да је први, зато што се у том случају сума поставља на вредност 0. Конфигурише се и као последњи зато што се тада на излаз прослеђује само резултат прорачуна, а не и остале улазне вредности. Прорачун је идентичан већ описаном прорачуну код примера рада архитектуре као SVM класификатора (Слика 5.15). Вектор коефицијената за множење резултата прорачуна скривеног слоја је идентичан као један вектор подршке. Ово је уједно и једини вектор подршке. Пошто у овом случају не треба да се додаје додатна вредност, она је постављена на 0. Када архитектура заврши прорачун на излазном порту ће се појавити вредност 0.125 (0x08), док је тачна вредност 0.20900. Ова вредност се прослеђује до наредних модула. Као што се може приметити модул RB 1 има знатно мање тога да рачуна, па добар део времена стоји у стању "IDLE".

У модулима RB 2 и RB 3 неће бити никаквог прорачуна већ ће се само добијена вредност од модула RB 1 проследити на излазне портове система (Слика 5.16). Пошто је добијена вредност већа од 0, улазној инстанци ће као резултат прорачуна бити додељена класа 1, што је исти резултат који је добијен прорачунима 5.2 и 5.3. Може се видети да се као резултат прорачуна за другу инстанцу добија вредност -0.25 (0xF0) док је тачна вредност на основу прорачуна 5.4 и 5.5 -0.09583. У оба случаја ће улазној инстанци као резултат класификације бити додељена класа 2. Мали број бита узет приликом параметризовања архитектуре, за представљање реалних бројева у репрезентацији са фиксном тачком, ни у овом случају није променио резултат класификовања.

Као и у случајевима рада архитектуре као DT и SVM класификатора, прорачун архитектуре је проточан. То као резултат даје велико кашњење за добијање првог резултата прорачуна али се сваки наредни добија знатно брже, уколико архитектура ради под највећим оптерећењем, па се стога добија велико убрзање прорачуна када се улазне инстанце класификују једна за другом. Величина самог убрзања биће презентована у поглављу посвећеном експерименталним резултатима. Пошто је организација архитектуре проточна то значи да сви модули раде на различитим улазним векторима током прорачуна (Слика 5.17). Када архитектура ради као ANN, слично као и код рада као SVM, може се предвидети тачан тренутак када ће стићи резултат класификације архитектуре. Да би RMLC архитектура могла да имплементира одговарајући ANN класификатор, потребно је да има бар онолико RB модула колики је број слојева неуронске мреже. У обрађеном примеру тај број је већи па су два RB модула била неактивна. RB ANN мреже имају само два слоја која је потребно имплементирати унутар RB модула, зато је у случају ових мрежа довољно само два модула RB у



Слика 5.16: Класификовање прве инстанце унутар модула RB 2 и 3 за ANN



Слика 5.17: Поглед на прорачун свих модула на 4 улазне инстанце ANN класификатора

једнодимензионалном низу. MLP мреже могу имати већи број скривених слојева па је за њихову имплементацију потребан већи број RB модула.

Глава 6

Универзална реконфигурабилна архитектура за убрзавање ансамбала класификатора

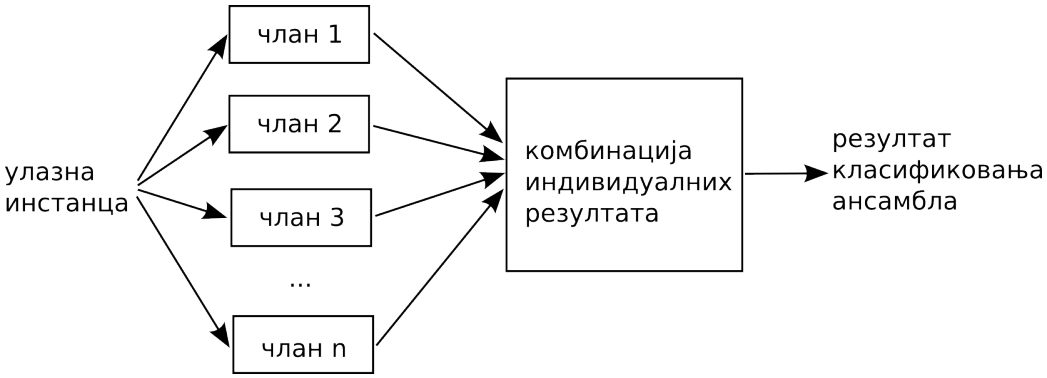
У претходном поглављу приказана је универзална реконфигурабилна дигитална архитектура која може да реализује SVM, DT и ANN класификаторе. Ту архитектуру смо назвали RMLC. У овом поглављу биће представљено проширење RMLC архитектуре које ће јој омогућити да ради и као хомогени или хетерогени анасамбл класификатора. Ову архитектуру ћемо звати REC. Као што је описано, RMLC архитектура је организована као једнодимензиони низ блокова са истом функционалношћу, способних да се конфигуришу у произвољан DT, SVM или ANN класификатор. Уколико би се уместо једног једнодимензионог низа користило више њих, добила би се матрична структура која би могла да реализује произвољан, хомоген или хетероген ансамбл састављен од DT, SVM или ANN класификатора.

6.1 Ансамбли класификатора

Ансамбли класификатора су у жижи пажње истраживача ваштачке интелигенције и машинског учења у последих неколико деценија. Ансамбли класификатора су се показали као ефикасни алати у решавању многих различитих проблема машинског учења, као што су одабир особина, процена сигурности, инкрементално учење, исправка грешака...

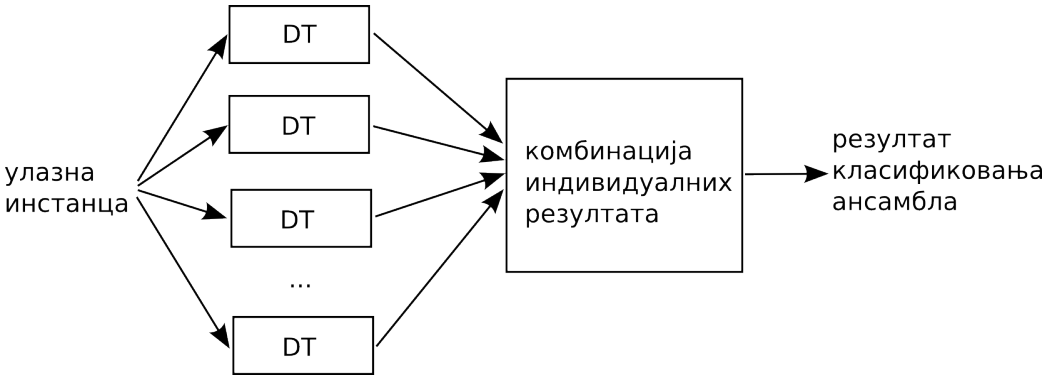
Главна идеја ансамбала је да се одабере одговарајуће правило за комбиновање предикција више резличитих индивидуалних класификатора. Класификатори су различити уколико праве

различите грешке на истим улазним инстанцама. Прецизност сваког од класификатора треба да је већа од 50%. Под овим претпоставкама, ансамбл класификатора може да досегне произвољну тачност доношењем колективне одлуке (Слика 6.1).



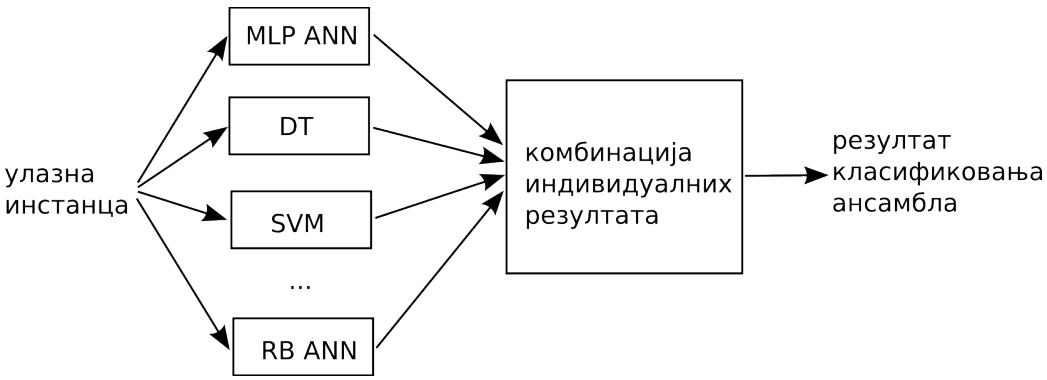
Слика 6.1: Ансамбл класификатора

У општем случају, ансамбли се могу састојати од различитих типова класификатора (Слика 6.3) и тада се називају хетерогени ансамбли. Уколико се ансамбл састоји само од једне врсте класификатора тада се називају хомогеним ансамблима (Слика 6.2).



Слика 6.2: Хомогени ансамбл класификатора

У литератури постоји много различитих алгоритама за конструкцију ансамбала класификатора а неки од њих су Bagging, Boosting, AdaBoost [41], Stacked generalization [42], Mixtures of experts [43] и разна комбинациона правила као што су већинско гласање (анонимно гласање, најпростије већинско гласање, тежинско већинско гласање), Behavioural Knowledge Space [44] и Borda Count. У овој дисертацији ћемо



Слика 6.3: Хетерогени ансамбл класификатора

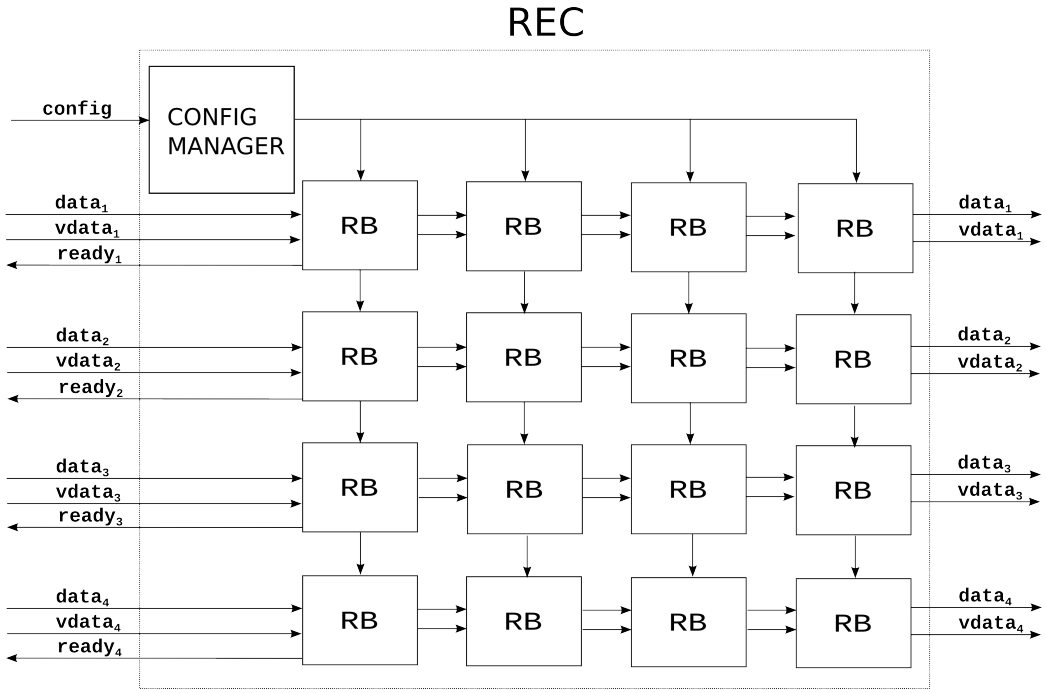
се бавити само реализацијом самих машинских модела, не и њиховом конструкцијом.

6.2 Детаљи REC архитектуре

Основна структура архитектуре REC је организована као дводимензиони низ (матрица) истоветних реконфигурабилних RB модула (Слика 6.4). Број врста у овој матрици мора бити већи или једнак броју чланова ансамбла који архитектура треба да имплементира. Број потребних колона RB матрице треба да је већи или једнак дубини најдубљег стабла члана ансамбла или највећем броју слојева ANN мреже која је члан ансамбла. Број колона и врста REC архитектуре се одређује параметрима архитектуре пре имплементације класификатора и не може се мењати током рада система.

Свака врста RB матрице реализује један класификатор који је члан ансамбла. Сваки од класификатора је одговоран за додељивање једне класе тренутној инстанци. Сваки класификатор је имплементиран на начин описан у претходним поглављима (Слика 6.5). Подаци о атрибутима инстанци теку само у хоризонталном смеру. У сваком низу REC архитектуре, прорачун је проточан. Сваки RB модул у низу прима податке од претходног модула. Сваки RB модул обрађује податке у зависности од тога да ли је низ конфигурисан као DT, SVM или ANN. Због проточне обраде, више инстанци се обрађује истовремено на различитим модулима RB.

