Univerzitet u Nišu
Elektronski fakultet

Miloš M. Radmanović

Razvoj algoritama za izračunavanje autokorelacije prekidačkih funkcija preko dijagrama odlučivanja

doktorska disertacija

Niš, 2014.

Univerzitet u Nišu
Elektronski fakultet

Miloš M. Radmanović

Razvoj algoritama za izračunavanje autokorelacije prekidačkih funkcija preko dijagrama odlučivanja

doktorska disertacija

Niš, 2014.

University of Niš
Faculty of Electronic Engineering

Miloš M. Radmanović

Development of Algorithms for Computation of the Autocorrelation of Switching Functions over Decision Diagrams

Doctoral Dissertation

Niš, 2014

I Autor	
Ime i prezime	**Miloš Radmanović**
Sadašnje zaposlenje	Elektronski fakultet, Univerzitet u Nišu
II Doktorska disertacija	
Naslov	**Razvoj algoritama za izračunavanje autokorelacije prekidačkih funkcija preko dijagrama odlučivanja**
Ključne reči	prekidačka teorija, prekidačke funkcije, spektralne transformacije, autokorelacija, FFT, dizajn i analiza algoritama, dijagrami odlučivanja, BDD, BDD paket
Broj stranica	119 + xxiv
Broj slika	42
Broj tabela	12
Broj bibliografskih jedinica	96
Ustanova i mesto gde je disertacija realizovana	Elektronski fakultet Univerzitet u Nišu
Naučno polje	Tehničko-tehnološke nauke
Naučna oblast	Računarstvo i informatika
UDK	004.312 004.421 004.422.63
Mentor	**prof. dr Radomir S. Stanković**, redovni profesor Elektronski fakultet, Univerzitet u Nišu

I Author	
Name and surname	**Miloš Radmanović**
Date and place of birth	June 16, 1976, Niš
Current workplace	Faculty of Electronic Engineering, University of Niš

II Doctoral Dissertation	
Title	**Development of Algorithms for Computation of the Autocorrelation of Switching Functions over Decision Diagrams**
Keywords	switching theory, switching functions, spectral transform, autocorrelation, FFT, design and analysis of algorithms, decision diagram, BDD, BDD package
Number of pages	119 + xxiv
Number of figures	42
Number of tables	12
Number of references	96
Institution at which the dissertation has been concluded	Faculty of Electronic Engineering University of Niš
Scientific field	technical sciences
Scientific area	computer science
UDC	004.312 004.421 004.422.63
Advisor	**Prof. Dr. Radomir S. Stanković**, Full Profesor Faculty of Electronic Engineering, University of Niš

Tok prijave doktorske disertacije	
Datum prijave teme disertacije	12.02.2014.
Broj i datum odluke o prihvatanju teme doktorske disertacije	NNV br. 07/03-016/14-005 24.04.2014. NSV br. 08/20-01-004/14-020 27.05.2014.
Komisija za pisanje izveštaja o naučnoj zasnovanosti teme	dr Radomir Stanković, redovni profesor *Elektronski fakultet, Univerzitet u Nišu* dr Dragan Janković, redovni profesor *Elektronski fakultet, Univerzitet u Nišu* dr Zoran Ognjanović, naučni savetnik *Matematički institut SANU, Beograd* dr Branimir Todorović, vanredni profesor *Prirodno matematički fakultet, Univerzitet u Nišu* dr Suzana Stojković, docent *Elektronski fakultet, Univerzitet u Nišu*
Komisija za ocenu i odbranu doktorske disertacije	dr Radomir S. Stanković, redovni profesor *Elektronski fakultet, Univerzitet u Nišu* dr Dragan Janković, redovni profesor *Elektronski fakultet, Univerzitet u Nišu* dr Zoran Ognjanović, naučni savetnik *Matematički institut SANU, Beograd* dr Branimir Todorović, vanredni profesor *Prirodno matematički fakultet, Univerzitet u Nišu* dr Suzana Stojković, docent *Elektronski fakultet, Univerzitet u Nišu*
Datum predaje doktorske disertacije Datum odbrane doktorske disertacije	11.12.2014.