Осим RB модула унутар REC архитектуре налази се и додатни модул који служи за конфигурисање појединачних RB модула. Овај модул ћемо звати CM (енглески: Configuration Manager). Модул CM прима податке



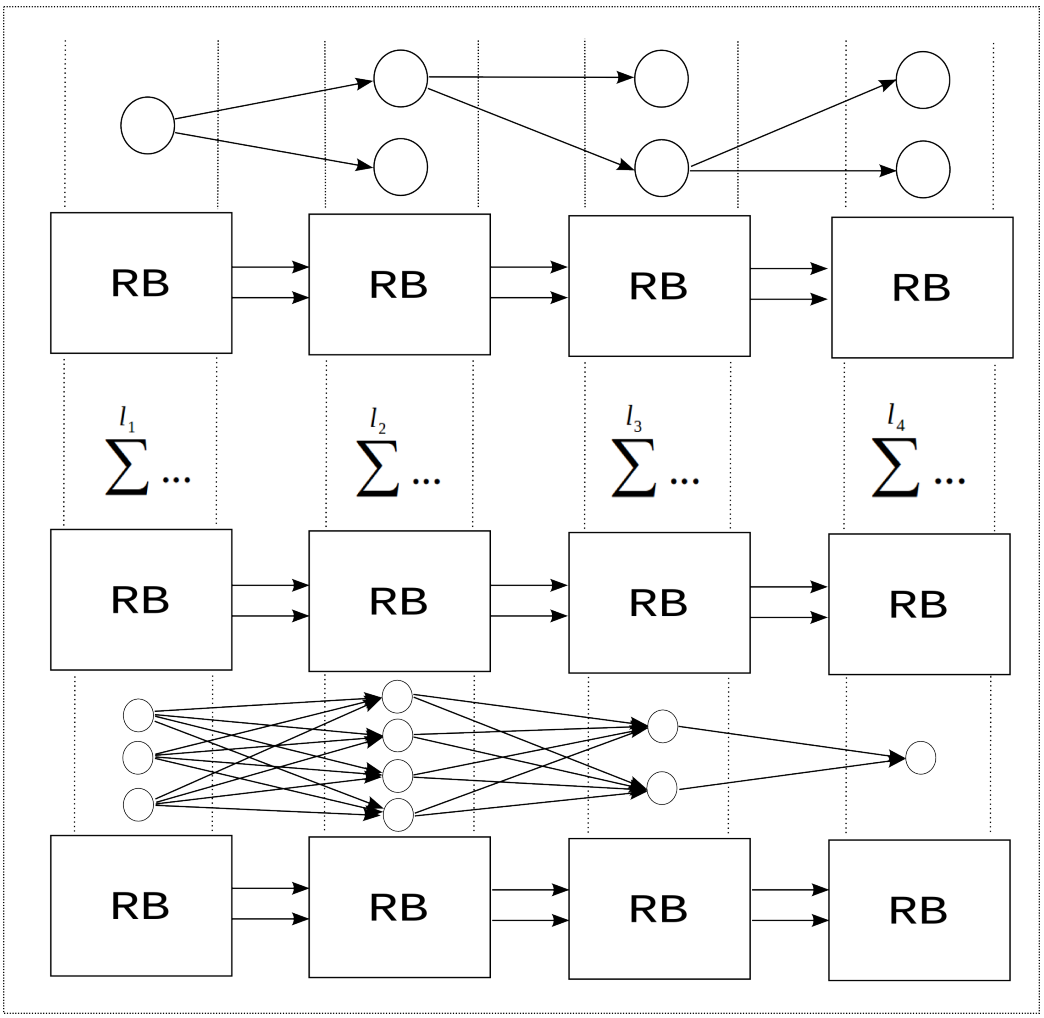
Слика 6.4: Основна структура REC архитектуре

од конфиурационих сигнала и бира којем од RB модула треба да их проследи. Конфигурациони сигнали су они исти, већ набројани: baddr, cfg, we_cfg, xlen, we_xlen, wlen, we_wlen, we_wdata, wdata, wr_waddr, wr_xaddr, we_fdata, fdata и wr_fdata. За разлику од тока података атрибута инстанце, ток конфигурационих података је вертикалан унутар REC архитектуре. Модул CM је повезан директно са свим RB модулима.

6.3 Изградња комплетног ансамбла

У овом поглављу ће бити показано како се може дизајнирати комплетан ансамбл класификатора који је заснован на REC архитектури. Генерално, хардверски системи за реализацију ансамбала класификатора се састоје од два модула:

- Модул EMI (енглески: Ensemble Members Implementation) за реализацију појединачних класификатора ансамбла који су задужени да доделе индивидуалну класу улазној инстанци;
- Модул CRC (енглески: Combination Rule Calculation) који преузима резултате класификовања појединачних модула и комбинује



Слика 6.5: Мапирање на REC архитектуру

их коришћењем неког од правила комбиновања и представља колективни резултат класификовања тренутне инстанце.

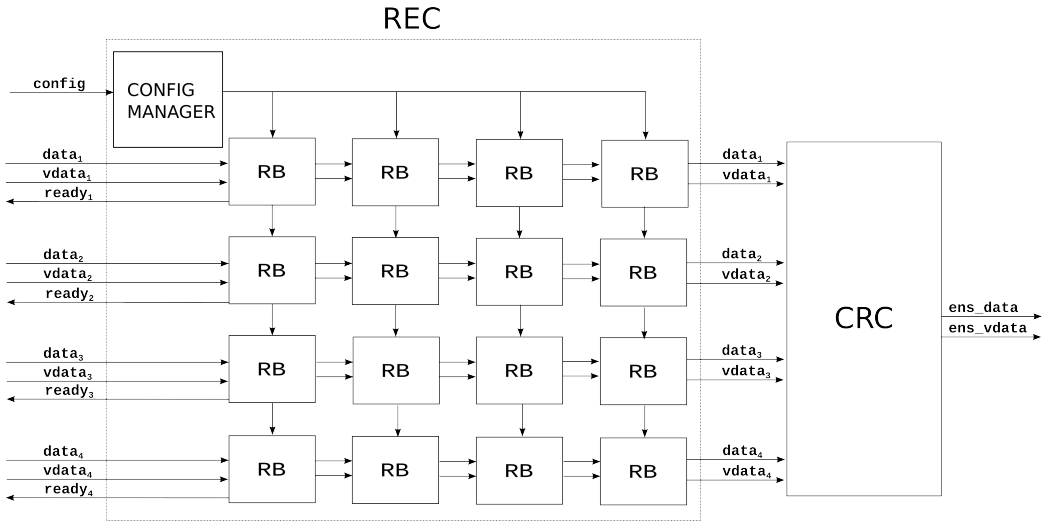
Архитектура REC се може употребити за имплементацију модула EMI свеукупног ансамбла класификатора. Пошто су RB модули универзални, у смислу да могу да имлпементирају DT, SVM или ANN, архитектура REC може бити употребљена за имплементацију хомогених и нехомогених чланова ансамбла. Штавише, пошто је REC архитектура конфигурабилна, могуће је модификовати архитектуру по потреби у току рада. Архитектура се са лакоћом може пребацити са хомогеног ансамбла на нехомогени и обрнуто, или се неки од чланова ансамбла може променити у тренутку када нови подаци за тренирање постану доступни.

Архитектура REC не може да се користи за реализацију CRC модула. За ову сврху нека друга архитектура треба да се размотри. Архитектуре за ефикасну имплементацију CRC модула са различитим правилима за комбиновање већ постоје и могу се видети у [45]. Неке од тих паралелних архитектура се лако могу повезати са REC архитектуром да би се добиле целокупне хардверске имплементације ансамбала класификатора.

Архитектура REC би у целокупном ансамбл систему рачунала у паралели резултате појединачних чланова ансамбла за улазну инстанцу (Слика 6.6). Када би резултати класификовања постали доступни унутар REC архитектуре, сви резултати би били послати, истовремено, на излазне портове. Сви резултати би ушли у неки од предложених паралелних CRC модула [45] који би их све искомбиновао неким од правила да би се на крају добио коначан резултат класификовања. Тај резултат се прослеђује за даљу обраду кроз јединствен излазни интерфејс за резултат класификовања.

Архитектуре модула CRC предложених у [45] захтевају да се сви појединачни резултати класификовања проследе истовремено. REC архитектура подржава овакав приступ, зато што се сваки RB модул може конфигурисати да буде линија за кашњење за програмабилном вредношћу кашњења. Због тога, целокупан ансамбл класификатор се састоји само од REC архитектуре која је директно повезана на модул CRC (Слика 6.6).

У случају када REC архитектура ради као хомогени ансамбл састављен од DT свако стабло је мапирано на једну врсту REC матрице (Слика 6.7). Прорачун унутар стабла је исти као у случају RMLC архитектуре. Број модула RB по врстама и колонама REC матрице да би се имплементирао одговарајући хомогени DT ансамбл може да се одреди према следећим правилима. Број врста RC архитектуре треба

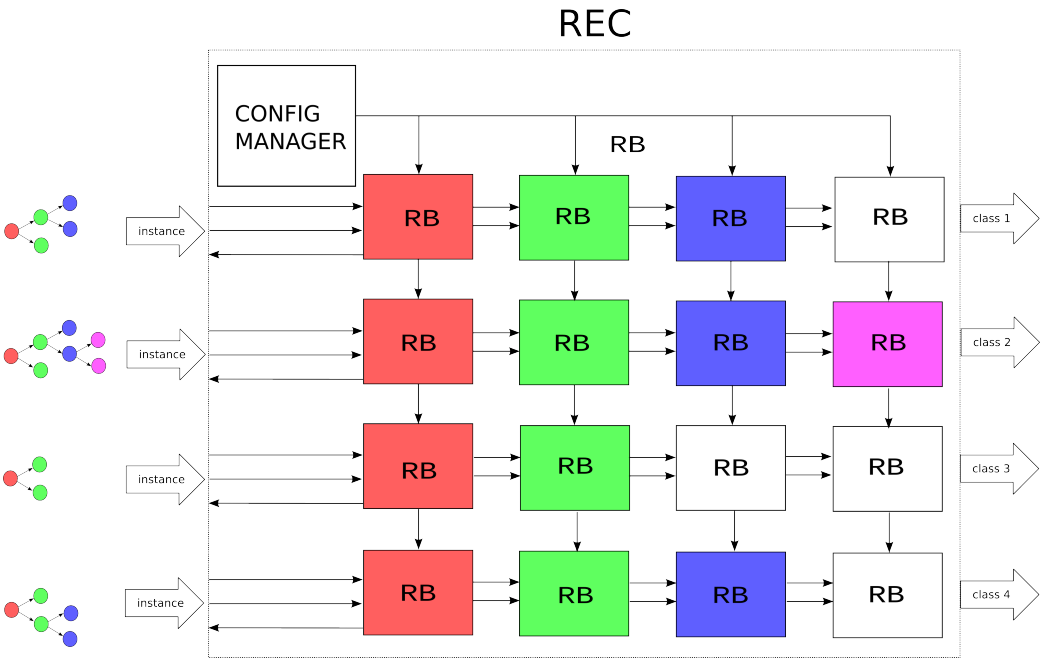


Слика 6.6: Структура целокупног хомогеног и нехомогеног ансамбла класификатора базираног на REC архитектури

да је једнак броју чланова ансамбла, док број колона треба да је једнак дубини најдубљег стабла у ансамблу. Пошто дубине стабала различитих елемената ансамбла могу да варирају, а и сама брзина једног члана стабла може да буде различита од случаја до случаја потребно је на неки начин синхронизовати појављивање резултата класификовања појединих чланова ансамбла.

За хомогени ансамбл приказан у примеру (Слика 6.7) дубина првог и четвртог стабла је 3, дубина другог стабла је 4, док је дубина трећег стабла 2. То значи да ће класификатори 1 и 4, у најгорем случају, завршити класификовање за троструко време потребно једном RB модулу да заврши прорачун, класификатору два четвороструко а класификатору 3 двоструко. Треба приметити да постоје и модули који су неискоришћени (Модули RB означени белом бојом на слици 6.7) пошто су одговарајућа стабла мање дубине од 4. Ови неискоришћени модули могу бити конфигурисани да пропуштају резултат класификовања или да га одлажу потребно време зарад синхронизације. На улаз сваког појединог класификатора се прослеђује иста инстанца за класификовање.