Doctoral Dissertation Application Process	
Date of the topic application	February 12, 2014
Number and date of the topic approval decision	NNV no. 07/03-016/14-005 April 24, 2014 NSV no. 08/20-01-004/14-020 May 27, 2014
Committee for the approval of the doctoral dissertation topic	Dr. Radomir S. Stanković, Full Professor *Faculty of Electronic Engineering, University of Niš* Dr. Dragan Janković, Full Professor *Faculty of Electronic Engineering, University of Niš* Dr. Zoran Ognjanović, Research Professor *Mathematical Institute SANU, Belgrade* Dr. Branimir Todorović, Associate Professor *Faculty of Sciences, University of Niš* Dr. Suzana Stojković, Docent *Faculty of Electronic Engineering, University of Niš*
Doctoral dissertation defence committee	Dr. Radomir S. Stanković, Full Professor *Faculty of Electronic Engineering, University of Niš* Dr. Dragan Janković, Full Professor *Faculty of Electronic Engineering, University of Niš* Dr. Zoran Ognjanović, Research Professor *Mathematical Institute SANU, Belgrade* Dr. Branimir Todorović, Associate Professor *Faculty of Sciences, University of Niš* Dr. Suzana Stojković, Docent *Faculty of Electronic Engineering, University of Niš*
Date of the submission of the doctoral dissertation Date of the defence of the doctoral dissertation	December 11, 2014

Naučni doprinos doktorske disertacije

- Pregled, diskusija i dopuna neophodne teorijske osnove u skladu sa razmatranim problemom.
- Pregled i analiza postojećih algoritama za izračunavanje autokorelacionih koeficijenata.
- Razvoj generalizovanih algoritama za izračunavanje pojedinačnih i kompletnih autokorelacionih koeficijenata višeizlaznih prekidačkih funkcija korišćenjem raznih vrsta dijagrama odlučivanja.
- Pregled i analiza osnovnih principa koji se koriste kod implementacije standardnih BDD paketa.
- Implementacija razvijenih algoritama za izračunavanje autokorelacionih koeficijenata korišćenjem raznih vrsta dijagrama odlučivanja.
- Komparativna analiza performansi postojećih i razvijenih algoritma za izračunavanje autokorelacionih koeficijenata na osnovu standardnog skupa benčmark funkcija.

Scientific Contribution of the Doctoral Dissertation

- Overview, discussion, and complement of the necessary theoretical basis in accordance with considered problem.
- Overview and analysis of the existing algorithms for the computation of autocorrelation coefficients.
- The development of generalized algorithms for calculating the individual and complete autocorrelation coefficients of multiple-output switching functions using various types of decision diagrams.
- Overview and analysis of the basic principles used in the implementation of standard BDD-packages.
- The implementation of the developed algorithms for the computation of autocorrelation coefficients using various types of decision diagrams.
- The comparative performance analysis of existing and proposed algorithms for computation of autocorrelation coefficients using a standard set of benchmark functions.

Sažetak

Razvoj algoritama za izračunavanje autokorelacije prekidačkih funkcija preko dijagrama odlučivanja

Autokorelacija je matematička operacija sa značajnim primenama u oblasti računarske tehnike i inžinjerstva. Prostorna i vremenska kompleksnost algoritama za izračunavanje autokorelacije je eksponencijalana u odnosu na broj promenljivih prekidačke funkcije. Većína postojećih algoritama je fokusirana na izračunavanja autokorelacionih koeficijenta jednoizlaznih prekidačkih funkcija. Međutim, u praktičnim primenama se često zahteva rad sa višeizlaznim prekidačkim funkcijama. Sa tom motivacijom, ova doktorska disertacija opisuje nove algoritme za efikasno izračunavanje kompletne totalne autokorelacije za višeizlazne prekidačke funkcije sa velikim brojem ulaza i izlaza korišćenjem raznih vrsta binarnih dijagrama odlučivanja. Eksperimentalni rezultati korišćenjem benčmark funkcija potvrđuju efikasnost predloženih algoritama.