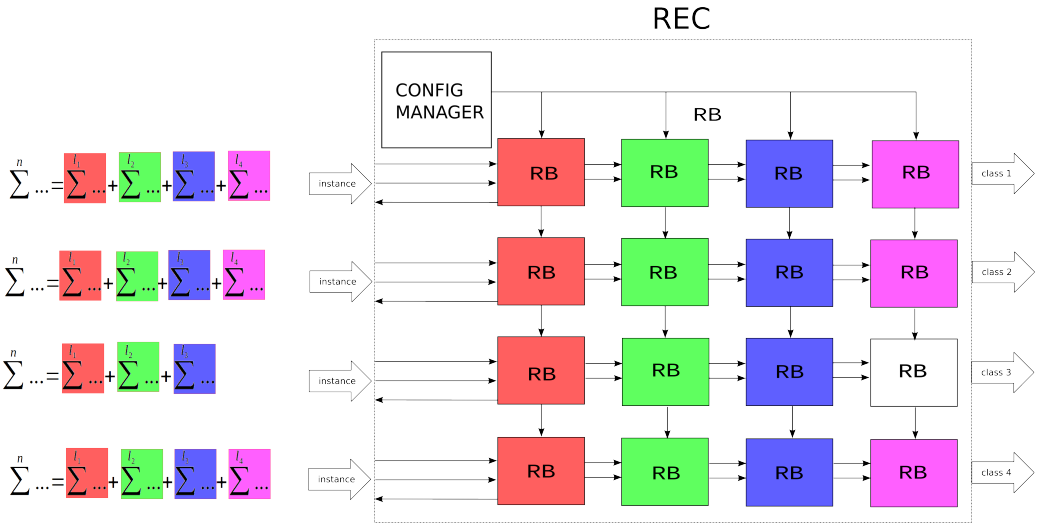
Када се имплементира хомогени SVM ансамбл, сваки SVM члан ансамбла се мапира на једну врсту REC архитектуре (Слика 6.8). Сваки SVM класификатор се имплементира исто као и у случају RMLC архитектуре. Величина REC архитектуре потребна за реализацију



Слика 6.7: Имплементација појединачних чланова хомогеног ансамбла, састављеног од DT-ова

хомогеног SVM ансамбла се одређује помоћу наредне процедуре. Број потребних врста једнак је броју чланова ансамбла, док је број колона једнак највећој подели укупне суме потребне за срачунавање класификације једног SVM класификатора. Треба нагласити да што је већи број колона то се укупна сума може поделити на више мањих, тако да је у случају хомогеног SVM ансамбла, са аспекта бржег прорачуна већи број колона увек пожељан. У сваком случају, пошто број вектора подршке по класификатору може бити различит у појединим члановима ансамбла и у случају хомогених SVM ансамбала може доћи до проблема синхронизације који се може решити на већ описани начин.

На приказаном примеру хомогеног SVM ансамбла (Слика 6.8) види се ансамбл од 4 члана са различитим парцијалним сумама. Само трећи члан ансамбла има блок који је неискоришћен. Као што је већ напоменуто и код RMLC архитектуре ово је случај који се ретко дешава, пошто се сваки додатни модул RB може искористити за додатно убрзање у прорачуну. Као што се може видети (Слика 6.8) на улаз сваког појединог класификатора се прослеђује иста инстанца за класификовање.

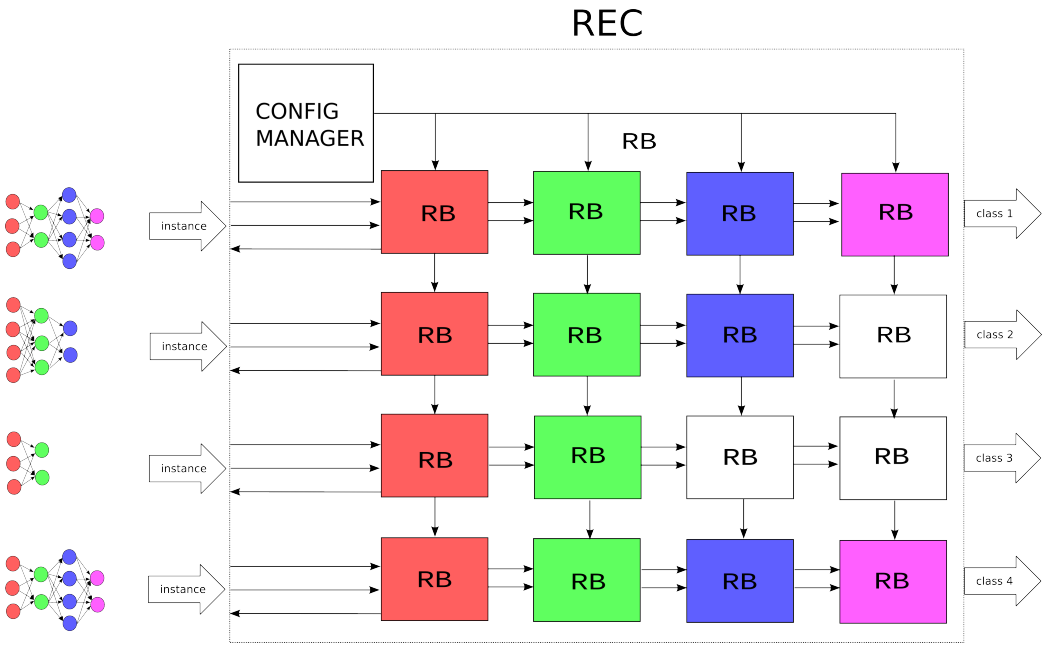


Слика 6.8: Имплементација појединачних чланова хомогеног ансамбла, састављеног од SVM-ова

И на крају, у случају када REC архитектура ради као хомогени ANN ансамбл, сваки члан ансамбла се мапира на једну врсту REC матрице (Слика 6.9). Свака неуронска мрежа се имплементира на исти начин као код RMLC архитектуре. Одређивање величине REC архитектуре, да би се омогућила имплементација хомогеног ANN ансамбла, може се урадити на следећи начин: број потребних врста једнак је броју чланова ансамбла, док је број колона једнак дубини најдубље мреже од свих појединих чланова. Чланови ансамбла, слично као и у случају стабала, могу имати различите дубине. У овом случају и сами слојеви могу имати различит број неурона па се стога и у овом случају може појавити проблем синхронизације који се решава на већ описани начин.

За хомогени ансамбл приказан као пример (Слика 6.9) постоје четири члана од којих први и четврти имају дубину 4, други има дубину 3, док трећи има дубину 2. За прву и четврту мрежу ће требати знатно веће време за класификацију него за, на пример, трећу мрежу. И у овом случају постоје модули који су остали неискоришћени и који се могу употребити за већ описани проблем синхронизације, док се иста инстанца за класификовање прослеђује сваком појединoм класификатору.

Због своје универзалности и реконфигурабилности, REC архитектура се може користити за имплементацију хетерогених ансамбала који се састоје од DT-ова, SVM-ова и ANN-ова. У случају хетерогених ансамбала класификатора, појединi класификатори моги припадати



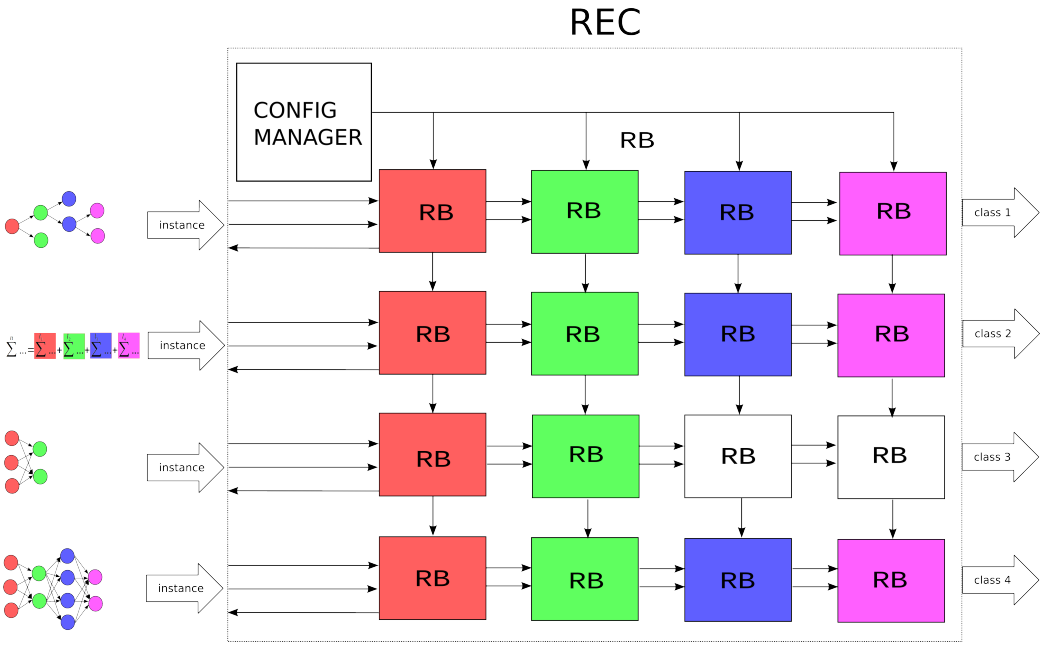
Слика 6.9: Имплементација појединачних чланова хомогеног ансамбла, састављеног од ANN-ова

различитим врстама модела машинског учења. Пошто се REC архитектура може индивидуално, врсту по врсту, конфигурисати у различит тип класификатора, могуће је реализовати било који хетерогени ансамбл који се састоји од DT, SVM или ANN класификатора.

Као пример, представљен је хетерогени ансамбл који се састоји од четири члана (Слика 6.10). Један члан је стабло одлуке и мапиран је на прву врсту REC архитектуре. Један члан је SVM класификатор који је имплементиран у другој врсти архитектуре. Преостала два члана су неуронске мреже које су реализоване у последње две врсте архитектуре. Одговарајућим конфигурисањем сваког од RB модула, REC архитектура може да класификује произвољну улазну инстанцу и за хетерогени ансамбл.

И у овом случају, као и код хомогених ансамбала, постоји проблем синхронизације. Штавише, у случају хетерогених ансамбала овај проблем је још израженији, пошто време класификовања различитих врста класификатора може одступати много више, него када су класификатори истоветни. Пошто се модули RB могу конфигурисати да унесу додатно кашњење, овај проблем се решава исто као и код

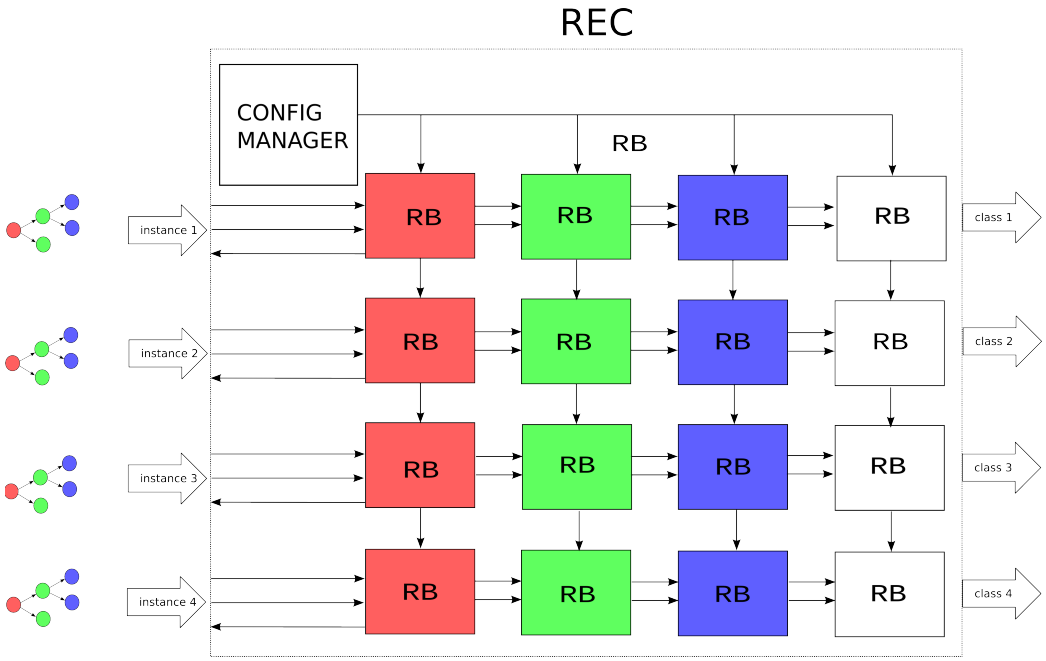
хомогених ансамбала, док се иста инстанца за класификовање прослеђује сваком поједином класификатору.