Summary

Development of Algorithms for Computation of the Autocorrelation of Switching Functions over Decision Diagrams

The autocorrelation is a mathematical operation with important applications in computer science and engineering. The space and time complexity of algorithms for computing the autocorrelation is exponential in the number of variables in the switching function. Most of existing algorithms focus on obtaining the autocorrelation coefficients of single-output switching function. However, in practical applications are usually required to work with multi-output switching functions. With this motivation, this doctoral thesis describes new algorithms for the efficient computation of the complete total autocorrelation for multiple-output switching functions with large number of inputs and outputs over various types of binary decision diagrams. Experimental results over benchmarks confirm the efficiency of the proposed algorithms.

Zahvalnica

Autor se zahvaljuje na pomoći Grupi za spektralne tehnike u okviru CIITLAB laboratorije Elektronskog fakulteta u Nišu, a posebno na korisnim savetima Prof. Dr. Radomira Stankovića sa Elektronskog fakulteta u Nišu i Prof. Dr. Claudia Morage sa Univerziteta u Dortmundu u Nemačkoj i Evropskog Centra za Soft Computing u Španiji.

The author is grateful to Spectral Technique group from CIITLAB lab of the Faculty of Electronic Engineering Niš, for the help, in particular to Prof. Dr. Radomir Stanković from the Faculty of Electronic Engineering Niš, and Prof. Dr. Claudio Moraga from the University of Dortmund, Germany and European Centre for Soft Computing, Spain, for useful advices.

U Nišu, decembra 2014.

Sadržaj

Spisak slika

Spisak tabela

Spisak algoritama

Skraćenice

ACMF	Autocorrelation Memory Function autokorelaciona memorijska funkcija
ATT	American Telephone and Telegraph Company američka kompanija za telefoniju i telegrafiju
BDD	Binary Decision Diagram binarni dijagram odlučivanja
BDT	Binary Decision Tree binarno stablo odlučivanja
BMD	Binary Moment Diagram dijagram binarnog momenta
C	Complement bit bit za komplementiranje
CAD	Computer Aided Design dizajn uz pomoć računara
CAL	California Kalifornija
CDMA	Code Division Multiple Access višestruki pristup sa kodovanom raspodelom kanala
CMU	Carnegie Mellon University Carnegie Mellon Univerzitet
CUDD	Colorado University Decision Diagram dijagrami odlučivanja na univerzitetu u Koloradu
DFT	Discrete Fourier Transform diskretna Fourierova transformacija

EVBDD	Edge Valued Binary Decision Diagram
	binarni dijagram odlučivanja sa težinskim granama
F	Flag bit
	bit za označavanje
FDD	Functional Decision Diagram
	funkcionalni dijagram odlučivanja
FFT	Fast Fourier Transform
	brza Fourierova transformacija
FWT	Fast Walsh Transform
	brza Walshova transformacija
HDD	Hybrid Decision Diagram
	hibridni dijagram odlučivanja
IBM	International Business Machines
	internacionalni poslovni računari
ITE	If Then Else
	ako, onda, u suprotnom
IWLS	International Workshop on Logic and Synthesis
	Internacionalni skup za logičku sintezu
KDD	Kronecker Decision Diagram
	kronekerov dijagram odlučivanja
KFDD	Kronecker Functional Decision Diagram
	Kronekerov funkcionalni dijagram odlučivanja
MCNC	Microelectronics Center of North Carolina
	Centar za mikroelektroniku u Severnoj Karolini
MDD	Multiple-place Decision Diagram
	višeznačni dijagram odlučivanja
MOS	Metal Oxide Semiconductor
	metal–oksid poluprovodnik
MTBDD	Multi–Terminal Binary Decision Diagram
	multi–terminalni binarni dijagram odlučivanja
MTBDT	Multi–Terminal Binary Decision Tree
	multi–terminalno binarno stablo odlučivanja

MV	Multiple Valued višeznačni
MVL	Multiple Valued Logic višeznačna logika
OBDD	Ordered Binary Decision Diagram uređeni binarni dijagram odlučivanja
PC	Personal Computer personalni računar
PLA	Programmable Logic Array programirljivo logičko polje
RC	Reference Counter brojač referenci
SBDD	Shared Binary Decision Diagram razdeljeni binarni dijagram odlučivanja
SIS	Simulation Interacting with Synthesis simulacija u interakciji sa sintezom
SMTBDD	Shared Multi–Terminal Binary Decision Diagram razdeljeni multi–terminalni binarni dijagram odlučivanja
TDD	Ternary Decision Diagram ternarni dijagram odlučivanja
TUD	Technische Universität Darmstadt Tehnički univerzitet u Darmštadu
UML	Unified Modeling Language objedinjeni jezik za modelovanje
USA	United States of America Sjedinjene američke države
VAR	Varibale promenljiva
XOR	Exlusive OR eksluzivno ili
ZBDD	Zero–suppressed Binary Decision Diagram binarni dijagram odlučivanja sa potiskivanjem nule