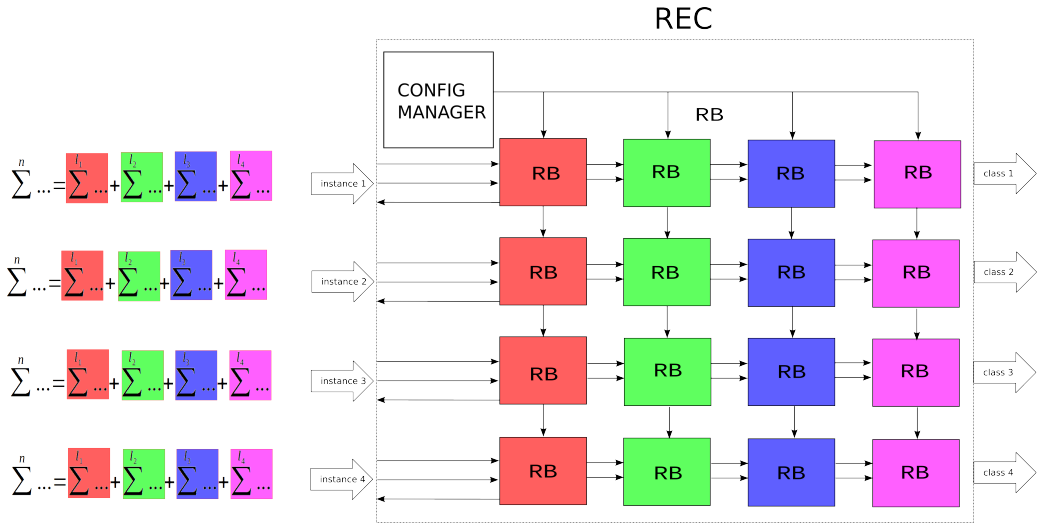
Слика 6.10: Имплементација појединачних чланова хетерогеног ансамбла, састављеног од DT-ова, SVM-ова и ANN-ова

На крају, треба напоменути да се REC архитектура може употребити и за вишеструко класификовање истоветног класификатора. Пошто се свака врста REC архитектуре може програмирати на произвољан начин, могуће је имплементирати исти DT, SVM или ANN класификатор у свакој врсти (Слике 6.11, 6.12 и 6.13). Надаље, пошто свака врста има свој посебан улаз за прослеђивање атрибута улазне инстанце и посебан излаз за давање резултата класификовања, архитектура може класификовати различите инстанце за исти проблем класификовања у паралели. Све врсте се при томе конфигуришу као исти класификатор. Овакав приступ класификовању може драстично да повећа проточност архитектуре. Ако се REC архитектура користи са M врста, проточност ће се повећати M пута. Ово је уједно и главни разлог зашто је у REC архитектури остављен посебан улаз за сваку врсту. Да би се архитектура користила на овакав начин за сваку врсту је потребан посебан улаз. У случају ансамбала био би потребан само један улаз за атрибуте улазне инстанце. Треба обратити пажњу да у датим примерима (Слике 6.11, 6.12 и 6.13) вишеструког повећања протока

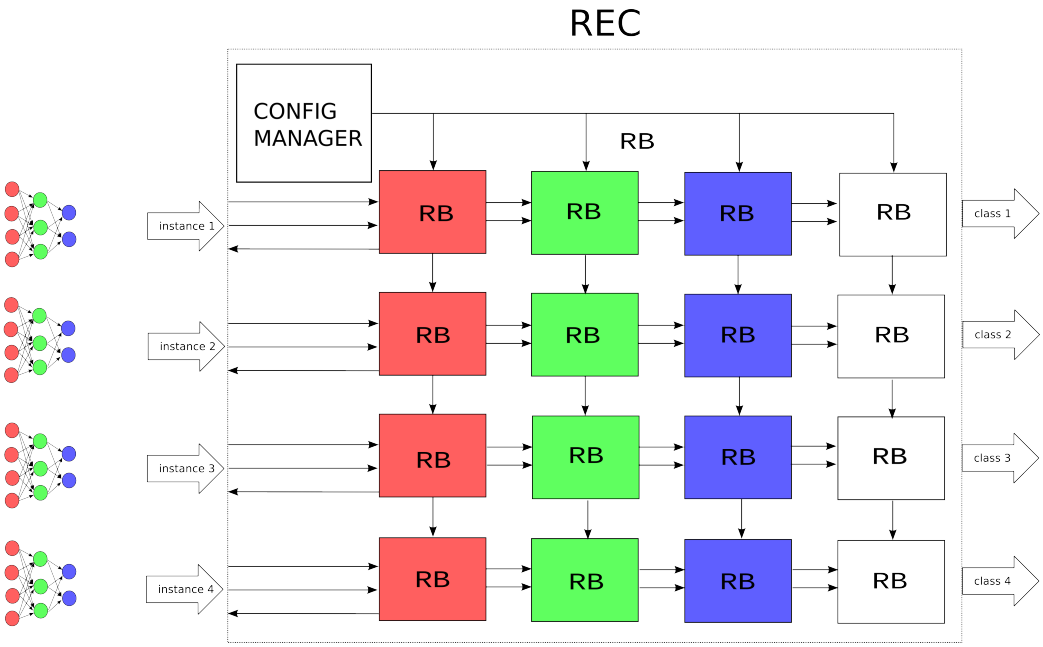
индивидуалног класификатора, на улазе за атрибуте инстанци доводе различите инстанце. Када архитектура ради на овај начин модул CRC је сувишан и проблем синхронизације не постоји.



Слика 6.11: Вишеструка паралелна имплементација једног DT класификатора



Слика 6.12: Вишеструка паралелна имплементација једног SVM класификатора



Слика 6.13: Вишеструка паралелна имплементација једног ANN класификатора

Глава 7

Експериментални резултати

У овом поглављу ће бити представљени резултати експеримената чији је циљ да се упореде перформансе RMLC и REC архитектуре са софтверским пакетима, WEKA и R project.

Софтверски пакет R project [46] је софтвер отвореног кода, који, између осталог, нуди и различите врсте класификатора и алгоритама који се користе у машинском учењу. Према Rexer-овом годишњем прегледу рударења података из 2010 [47], R project је био коришћен у 43% пројеката рударења података, што чини овај софтверски алат најкоришћенијим. R project је писан коришћењем програмских језика C и Fortran, као и посебно развијеним скриптним језиком R. Апликације у R project се развијају углавном коришћењем R скрипти, док су неки делови софтвера са интензивним рачунањем повезани са библотекама писаним у језицима C и Fortran. Једна од главних карактеристика софтверског пакета R project је да се може лако проширивати додатним пакетима које развијају сами корисници. Наредни пакети су коришћени у разним експериментима за демонстрацију убрзања: e1071 пакет за SVM-ове, rpart пакет за ортогонална стабла и rsnns пакет за MLP и RBF неуронске мреже. Сви ови пакети су писани у програмском језику C.

Изведене су две групе експеримената. Циљ прве групе експеримената је био да одреди који је оптималан број бита потребан за представљање бројева унутар RMLC архитектуре. Друга група експеримената је имала за циљ да одреди могуће убрзање хардверске имплементације класификатора у односу на софтверску. Експерименти су спроведени над 18 скупова из UCI базе [48] (Табела 7.1).

Скуп	Скраћеница	Број атрибута	Број инстанци
Breast Cancer	bc	9	286
Wisconsin Breast Cancer	bcw	9	699
Credit Approval	ca	15	690
German Credit	cg	20	1000
Horse Colic	col	22	368
Congressional Voting Records	cvr	16	435
Heart Disease	hd	13	270
Hepatitis	hep	19	155
Haberman's Survival	hs	3	306
Ionosphere	ion	34	351
Liver Disorders	ld	6	345
Labor Rleations	lr	16	57
Mushrooms	mus	22	8124
Pima Indians Diabetes	pid	8	768
King Rock vs King Pawn	rvp	36	3196
Spambase	sb	57	4601
Sonar, Mines vs Rocks	son	60	208
Tic-Tac-Toe	ttt	9	958

Табела 7.1: Коришћени скупови података из UCI базе

7.1 Процена оптималне репрезентације бројева

Важан корак у релизацији нумеричких алгоритама коришћењем дигиталног хардвера је бирање одговарајуће репрезентације бројева. Овај корак, који се зове квантизација, треба да одреди најмањи број бита потребних за репрезентацију нумеричких вредности тако да се добије најбоља прецизност прорачуна. Квантизација треба да смањи потребне ресурсе за имплементацију система и повећа брзину класификовања.

У општем случају квантизација уноси грешке у систем које му мењају неке карактеристике. У случају RMLC архитектуре, квантизација вредности смештених у меморијама W RAM и F RAM, као и самих атрибута инстанце утиче на прецизност предикције. Од посебног је значаја испитати утицај квантизације ових вредности на прецизност класификације.

Да би се сагледала осетљивост прецизности RMLC класификатора у односу на репрезентацију бројева, развијен је симулациони модел класификатора који омогућава мењање репрезентације бројева. Коришћењем овог модела, могуће је проценити прецизност класификатора за различите бројевне формате. За сваки сет из

табеле 7.1 спроведен је истоветан експеримент који се састоји од пет десетоструких кросвалидација.

Поставка експеримента који је коришћен да би се проценила оптимална репрезентација бројева коришћењем фиксне тачке је следећа:

- У свакој кросвалидацији се изгради један DT, SVM или ANN класификатор коришћењем стандардне репрезентације реалних бројева са 64-битном покретном тачком за атрибуте, коефицијенте и нелинеарну функцију. За овај корак је коришћен WEKA софтверски алат.

- Направи се скуп DT-ова, SVM-ова и ANN-ова и у сваком се заокруже атрибути, коефицијенти и нелинеарне функције на одговарајућу репрезентацију броја. Процедура за одговарајућу репрезентацију броја је следећа. Прво се одабере укупан број бита са којим ће број бити представљен. Овај укупан број иде од 8 до 32, и повећава се за 4 у сваком кораку. Потом се одреди број бита потребних да се представи највећи целобројни део атрибута из датог скупа. Преостали бити се користе за представљање разломљениог дела. Ова процедура се користи и за одређивање репрезентације коефицијената и вредности нелинеарне функције. У табели 7.2 приказани су детаљи репрезентације бројева за сваку дужину речи коришћену у експериментима, за атрибуте, коефициенте и нелинеарну функцију. Укупно 7 скупова DT, SVM и ANN класификатора са различитим репрезентацијима бројева коришћено је за сваки DT, SVM и ANN класификатор који је користио стандардну 64-битну репрезентацију са покретном тачком.

- Свих 8 класификатора са различитом репрезентацијом бројева се тестирају на одговарајућем скупу и њихова прецизност се бележи.

- Стандардни Студентов т-тест за два узорка је коришћен за поређење процењене прецизности сваког решења које је користило репрезентацију фиксном тачком са решењем које је користило покретну тачку, за сваки тип класификатора. Подразумевано је да се решење са фиксном тачком понаша упоредиво са решењем са покретном тачком ако је статистички значај теста већи од 95%.

Оваква поставка експеримената је већ коришћена са успехом за процену оптималне репрезентације бројева са фиксном тачком приликом

Вредност	8 бита	12 бита	16 бита	20 бита	24 бита	28 бита	32 бита
Атрибут	2,6	4,8	6,10	8,12	8,16	8,20	8,24
Коефицијент	2,6	4,8	6,10	8,12	8,16	8,20	8,24
Нелинеарна функција	6,2	10,2	14,2	16,4	16,8	16,12	16,14

Табела 7.2: Различите репрезентације бројева коришћене у експериментима

Скуп	FP	8 бита	12 бита	16 бита	20 бита	24 бита	28 бита	32 бита
bc	0.817 ± 0.016	**0.789 ± 0.027**	0.814 ± 0.014	0.817 ± 0.016	0.817 ± 0.016	0.817 ± 0.016	0.817 ± 0.016	0.817 ± 0.016
bcw	0.974 ± 0.001	**0.972 ± 0.003**	**0.973 ± 0.002**	0.974 ± 0.002	0.974 ± 0.001	0.974 ± 0.001	0.974 ± 0.001	0.974 ± 0.001
ca	0.904 ± 0.010	**0.715 ± 0.133**	**0.780 ± 0.131**	**0.890 ± 0.015**	0.903 ± 0.090	0.904 ± 0.010	0.904 ± 0.001	0.904 ± 0.001
cg	0.879 ± 0.009	**0.819 ± 0.026**	0.875 ± 0.010	0.879 ± 0.01	0.879 ± 0.009	0.879 ± 0.009	0.879 ± 0.009	0.879 ± 0.009
col	0.918 ± 0.013	**0.849 ± 0.053**	**0.894 ± 0.041**	0.916 ± 0.014	0.918 ± 0.013	0.918 ± 0.013	0.918 ± 0.013	0.918 ± 0.013
cvr	0.973 ± 0.003	**0.970 ± 0.004**	0.972 ± 0.003	0.972 ± 0.003	0.973 ± 0.003	0.973 ± 0.003	0.973 ± 0.003	0.973 ± 0.003
hd	0.865 ± 0.028	**0.857 ± 0.022**	0.864 ± 0.027	0.865 ± 0.028	0.865 ± 0.028	0.865 ± 0.028	0.865 ± 0.028	0.865 ± 0.028
hep	0.905 ± 0.012	**0.893 ± 0.020**	0.904 ± 0.012	0.905 ± 0.012	0.905 ± 0.012	0.905 ± 0.012	0.905 ± 0.012	0.905 ± 0.012
hs	0.791 ± 0.013	**0.758 ± 0.033**	0.791 ± 0.013	0.791 ± 0.013	0.791 ± 0.013	0.791 ± 0.013	0.791 ± 0.013	0.791 ± 0.013
ion	0.947 ± 0.010	**0.884 ± 0.088**	0.944 ± 0.011	0.946 ± 0.01	0.947 ± 0.010	0.947 ± 0.010	0.947 ± 0.010	0.947 ± 0.010
ld	0.764 ± 0.021	**0.642 ± 0.039**	0.746 ± 0.026	0.764 ± 0.021	0.764 ± 0.021	0.764 ± 0.021	0.764 ± 0.021	0.764 ± 0.021
lr	0.979 ± 0.019	**0.960 ± 0.040**	0.976 ± 0.018	0.978 ± 0.019	0.978 ± 0.019	0.979 ± 0.019	0.979 ± 0.019	0.979 ± 0.019
mus	0.997 ± 0.002	**0.590 ± 0.210**	**0.645 ± 0.241**	**0.992 ± 0.008**	0.996 ± 0.003	0.997 ± 0.002	0.997 ± 0.002	0.997 ± 0.002
pid	0.785 ± 0.015	**0.759 ± 0.035**	0.784 ± 0.014	0.785 ± 0.015	0.785 ± 0.0150	0.785 ± 0.015	0.785 ± 0.015	0.785 ± 0.015
rvp	0.987 ± 0.007	**0.934 ± 0.040**	0.990 ± 0.005	0.988 ± 0.007	0.987 ± 0.007	0.987 ± 0.007	0.987 ± 0.007	0.987 ± 0.007
sb	0.969 ± 0.005	N/A	**0.524 ± 0.123**	**0.824 ± 0.091**	**0.945 ± 0.024**	**0.966 ± 0.005**	0.969 ± 0.005	0.969 ± 0.005
son	0.906 ± 0.020	**0.869 ± 0.033**	0.903 ± 0.019	0.906 ± 0.020	0.906 ± 0.020	0.906 ± 0.020	0.906 ± 0.020	0.906 ± 0.020
ttt	0.963 ± 0.004	**0.904 ± 0.004**	0.962 ± 0.003	0.963 ± 0.004	0.963 ± 0.004	0.963 ± 0.004	0.963 ± 0.004	0.963 ± 0.004

Табела 7.3: Просечна прецизност за DT-ове

имплементирања машинских класификатора као дигиталног харвера [49, 50, 22].

Табеле 7.3, 7.4 и 7.5 приказују просечну прецизност класификатора и стандардну девијацију мерену на 50 кросвалидација за DT-ове, SVM-ове и ANN-ове, респективно. У свим табелама колона означена са FP показује прецизност класификатора који ради са покретним зарезом, док наредне колоне приказују прецизност класификатора са фиксним зарезом и све већим бројем бита за репрезентацију вредности.

У табелама 7.3, 7.4 и 7.5, N/A означава случај за који специфицирана квантизација није прихватљива. Подебљаним бројевима приказани су случајеви код којих постоји значајна статистичка разлика између прецизности добијене када су вредности представљене презентацијом покретном тачком и прецизности добијене када су вредности представљене фиксном тачком.

На основу резултата експеримената испоставља се да је за прорачун DT класификатора потребно 28 бита да би се добила прецизност упоредива са класификатором који ради са покретном тачком. Ипак, из табеле 7.3 може се видети да у већини случајева (15 од 18) довољно је свега 16 бита да би се постигла иста прецизност. У случају SVM-ова ситуација је нешто боља. Резултати из табеле 7.4 показују да је 20 бита довољно да се постигну перформансе класификатора са покретном

Скуп	FP	8 бита	12 бита	16 бита	20 бита	24 бита	28 бита	32 бита
bc	0.934 ± 0.008	**0.864 ± 0.080**	0.934 ± 0.008	0.934 ± 0.008	0.934 ± 0.008	0.934 ± 0.008	0.934 ± 0.008	0.934 ± 0.008
bcw	0.971 ± 0.002	0.972 ± 0.002	0.971 ± 0.002	0.971 ± 0.002	0.971 ± 0.002	0.971 ± 0.002	0.971 ± 0.002	0.971 ± 0.002
ca	0.939 ± 0.003	**0.897 ± 0.160**	0.941 ± 0.005	0.939 ± 0.003	0.939 ± 0.003	0.939 ± 0.003	0.939 ± 0.003	0.939 ± 0.003
cg	0.967 ± 0.005	**0.683 ± 0.141**	0.967 ± 0.004	0.967 ± 0.005	0.967 ± 0.005	0.967 ± 0.005	0.967 ± 0.005	0.967 ± 0.005
col	0.975 ± 0.006	**0.676 ± 0.142**	0.975 ± 0.006	0.975 ± 0.006	0.975 ± 0.006	0.975 ± 0.006	0.975 ± 0.006	0.975 ± 0.006
cvr	0.993 ± 0.003	**0.962 ± 0.046**	0.993 ± 0.003	0.993 ± 0.003	0.993 ± 0.003	0.993 ± 0.003	0.993 ± 0.003	0.993 ± 0.003
hd	0.901 ± 0.008	**0.892 ± 0.109**	0.900 ± 0.008	0.901 ± 0.008	0.901 ± 0.008	0.901 ± 0.008	0.901 ± 0.008	0.901 ± 0.008
hep	0.975 ± 0.009	**0.919 ± 0.095**	0.975 ± 0.009	0.975 ± 0.009	0.975 ± 0.009	0.975 ± 0.009	0.975 ± 0.009	0.975 ± 0.009
hs	0.754 ± 0.013	0.754 ± 0.033	0.754 ± 0.012	0.754 ± 0.013	0.754 ± 0.013	0.754 ± 0.013	0.754 ± 0.013	0.754 ± 0.013
ion	0.976 ± 0.005	**0.504 ± 0.152**	**0.770 ± 0.166**	0.976 ± 0.006	0.976 ± 0.005	0.976 ± 0.005	0.976 ± 0.005	0.976 ± 0.005
ld	0.602 ± 0.007	**0.607 ± 0.006**	0.602 ± 0.007	0.602 ± 0.007	0.602 ± 0.007	0.602 ± 0.007	0.602 ± 0.007	0.602 ± 0.007
lr	0.988 ± 0.015	0.979 ± 0.020	0.988 ± 0.015	0.988 ± 0.015	0.988 ± 0.015	0.988 ± 0.015	0.988 ± 0.015	0.988 ± 0.015
mus	1.000 ± 0.000	**0.445 ± 0.040**	**0.935 ± 0.111**	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
pid	0.784 ± 0.002	**0.749 ± 0.031**	0.784 ± 0.004	0.784 ± 0.002	0.784 ± 0.002	0.784 ± 0.002	0.784 ± 0.002	0.784 ± 0.002
rvp	0.999 ± 0.000	**0.506 ± 0.027**	**0.623 ± 0.149**	**0.961 ± 0.041**	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000	0.999 ± 0.000
sb	0.811 ± 0.004	0.810 ± 0.003	0.811 ± 0.004	0.811 ± 0.004	0.811 ± 0.004	0.811 ± 0.004	0.811 ± 0.004	0.811 ± 0.004
son	0.985 ± 0.007	**0.555 ± 0.104**	**0.923 ± 0.095**	0.985 ± 0.007	0.985 ± 0.007	0.985 ± 0.007	0.985 ± 0.007	0.985 ± 0.007
ttt	0.999 ± 0.004	**0.928 ± 0.076**	0.999 ± 0.004	0.999 ± 0.004	0.999 ± 0.004	0.999 ± 0.004	0.999 ± 0.004	0.999 ± 0.004

Табела 7.4: Просечна прецизност за SVM-ове

Скуп	FP	8 бита	12 бита	16 бита	20 бита	24 бита	28 бита	32 бита
bc	0.870 ± 0.045	0.870 ± 0.045	0.870 ± 0.045	0.870 ± 0.045	0.870 ± 0.045	0.870 ± 0.045	0.870 ± 0.045	0.870 ± 0.045
bcw	0.979 ± 0.003	**0.973 ± 0.004**	0.979 ± 0.003	0.979 ± 0.003	0.979 ± 0.003	0.979 ± 0.003	0.979 ± 0.003	0.979 ± 0.003
ca	0.924 ± 0.012	**0.911 ± 0.017**	0.924 ± 0.012	0.924 ± 0.012	0.924 ± 0.012	0.924 ± 0.012	0.924 ± 0.012	0.924 ± 0.012
cg	0.896 ± 0.037	**0.878 ± 0.048**	0.896 ± 0.037	0.896 ± 0.037	0.896 ± 0.037	0.896 ± 0.037	0.896 ± 0.037	0.896 ± 0.037
col	0.941 ± 0.016	0.938 ± 0.017	0.941 ± 0.016	0.941 ± 0.016	0.941 ± 0.016	0.941 ± 0.016	0.941 ± 0.016	0.941 ± 0.016
cvr	0.982 ± 0.005	0.980 ± 0.005	0.982 ± 0.004	0.982 ± 0.005	0.982 ± 0.005	0.982 ± 0.005	0.982 ± 0.005	0.982 ± 0.005
hd	0.910 ± 0.028	**0.876 ± 0.022**	0.909 ± 0.027	0.910 ± 0.028	0.910 ± 0.028	0.910 ± 0.028	0.910 ± 0.028	0.910 ± 0.028
hep	0.937 ± 0.047	**0.932 ± 0.047**	0.937 ± 0.047	0.937 ± 0.047	0.937 ± 0.047	0.937 ± 0.047	0.937 ± 0.047	0.937 ± 0.047
hs	0.757 ± 0.028	**0.728 ± 0.055**	0.757 ± 0.027	0.757 ± 0.028	0.757 ± 0.028	0.757 ± 0.028	0.757 ± 0.028	0.757 ± 0.028
ion	0.972 ± 0.016	**0.962 ± 0.018**	0.972 ± 0.016	0.972 ± 0.016	0.972 ± 0.016	0.972 ± 0.016	0.972 ± 0.016	0.972 ± 0.016
ld	0.618 ± 0.066	**0.590 ± 0.031**	0.618 ± 0.066	0.618 ± 0.066	0.618 ± 0.066	0.618 ± 0.066	0.618 ± 0.066	0.618 ± 0.066
lr	0.909 ± 0.148	0.909 ± 0.148	0.908 ± 0.148	0.909 ± 0.148	0.909 ± 0.148	0.909 ± 0.148	0.909 ± 0.148	0.909 ± 0.148
mus	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
pid	0.775 ± 0.036	**0.724 ± 0.047**	0.773 ± 0.036	0.775 ± 0.036	0.775 ± 0.036	0.775 ± 0.036	0.775 ± 0.036	0.775 ± 0.036
rvp	0.996 ± 0.003	**0.992 ± 0.007**	0.996 ± 0.003	0.996 ± 0.003	0.996 ± 0.003	0.996 ± 0.003	0.996 ± 0.003	0.996 ± 0.003
sb	0.935 ± 0.007	**0.641 ± 0.049**	**0.922 ± 0.011**	0.935 ± 0.007	0.935 ± 0.007	0.935 ± 0.007	0.935 ± 0.007	0.935 ± 0.007
son	0.879 ± 0.166	**0.867 ± 0.164**	0.879 ± 0.166	0.879 ± 0.166	0.879 ± 0.166	0.879 ± 0.166	0.879 ± 0.166	0.879 ± 0.166
ttt	0.987 ± 0.004	0.986 ± 0.004	0.987 ± 0.003	0.987 ± 0.004	0.987 ± 0.004	0.987 ± 0.004	0.987 ± 0.004	0.987 ± 0.004

Табела 7.5: Просечна прецизност за ANN-ове

тачком, док је за већину случајева (14 од 18) свега 12 бита довољно. Ако се посматрају ANN-ови, ситуација је још боља као што је показано у табели 7.5. Коришћењем свега 16 бита постиже се исти резултат као и да се користи репрезентација покретном тачком, а за већину скупова (17 од 18) довољно је свега 12 бита.