ZDD	Zero–suppressed Decision Diagram dijagram odlučivanja sa potiskivanjem nule

Poglavlje 1

Uvod

1.1 Opis problema i motivacija

Autokorelacija je matematička operacija kros–korelacije signala (funkcije) sa samim sobom [?]. Autokorelacija signala (funkcije) daje meru sličnosti između signala (funkcije) koji se razmatra i istog tog signala (funkcije) koji je modifikovan na neki način. Autokorelacija je matematička operacija sa značajnim primenama u oblasti računarske tehnike i inžinjerstva [?], [?], [?], [?], [?], [?].

U oblasti prekidačke teorije i logičkog projektovanja autokorelacija predstavlja jezgro mnogih optimizacionih algoritama, kao što su algoritmi za redukciju broja promenljivih u prekidačkim funkcijama [?], smanjenje broja čvorova [?], broja puteva [?] i prosečne dužine puteva u dijagramima odlučivanja [?] i funkcionalnu dekomoziciju [?].

Ostale primene autokorelacije odnose se na određivanje opsega detekcije radara, hardversko testiranje, projektovanje linearnih sistema i drugo. Binarne sekvence sa optimalnom autokorelacijom veoma su značajne u kriptografiji za generisanje kriptografskih ključeva. Kod CDMA (eng. CDMA – Code Division Multiple Access) komunikacionih sistema takve sekvence se primenjuju za određivanje ispravnosti primljenih informacija. Više detalja o ovim i sličnim primenama autokorelacije mogu se naći u [?], [?], [?] i [?]. Zbog toga, istraživanje osobina binarnih sekvenci sa optimalnom autokorelacijom kao i metoda za generisanje takvih sekvenci predstavljaju aktuelne istraživačke teme.

Međutim, u praktičnim primenama se najčešće zahteva efikasno izračunavanje autokoralcionih koeficijenata pri čemu se efikasnost odnosi kako na vremenske, tako i na raspoložive memorijske resurse. Kompleksnost problema efikasnog izračunavanja vezana je za dužinu binarnih sekvenci čija autokorelacija treba da se izračuna, što je posebno izraženo u

slučaju sistema sa ograničenim resursima, što je detaljno opisano u [**?**]. Efikasnost izračunavanja autokorelacionih koeficijenata je direktno uslovljena osobinama algoritama za izračunavanje autokorelacije i odgovarajućih struktura podataka. U slučaju izračunavanja autokorelacije prekidačkih funkcija, autokorelacioni koeficijenti se predstavljaju preko vektora dužine 2^n, gde je n broj promenljivih prekidačke funkcije. Zbog toga algoritmi za izračunavanje autokorelacionih koeficijenata imaju eksponencijalnu kompleksnost u odnosu na broj promenljivih funkcije. Ubrzanje postojećih algoritama zasnovano na poboljšanju korišćenog hardvera zbog njihove eksponencijalne složenosti ne može uvek da se koristi niti značajno doprinosi rešenju problema. Stoga je razvoj novih algoritama i izbor odgovarajućih struktura podataka za efikasno izračunavanje autokorelacije od suštinskog značaja za mnoga područija primene.