7.2 Процена убрзања у случају RMLC архитектуре

Да би се проценило убрзање RMLC архитектуре у односу на софтверски пакет R project, просечна брзина класификовања за једну инстанцу је мерена за сваки од већ спомењутих скупова. За овај експеримент је одабран софтвер R project јер има боље перформансе што се тиче брзине класификовања у односу на WEKA софтвер. R софтвер је покретан под оперативним системом Ubuntu 13.10 Linux OS на процесору AMD Phenom II 1090T (3.2 Ghz) са 16 GB DDR3 RAM системске меморије.

Архитектура RMLC је параметризована да се састоји од 12 RB блокова. Овај број блокова је одређен као већи број од две вредности: максималне дубине стабла и максималног броја слојева неуронских мрежа који се се добијали током експеримената. Величина меморије X RAM је постављена на 128 локација, зато што је максималан број атрибута након конверзије 122, као што се може видети у табели 7.1. Величина меморија W RAM је постављена на 32768 бајтова да би се у њу могао сместити максималан број вектора подршке добијен током експеримената. Величина меморије F RAM је постављена на 4096. На крају, репрезентација фиксном тачком која је коришћена за RMLC и REC архитектуру је имала следећи формат:

- Атрибути су представљени са фиксном тачком која је користила 28 бита. Осам бита је коришћено за целобројни део док је 20 преосталих бита коришћено за разломљени део.
- Коефицијенти су представљени са фиксном тачком која је користила 28 бита са истом дистрибуцијом као и атрибути.
- Вредности нелинеарне функције представљене су фиксном тачком која је користила 28 бита. Шеснаест бита је коришћено за целобројни део док је 12 преосталих бита коришћено за разломљени део.

Архитектура	Слајсови	Флип-флопови	DSP блокови	Блок RAM	Максимална фреквенција
RMLC	1062	1813	12	66	113 MHz

Табела 7.6: Хардверски ресурси неопходни имплементирање RMLC архитектуре коришћене у експериментина за мерење убрзања

Оваква репрезентација је одабрана зато што су експерименти приказани у поглављу 7.1 показали да је то оптимална репрезентација када су у питању скупови приказани у табели 7.1.

Архитектура RMLC је имплементирана коришћењем Xilinx Virtex-7 FPGA чипа. Софтвер Xilinx Vivado 2014.2 коришћен је за логичку синтезу и имплементацију са подразумеваним подешавањима за синтезу и имплементцију. Није коришћена никакава посебна датотека за ограничавање дизајна. Табела 7.6 приказује потребне ресурсе да би се имплементирала архитектура RMLC која је коришћена у експериментима.

Поређење између хардверске и софтверске имплементације је спроведено над следећих 5 врста класификатора који су доступни у оквиру софтверског пакета R project: ортогоналани DT (AP-DT), SVM са полиномијалним кернелом (SVM-P), SVM са радијалним кернелом (SVM-R), MLP ANN (MLP-ANN) и RB ANN (RB-ANN). За сваки скуп из табеле 7.1 покренут је експеримент који се састоји од пет десетоструких кросвалидација. Коришћењем софтвера истренирани су класификатори и измерено је време њихове класификације. Потом је имплементиран исти класификатор коришћењем дигиталних архитектура и измерено је просечно време класификовања.

Време потребно да софтвер R заврши класификовање је мерено од линије у C коду у којој почиње одговарајућа функција за класификацију па до тренутка када та функција заврши свој задатак. Овакав начин мерења је најприближнији времену које је потребно за само класификовање јер он не садржи времена потребна да би се учитали подаци као ни само време позива и повратка из функције. Потребно је још нагласити да софтвер R користи само једно језгро процесора током прорачуна.

Да би се измерило време потребно за класификовање архитектуре RMLC, мерење није почето пре него што су сви подаци унесени у одговарајуће меморије. Додатно, време потребно да се конфигурише сам класификатор није узето у обзир. Број циклуса неопходних за прорачун измерен је и затим помножен са периодом радне фреквенције. На тај начин се добија време потребно да би се класификовала једна инстанца.

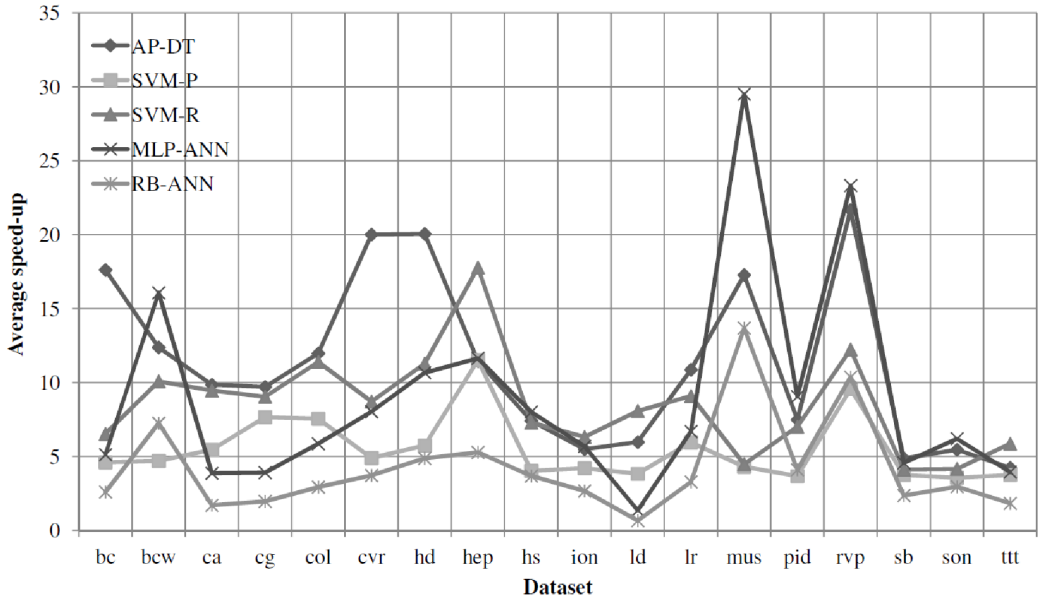
Скуп	AP-DT	SVM-P	SVM-R	MLP-ANN	RB-ANN
bc	17.62	4.60	6.53	5.13	2.60
bcw	12.38	4.70	10.09	16.07	7.25
ca	9.86	5.46	9.46	3.87	1.70
cg	9.71	7.66	9.04	3.91	1.97
col	11.98	7.55	11.4	5.86	2.93
cvr	20.01	4.90	8.72	8.03	3.72
hd	20.05	5.74	11.29	10.67	4.87
hep	11.46	11.52	17.78	11.65	5.28
hs	7.41	4.03	7.29	8.01	3.68
ion	5.49	4.21	6.32	5.68	2.67
ld	5.97	3.83	8.07	1.36	0.67
lr	10.87	5.95	9.09	6.69	3.28
mus	17.28	4.30	4.48	29.51	13.69
pid	7.48	3.67	7.01	9.06	4.10
rvp	21.67	9.57	12.24	23.33	10.38
sb	4.87	3.75	4.12	4.57	2.36
son	5.46	3.56	4.16	6.20	2.96
ttt	4.23	3.75	5.86	3.95	1.83
Просечно убрзање	11.32	5.49	8.50	9.09	4.22

Табела 7.7: Просечно убрзање RMLC архитектуре у односу софтвер R

Просечно убрзање које пружа FPGA имплементација RMLC архитектуре над софтвером R је процењено на просечном времену неопходном за класификацију на свим скуповима приказаним у табели 7.1. Процедура за мерење убрзања је спроведена над свим типовима класификатора које пружа софтвер R и резултати су приказани у табели 7.7 као и на слици 7.1.

Као што се може видети из табеле 7.7, архитектура RMLC пружа значајно убрзање у односу на софтвер R за све типове машинских класификатора који су коришћени у експериментима. Просечно убрзање RMLC архитектуре у односу на софтвер R је 11.32 (5.49, 8.50, 9.09, 4.22) за AP-DT (SVM-P, SVM-R, MLP-ANN, RB-ANN). Ова убрзања су постигнута на хардверској архитектури која ради на свега 113 Mhz, која је 28.32 пута спорија од процесора коришћеног да извршава софтвер R. Ови резултати су постигнути коришћењем ресурса који су веома скромни у односу на оно што пружају најмодернији FPGA чипови.

До сада приказана убрзања су процењена без урачунатог времена за трансфер података до FPGA чипа. Време трансфера података зависи од



Слика 7.1: Просечно убрзање RMLC архитектуре у односу на софтвер R

података који се преносе као и од брзине комуникације која се користи. На пример, у случају да се користи USB 3.0 комуникација, теоретски максимална брзина трансфера би била до 5 Gbit/s, за SATA-III 6 Gbit/s, PCI Express 3.0 нешто испод 7.877 Gbit/s по линији, док Ethernet комуникација може да иде и до 100 Gbit/s. Одабир комуникационог линка значајно може да утиче на време трансфера података.

Уколико би се узело време трансфера потребног да се коришћењем једног 10Gbit/s Ethernet линка пребаце подаци на FPGA чип убрзања из табеле 7.7 би била другачија. Табела 7.8 приказује убрзања која се добијају када се урачуна и време потребно за трансфер података.

Уколико се упореде резултати из табеле 7.7 са резултатима из табеле 7.8 и 7.11 могу се извући следећи закључци. У случају RMLC архитектуре, за AP-DT класификатор, трансфер података би у просеку смањио убрзање за 4.47%. Минимално и максимално смањење убрзања је 0.61% и 12.56%. У случају SVM-P просечно смањење је 0.58% док су минимално и максимално 0.02% и 2.08%. За SVM-R имплементацију, просечно смањење је 0.56%, док су минимално и максимално 0.02% и 2.04%. У случају MLP-ANN просечно смањење је 0.68% док су минимално и максимално 0.08% и 1.84%. И на крају, у случају RB-ANN имплементације просечно смањење би било 0.65%, док би минимално

Скуп	AP-DT	SVM-P	SVM-R	MLP-ANN	RB-ANN
bc	17.489	4.597	6.526	5.125	2.597
bcw	11.448	4.613	9.917	15.868	7.166
ca	9.672	5.454	9.450	3.860	1.696
cg	9.514	7.655	9.035	3.899	1.965
col	11.776	7.540	11.385	5.847	2.924
cvr	19.687	4.882	8.690	8.011	3.711
hd	19.694	5.726	11.263	10.641	4.858
hep	10.917	11.397	17.597	11.568	5.244
hs	6.781	4.006	7.248	7.901	3.633
ion	5.009	4.140	6.218	5.607	2.637
ld	5.651	3.819	8.048	1.347	0.664
lr	10.571	5.861	8.959	6.664	3.268
mus	17.146	4.299	4.479	29.481	13.676
pid	7.112	3.663	6.998	8.986	4.069
rvp	21.293	9.559	12.225	23.274	10.356
sb	4.260	3.747	4.117	4.486	2.317
son	4.774	3.486	4.075	6.086	2.906
ttt	4.204	3.748	5.857	3.947	1.829
Просечно убрзање	10.94	5.46	8.45	9.03	4.20

Табела 7.8: Просечно убрзање RMLC архитектуре у односу на софтвер R са урачунатим временом трансфера података

и максимално било 0.05% и 1.82%. Из приказаних резултата јасно је да су смањења убрзања у сличају SVM-ова и ANN-ова минимална. У случају DT смањење је значајније али је и даље испод 10% за већину скупова коришћених у експериментима. Трансфер података има највећи утицај на DT-ове зато што су они најмање захтевни за прорачун. Време потребно да би се класификовала инстанца за DT класификатор је најкраће па стога додато трансфер време најнеповољније утиче на овај тип класификатора.

Треба напоменути да, уколико се у некој апликацији испостави да је смањење убрзање због времена трансфера велико, оно се може и потпуно елиминисати. Архитектуре RMLC и REC могу се користити у проточном режиму или се може имплементирати двоструки бафер. У проточном режиму подаци се не би смештали на FPGA чипу, већ би се одмах класификовали док би се резултат проточно слао до примаоца. Ако би се користио двоструки бафер, тада би се две меморије неизменично користиле за примање података. Док се у једну меморију примају подаци, дотле би се подаци из друге меморије класификовали и обрнуто. На ова два начина је могуће у потпуности уклонити утицај времена трансфера података на убрзање самих архитектура.

7.3 Процена убрзања у случају REC архитектуре

Архитектура REC се може користити и за имплементацију појединачних класификатора као и за реализацију ансамбала. У наставку поглавља прво ће бити описани експерименти везани за појединачне класификаторе а потом за ансамбле.

Архитектура REC коришћена за експерименте убрзања појединачних класификатора састојала се од 15 врста а свака врста је имала по 12 RB блокова са репрезентацијом бројева која је иста као и за RMLC архитектуру.

Поставка експеримената је иста као и за архитектуру RMLC описану у одељку 7.2. И у овом случају је коришћен софтвер R project а скраћенице за коришћене класификаторе су исте: AP-DT, SVM-P, SVM-R, MLP-ANN и RB-ANN.

Архитектура REC је имплементирана коришћењем Xilinx Virtex-7 FPGA чипа. Софтвер Xilinx Vivado 2014.2 коришћен је за логичку синтезу и имплементацију са подразумеваним подешавањима за синтезу и имплементцију. Није коришћена никакава посебна датотека за

Архитектура	Слајсови	Флип-флопови	DSP блокови	Блок RAM	Максимална фреквенција
REC	9680	24541	180	954	113 MHz

Табела 7.9: Хардверски ресурси неопходни за имплементирање REC архитектуре коришћене у експериментина за мерење убрзања

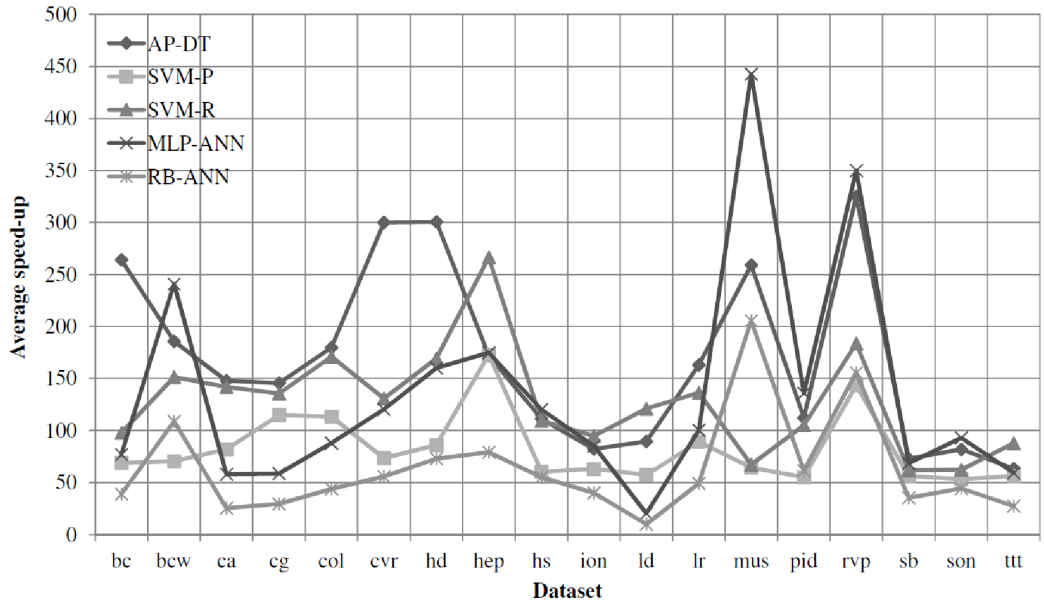
Скуп	AP-DT	SVM-P	SVM-R	MLP-ANN	RB-ANN
bc	264.30	69.00	97.95	76.95	39.00
bcw	185.70	70.50	151.35	241.05	108.75
ca	147.90	81.90	141.90	58.05	25.50
cg	145.65	114.9	135.60	58.65	29.55
col	179.70	113.25	171.00	87.90	43.95
cvr	300.15	73.50	130.80	120.45	55.80
hd	300.75	86.10	169.35	160.05	73.05
hep	171.90	172.8	266.70	174.75	79.20
hs	111.15	60.45	109.35	120.15	55.20
ion	82.35	63.15	94.80	85.20	40.05
ld	89.55	57.45	121.05	20.40	10.05
lr	163.05	89.25	136.35	100.35	49.20
mus	259.20	64.50	67.20	442.65	205.35
pid	112.20	55.05	105.15	135.90	61.50
rvp	325.05	143.55	183.60	349.95	155.7
sb	73.05	56.25	61.80	68.55	35.40
son	81.90	53.40	62.40	93.00	44.40
ttt	63.45	56.25	87.90	59.25	27.45
Просечно убрзање	169.83	82.29	127.46	136.29	63.28

Табела 7.10: Просечно убрзање REC архитектуре у односу на софтвер R

ограничавање дизајна. Табела 7.9 приказује потребне ресурсе да би се имплементирала архитектура REC која је коришћена у експериментима.

Просечно убрзање које пружа FPGA имплементације REC архитектуре над софтвером R је процењено на просечном времену неопходном за класификацију на свим скуповима приказаним у табели 7.1. Процедура за мерење убрзања је спроведена над свим типовима класификатора које пружа софтвер R и резултати су приказани у табели 7.10 као и на слици 7.2.

Као што се може видети из табеле 7.10, архитектура REC је очекивано постигла значајно већа убрзања од RMLC архитектуре . Ова архитетура је убрзала калсификаторе AP-DT (SVM-P, SVM-R, MLP-ANN, RB-ANN)



Слика 7.2: Просечно убрзање REC архитектуре у односу на софтвер R

просечно 69.83 (82.29, 127.46, 136.29, 63.28) пута. И у овом случају нису узета времена потребна за трансфер података на FPGA чип.

Уколико би се узело време трансфера потребног да се коришћењем једног 10Gbit/s Ethernet линка пребаце подаци на FPGA чип убрзања из табеле 7.10 би била другачија. Табела 7.11 приказује убрзања која се добијају када се урачуна и време потребно за трансфер података.

Анализом података из табела 7.10 и 7.11 види се да је утицај трансфер времена другачије него у случају RMLC архитектуре. У случају REC архитектуре смањење убрзања ће бити значајније него у случају RMLC архитектуре. То је стога што је време класификовања по једној инстанци још краће па додато време трансфера утиче значајније. И у овом случају за очекивати је да ће убрзање DT класификатора бити највише смањено. Ако се упореде резултати долази се до закључка да се убрзање класификатора AP-DT у просеку смањило 34.93%, док је минимално и максимално смањење 8.51% и 68.32%. У случају SVM-P класификатора, прсечно смањење је 7.34% док су минимално и максимално 0.30% 24.13%, а за SVM-R класификаторе, просечна вредност је 7.09% док су најмања и највећа вредност 0.30% и 23.87%. У случају ANN-ова ситуација је слична те је код MLP-ANN просечно смањење 8.77%, а најмање и највеће су 1.19% and 22.00% а код RB-ANN, просечно је 8.43%, а најмање и највеће су 1.18% and 21.72%.

Скуп	AP-DT	SVM-P	SVM-R	MLP-ANN	RB-ANN
bc	237.573	68.385	97.076	75.813	38.424
bcw	83.611	54.992	120.024	202.478	92.514
ca	114.562	80.571	139.597	55.871	24.578
cg	111.268	113.875	134.39	56.367	28.4
col	142.676	111.084	167.729	84.969	42.546
cvr	240.891	69.734	124.453	116.264	53.939
hd	236.532	82.988	163.465	153.82	70.41
hep	98.482	148.709	230.709	157.93	71.869
hs	46.467	55.383	100.598	99.544	46.192
ion	33.729	50.379	76.083	71.297	33.684
ld	48.471	55.055	116.338	17.879	8.914
lr	114.461	72.768	111.854	94.804	46.547
mus	232.05	64.307	66.999	436.108	202.315
pid	63.193	53.525	102.535	121.015	55.281
rvp	256.855	141.012	180.354	337.79	150.507
sb	23.216	55.583	61.158	53.471	27.71
son	25.942	40.516	47.507	72.543	34.755
ttt	58.051	55.831	87.246	58.547	27.125
Просечно убрзање	120.446	76.372	118.229	125.917	58.651

Табела 7.11: Просечно убрзање REC архитектуре у односу на софтвер R са урачунатим временом трансфера података

Време трансфера података се може у потпуности уклонити неким од поступака описаним у поглављу 7.2.

Архитектура REC може да ради и као део система за имплементацију ансамбала класификатора. Да би се проценило убрзање које архитектура пружа у односу на постојећа софтверска решења одабран је софтверски алат Waikato Environment for Knowledge Analysis (WEKA). WEKA је софтверски алат који је већим делом написан у програмском језику Java. Обично се програми написани у Java програмском језику извршавају спорије него програми написани у програмиским језицима C/C++ и Fortran. За поређење је одабран програмски пакет WEKA зато што има одличну подршку за ансамбле класификатора. Програмски пакет R project није узет, и поред бољих перформанси, јер недостаје имплементација ансамбала.

Програмски пакет WEKA подржава и функционална стабла одлука па су у овом експериментима и ова стабла узета у обзир.

Поређење између хардверске и софтверске имплементације је спроведено над следећих 10 врста ансамбала који се могу имплементиати у оквиру софтверског пакета WEKA:

- хомогени ансамбл функционалних DT (F-DT EC)
- хомогени ансамбл ортогоналаних DT (AP-DT EC)
- хомогени ансамбл SVM са полиномијалним кернелом (SVM-P EC)
- хомогени ансамбл SVM са радијалним кернелом (SVM-R EC)
- хомогени ансамбл MLP-ANN (MLP-ANN EC)
- хомогени ансамбл RB ANN (RB-ANN EC)
- хетерогени ансамбл састављен од DT и SVM класификатора (DTSVM EC)
- хетерогени ансамбл састављен од DT и ANN класификатора (DTANN EC)
- хетерогени ансамбл састављен од SVM и ANN класификатора (SVMANN EC)
- хетерогени ансамбл састављен од DT, SVM и ANN класификатора (DTSVMANN EC)

Скуп	F-DT EC	AP-DT EC	SVM-P EC	SVM-R EC	MLP-ANN EC	RB-ANN EC	DTSVM EC	DTANN EC	SVMANN EC	DTSVMANN EC
bc	226867.70	4474.71	360.72	331.75	6768.00	10403.70	949.93	38035.02	382.68	1071.41
bcw	38942.77	2559.62	260.20	188.36	4022.26	4875.46	594.47	8836.78	378.56	837.25
ca	195000.00	2444.00	281.00	296.00	4468.00	7052.00	597.00	31451.00	273.00	660.00
cg	499509.00	2300.00	329.00	299.00	4584.00	7613.00	615.00	63953.00	318.00	670.00
col	306018.13	4099.70	355.63	359.95	6470.92	10712.49	899.05	45513.60	381.70	1034.82
cvr	53352.94	3823.53	353.17	167.13	5220.59	6495.10	641.22	13284.31	357.77	775.61
hd	61144.00	5679.00	272.00	184.00	6921.00	7454.00	592.00	14631.00	357.00	799.00
hep	88194.00	8439.00	542.00	298.00	10238.00	11376.00	1288.00	22200.00	756.00	1770.00
hs	79538.00	6022.00	236.00	169.00	6691.00	7527.00	378.00	15926.00	241.00	484.00
ion	150047.62	3650.79	322.01	214.25	5339.29	7392.86	1042.86	24429.89	412.74	1312.33
ld	61803.23	5861.29	150.11	108.40	6491.94	5873.66	263.86	14529.57	178.03	351.08
lr	130784.31	13980.39	1992.01	820.29	18602.94	23112.75	4573.98	37957.52	2563.65	6519.69
mus	404272.74	843.11	458.06	462.73	4734.65	10255.78	450.23	43868.88	396.21	436.05
pid	55420.00	2330.00	191.00	166.00	3703.00	4216.00	191.00	10485.00	147.00	224.00
rvp	152587.00	960.00	183.00	195.00	2754.00	4179.00	274.00	20020.00	195.00	256.00
son	239840.00	5781.00	427.00	306.00	6918.00	11275.00	1449.00	37718.00	521.00	1684.00
ttt	122978.00	1628.00	185.00	180.00	3135.00	4558.00	257.00	18166.00	162.00	280.00
Просечно убрзање	181309.25	4218.73	395.44	277.60	6110.37	8308.49	852.31	28173.64	456.80	1080.85

Табела 7.12: Просечно убрзање REC архитектуре у односу на софтвер WEKA

Скуп	F-DT EC			SVM-P EC			MLP-ANN EC		
	Време (WEKA)	Време (REC)	Убрзање	Време (WEKA)	Време (REC)	Убрзање	Време (WEKA)	Време (REC)	Убрзање
Higgs	181977.8 s	0.47826 s	380498.9	249400.0 s	1733.78 s	143.847	1800 s	0.76522 s	2352.27
Susy	21088.9 s	0.21739 s	97008.9	40955.6 s	256.83 s	159.468	800 s	0.34783 s	2300.00
Skin	292.9 s	0.01065 s	27487	312.7 s	4.67 s	70	41.8 s	0.01705 s	2452.4

Табела 7.13: Време потребно да би се класификовале све инстанце великих скупова

За 17 скупова из табеле 7.1 покренут је експеримент који се састоји од пет десетоструких кросвалидација. Коришћењем софтвера истренирани су класификатори и измерено је време њихове класификације. Потом је имплементиран исти класификатор коришћењем дигиталних архитектура и измерено је просечно време класификовања. Мерење времена софтверског решења у хардверске архитектуре је спроведено на начин описан у поглављу 7.2.