Problem izračunavanja autokorelacionih koeficijenata može biti rešen (do određene mere) ako se binarne sekvence posmatraju kao prekidačke funkcije a za njihovo predstavljanje koriste redukovane reprezentacije prekidačkih funkcija, kao što su binarni dijagrami odlučivanja (eng. BDD – Binary Decision Diagram), koji mogu da se koriste i kao struktura podataka za potrebna izračunavanja. Binarni dijagrami odlučivanja imaju široku primenu u različitim CAD (eng. Computer Aided Design) aplikacijama, posebno u oblastima simboličke simulacije i verifikacije kombinacione i sekvencijalne prekidačke logike [**?**]. Binarni dijagrami odlučivanja predstavljaju strukturu podataka pogodnu za predstavljanje prekidačkih funkcija sa velikim brojem promenljivih i dobijaju se primenom redukcije odgovarajućeg binarnog stabla odlučivanja. Redukcija se izvodi deljenjem izomorfnih podstabala i brisanjem redundantnih informacija u binarnom stablu odlučivanja, korišćenjem odgovarajućih redukcionih pravila. Višeizlazne prekidačke funkcije mogu da se predstave preko razdeljenih binarnih dijagrama odlučivanja, gde se za svaki izlaz kreira poseban dijagram odlučivanja. Razdeljeni binarni dijagram odlučivanja se dobijaju deljenjem izomorfnih podstabala između dijagrama svih izlaza funkcije.

Postoje različiti algoritmi za izračunavanje autokorelacionih koeficijenata prekidačkih funkcija za različite strukture podataka, uključujući i dijagrame odlučivanja. Postojeći algoritmi bazirani na binarnim dijagramima odlučivanja ograničeni su na izračunavanje pojedinačnih autokorelacionih koeficijenata za prekidačke funkcije sa jednim izlazom i pogodna samo za izračnavanja koeficijenata prvog reda [**?**], [**?**], [**?**]. Međutim, u praksi se vrlo često zahteva izračunavanje kompletnih autokorelacionih koeficijenata (autokorelacioni spektar) što zahteva da se isti algoritam ponavlja 2^n puta, gde je n broj promenljivih funkcije. Osim toga kod višeizlaznih prekidačkih funkcija se vrlo često zahteva izračunavanje totalnih autokorelacionih koeficijenata definisanih kao zbir autokorelacionih koeficijenata pojedinačnih izlaza. To zahteva da se isti algoritam izračunavanja ponavlja k puta, gde je k broj izlaza funkcije. Stoga razmatranje alternativnih pristupa

za izračunavanje autokorelacionih koeficijenata kao i primena različitih vrsta dijagrama odlučivanja jeste značajan i aktuelan problem.

1.2 Doprinos istraživanja u doktorskoj tezi

Algoritmi za izračunavanje pojedinačnih autokorelacionih koeficijenata višeizlaznih prekidačkih funkcija preko BDD-a zahtevaju višestruko ponavljanje izvršenja algoritma za svaki izlaz posebno. Iz tog razloga, u cilju povećanja efikasnosti izračunavanja, u ovoj tezi biće izvršena generalizacija metoda za izračunavanje pojedinačnih autokorelacionih koeficijenata preko BDD-a na nove algoritme preko razdeljenih BDD-a (eng. SBDD – Shared BDD) [**?**]. Generalizacija algoritama nije jednostavna jer se zahteva promena strukture podataka, kao i modifikacija algoritama za manipulaciju tim strukturama [**?**]. Kao rezultat istraživanja razvijena su dva nova algoritma za izračunavanje pojedinačnih autokorelacionih koeficijenata:

1. SBDD algoritam sa permutovanim labelama [**?**],
2. SBDD algoritam sa bočnim kretanjem [**?**].

Oba algoritma su implementirana i testirana na skupu standardnih benčmark funkcija. Kao dodatak, kod SBDD algoritma sa bočnim kretanjem, kod implementacije jezgra BDD paketa je uvedena i nova memorijska funkcija i heš tabela za efikasnije izračunavanje totalnih autokorelacionih koeficijenata prilikom obilaska SBDD-a.