Као што табела 7.12 показује REC архитектура пружа убрзања од 2 до 5 редова величине већа од WEKA софтверских решења за све врсте ансамбала класификатора. Просечно убрзање REC архитектуре у односу на WEKA софтвер је 181309.25 (4218.73, 395.44, 277.6, 6110.37, 8308.49) за хомогене ансамбле класификатора и 852.31 (28173.64, 456.8, 1080.85) за хетерогене ансамбле.

На крају, да би се боље илустровало потенцијална уштеда времена када се користи REC реконфигурабилна архитектура, узета су три скупа са великим бројем инстанци из UCI базе скупова. У табели 7.13 приказана су времена потребна да би се класификовале све инстанце ових скупова за WEKA софтвер и REC архитектуру.

Скуп Higgs садржи 11000000 инстанци, има 28 атрибута и инстанце треба да се класификују у две класе. Скуп Susy садржи 5000000 инстанци, има 18 атрибута и инстанце треба да се класификују у две

класе. Скуп Skin садржи 245057 инстанци, има 2 атрибута и инстанце треба да се класификују у две класе.

Када се анализирају резултати из табеле 7.13, види се да се време потребно за класификовање свих инстанци великих скупова може значајно смањити уколико се користи REC архитектура уместо WEKA софтвера. За неке скупове, WEKA софтверу су потребни сати, чак и дани да заврши класификовање док архитектура REC исти задатак може да уради за време мерено у минутима, чак и секундима. Резултати у табелама 7.10 и 7.13 показују да се перформансе архитектуре REC одлично скалирају са повећањем величине скупа са којим се ради. Ови примери показују пун потенцијал који пружа хардверска акцелерација машинских класификатора.

Глава 8

Закључак

У докторској дисертацији представљена је универзална реконфигурабилна дигитална архитектура грубог степена гранулације која може да реализује стабла одлука - DT, SVM као и вештачке неуронске мреже - ANN. Уз то, показана је и архитектура која може да имплементира хомогене као и хетерогене ансамбле наведених класификатора. Архитектуре су погодне за имплементацију како на технологији FPGA тако и на технологији ASIC. У дисертацији је приказано како се архитектуре користе за имплементацију више врста DT-ова, SVM-ова и ANN-ова као и хомогених и хетерогених ансамбала ових класификатора. Имплементација архитектура на FPGA чипу је поређена са R и WEKA софтверским имплементацијама на већем броју стандардних проблема из UCI базе података за проблеме машинске класификације. За три врсте класификатора детаљно је илустровано како RMLC архитектура ради на конкретним примерима предикције.

Ефикасност рада архитектура показана је на већем броју експеримената над скуповима података из UCI базе података. Резултати експеримената показују да приказане архитектуре остварују убрзања и до неколико редова величина већа у односу на имплементације истих класификатора софтверским решењима. Архитектуре су поређене са R project и WEKA софтверским пакетима. Архитектура RMLC остварила је ред величине краће време предикције у односу на софтвер R project, при чему је постигнута иста прецизност класификовања. Архитектура REC има убрзање које је два реда величине веће него софтвер R project док убрзање иде и до шест редова величине у поређењу са WEKA софтверским пакетом. При томе, постигнута је иста прецизност предикције. Архитектуре су имплементиране на FPGA чипу и радиле су на скоро 30 пута споријој фреквенцији у односу на CPU на коме су реализована софтверска решења.

Референце

[1] Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[2] Kevin Patrick Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[3] Simon O. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 2008.

[4] Crina Grosan and Ajith Abraham. *Decision Trees, Intelligent Systems: A Modern Approach*. Springer, 2011.

[5] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers-a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35(4):476–487, 2005.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[7] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[8] Xindong Wu and Vipin Kumar. *The Top Ten Algorithms in Data Mining*. Chapman and Hall, 2009.

[9] Lior Rokach and Oded Maimon. *Data Mining With Decision Trees: Theory and Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2nd edition, 2014.

[10] Lipo Wang. *Data Mining with Computational Intelligence*. Springer-Verlag, Berlin, Heidelberg, 2009.

[11] Simon JD Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[12] Sudha Challa. *Fundamentals of object tracking.* Cambridge University Press, 2011.

[13] Seunghun Jin, Dongkyun Kim, Thuy Tuong Nguyen, Daijin Kim, Munsang Kim, and Jae Wook Jeon. Design and implementation of a pipelined datapath for high-speed face detection using fpga. *Industrial Informatics, IEEE Transactions on*, 8(1):158–167, 2012.

[14] Ming Yang, James Crenshaw, Bruce Augustine, Russell Mareachen, and Ying Wu. Adaboost-based face detection for embedded systems. *Computer Vision and Image Understanding*, 114(11):1116–1125, 2010.

[15] Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up machine learning: Parallel and distributed approaches.* Cambridge University Press, 2011.

[16] Alok N Choudhary, Daniel Honbo, Prabhat Kumar, Berkin Ozisikyilmaz, Sanchit Misra, and Gokhan Memik. Accelerating data mining workloads: current approaches and future challenges in system architecture design. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):41–54, 2011.

[17] *Apache hadoop.* http://hadoop.apache.org.

[18] Rastislav JR Struharik and Ladislav A Novak. Evolving decision trees in hardware. *Journal of Circuits, Systems, and Computers*, 18(06):1033–1060, 2009.

[19] Grigorios Chrysos, Panagiotis Dagritzikos, Ioannis Papaefstathiou, and Apostolos Dollas. Hc-cart: A parallel system implementation of data mining classification and regression tree (cart) algorithm on a multi-fpga system. *ACM Trans. Archit. Code Optim.*, 9(4):47:1–47:25, January 2013.

[20] Qingzheng Li and Amine Bermak. A low-power hardware-friendly binary decision tree classifier for gas identification. *Journal of Low Power Electronics and Applications*, 1(1):45–58, 2011.

[21] Rastislav JR Struharik and Ladislav A Novak. Hardware implementation of decision tree ensembles. *Journal of Circuits, Systems, and Computers*, 22(05), 2013.

[22] Rastislav JR Struharik and Ladislav A Novak. Intellectual property core implementation of decision trees. *IET computers & digital techniques*, 3(3):259–269, 2009.

[23] JR Struharik. Implementing decision trees in hardware. In *Intelligent Systems and Informatics (SISY), 2011 IEEE 9th International Symposium on*, pages 41–46. IEEE, 2011.

[24] Davide Anguita, Andrea Boni, and Sandro Ridella. A digital architecture for support vector machines: theory, algorithm, and fpga implementation. *Neural Networks, IEEE Transactions on*, 14(5):993–1009, 2003.

[25] Ta-Wen Kuan, Jhing-Fa Wang, Jia-Ching Wang, Po-Chuan Lin, and Gaung-Hui Gu. Vlsi design of an svm learning core on sequential minimal optimization algorithm. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(4):673–683, 2012.

[26] Chih-Hsiang Peng, Po-Chuan Lin, Shovan Barma, Jhing-Fa Wang, Hong-Yuan Peng, Karunanithi Bharanitharan, and Ta-Wen Kuan. Low-power enhanced system-on-chip design for sequential minimal optimisation learning core with tri-layer bus and butterfly-path accelerator. *IET Computers & Digital Techniques*, 2015.

[27] Davide Anguita, Luca Carlino, Alessandro Ghio, and Sandro Ridella. A fpga core generator for embedded classification systems. *Journal of Circuits, Systems, and Computers*, 20(02):263–282, 2011.

[28] Christos Kyrkou and Theocharis Theocharides. A parallel hardware architecture for real-time object detection with support vector machines. *Computers, IEEE Transactions on*, 61(6):831–842, 2012.

[29] V Vranjkovic and Rastislav Struharik. New architecture for svm classifier and its application to telecommunication problems. In *Telecommunications Forum (TELFOR), 2011 19th*, pages 1543–1545. IEEE, 2011.

[30] Davood Mahmoodi, Ali Soleimani, Hossein Khosravi, Mehdi Taghizadeh, et al. Fpga simulation of linear and nonlinear support vector machine. *Journal of Software Engineering and Applications*, 4(05):320, 2011.

[31] Janardan Misra and Indranil Saha. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, 74(1):239–255, 2010.

[32] Amos R Omondi and Jagath Chandana Rajapakse. *FPGA implementations of neural networks*, volume 365. Springer, 2006.

[33] Hirokazu Madokoro and Kazuhito Sato. Hardware implementation of back-propagation neural networks for real-time video image learning and processing. *Journal of Computers*, 8(3):559–566, 2013.

[34] Antony Savich, Medhat Moussa, and Shawki Areibi. A scalable pipelined architecture for real-time computation of mlp-bp neural networks. *Microprocessors and Microsystems*, 36(2):138–150, 2012.

[35] Dmitri Vainbrand and Ran Ginosar. Scalable network-on-chip architecture for configurable neural networks. *Microprocessors and Microsystems*, 35(2):152–166, 2011.

[36] João Gama. Functional trees. *Machine Learning*, 55(3):219–250, 2004.

[37] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[38] Stamatis Vassiliadis and Dimitrios Soudris. *Fine-and coarse-grain reconfigurable computing*, volume 16. Springer, 2007.

[39] Scott Hauck and Andre DeHon. *Reconfigurable computing: the theory and practice of FPGA-based computation*. Morgan Kaufmann, 2010.

[40] João MP Cardoso and Pedro C Diniz. *Compilation techniques for reconfigurable architectures*, volume 81. Springer Science & Business Media, 2011.

[41] James E Gentle, Wolfgang Karl Härdle, and Yuichi Mori. *Handbook of computational statistics: concepts and methods*. Springer Science & Business Media, 2012.

[42] M Ozay and F Vural. Performance analysis of stacked generalization classifiers. In *Signal Processing, Communication and Applications Conference, 2008. SIU 2008. IEEE 16th*, pages 1–4. IEEE, 2008.

[43] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[44] YS Huang and CY Suen. The behavior-knowledge space method for combination of multiple classifiers. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 347–347. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 1993.

[45] Rastislav Struharik and Ladislav Novak. Hardware implementation of decision tree ensembles. *Journal of Circuits, Systems and Computers*, 2013.

[46] *R Project*, 2013. http://www.r-project.org.

[47] *Apache hadoop*, 2013. http://www.rexeranalytics.com.

[48] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. *UCI Repository of machine learning databases*, 1998. http://www.ics.uci.edu/ mlearn/ML-Repository.html.

[49] Krste Asanović, Nelson Morgan, and John Wawrzynek. Using simulations of reduced precision arithmetic to design a neuro-microprocessor. *Journal of VLSI signal processing systems for signal, image and video technology*, 6(1):33–44, 1993.

[50] Davide Anguita, Alessandro Ghio, Stefano Pischiutta, and Sandro Ridella. A support vector machine with integer parameters. *Neurocomputing*, 72(1):480–489, 2008.