Algoritmi za izračunavanje kompletne totalne autokorelacije višeizlaznih prekidačkih funkcija imaju još veću eksponencijalnu kompleksnost od algoritama za izračunavanje pojedinačnih autokorelacionih koeficijenata [**?**]. Oni zahtevaju višestruko ponavljanje izvršenja algoritma za svaki koeficijent posebno. Ovi algoritmi mogu biti zasnovani na postupcima izračunavanja autokorelacije na osnovu Wiener-Khinchin-ove teoreme i brze Fourier-ove transformacije na odgovarajućim algebarskim strukturama [**?**]. Efikasnost izračunanja kompletne totalne autokorelacije se može povećati korišćenjem razdeljenih multi-terminalnih binarnih dijagrama odlučivanja (eng. SMTBDD – Shared Multi-Terminal BDD). Ova izračunavanja u slučaju prekidačkih funkcija sa velikim brojem promenljivih zahtevaju pamćenje i rad sa velikim celobrojnim vrednostima. Zbog toga se standardni BDD paketi sa celobrojnim terminalnim čvorovima ne mogu koristiti, tako da je bilo neophodno razviti specijalizovani BDD paket sa dinamičkim terminalnim čvorovima što predstavlja proširenje standardnih BDD tehnika [**?**], [**?**], [**?**], [**?**], [**?**]. Kao rezultat istraživanja razvijena su dva nova algoritma za izračunavanje kompletne autokorelacije:

1. SMTBDD algoritam preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije [**?**], [**?**],
2. SMTBDD algoritam preko BDD paketa sa dinamičkim terminalnim čvorovima [**?**], [**?**].

Oba algoritma su takođe implementirana i testirana na skupu standardnih benčmark funkcija.

1.3 Organizacija doktorske teze

Ova doktorska disertacija je organizovana na sledeći način. U poglavlju 2 je izložena, diskutovana i dopunjena neophodna teorijska osnova u skladu sa razmatranim problemom. U poglavlju 3 su predstavljene definicije, osobine i primene autokorelacije prekidačkih funkcija. Osim toga, izloženi su i analizirani postojeći algoritmi za izračunavanje autokorelacionih koeficijenata (algoritam iscrpljivanja, algoritam preko disjunktnih kubova i Wiener-Khinchin algoritam preko Walsh-ovih koeficijenata). Poglavlje 4 opisuje postojeće algoritme bazirane na dijagramima odlučivanja kao i predložene generalizovane algoritme za izračunavanje pojedinačnih i kompletnih autokorelacijonih koeficijenata višeizlaznih prekidačkih funkcija korišćenjem raznih vrsta dijagrama odlučivanja. Poglavlje 5 opisuje osnovne principe koji se koriste kod implementacije standardnih BDD paketa i implementacije predloženih algoritama za izračunavanje autokorelacije korišćenjem raznih vrsta dijagrama odlučivanja u okviru BDD paketa. Poglavlje 6 ilustruje, na osnovu skupa benčmark funkcija, efikasnost razvijenih algoritama za izračunavanje pojedinačnih i kompletnih autokorelacijonih koeficijenata prekidačkih funkcija. Osim toga prikazane su i neke osobine izračunavanja. Na kraju, u poglavlju 7, izneti su zaključci i diskusija mogućih pravaca budućih istraživanja.

Poglavlje 2

Predstavljanje prekidačkih funkcija

Ovo poglavlje opisuje teorijsku osnovu za teme prezentovane u ovoj disertaciji. Posebno je opisan i definisan pojam prekidačkih funkcija. Potom su prezentovani tradicionalni načini za predstavljanje prekidačkih funkcija preko analitičkih reprezentacija i kubova. Osim toga, u ovom poglavlju su opisani i netradicionalni načini za predstavljanje prekidačkih funkcija preko različitih vrsta binarnih dijagrama odlučivanja. Ovo poglavlje takođe prezentuje drugi domen za opisivanje prekidačkih funkcija korišćenjem spektralnih (Walsh-ovih) koeficijenata.

2.1 Prekidačke funkcije

Funkcije koje opisuju prekidačke ili logičke sisteme se nazivaju prekidačkim ili Boole-ovim funkcijama. Ulazi i izlazi ovih funkcija su ograničeni na Boole-ov domen od samo dve vrednosti: 0 i 1. Prekidačke funkcije se mogu proširiti tako da dozvoljavaju više od dve vrednosti na ulazima ili izlazima funkcija. Ovaj tip prekidačkih funkcija se naziva višeznačnim logičkim funkcijama (eng. MVL – Multiple-valed Logic Function). Ova disertacija se, međutim koncentriše samo na prekidačke funkcije definisane u Boole-ovom domenu $\{0, 1\}$.

Prekidačka funkcija je funkcija $f(x)$ gde je $f(x) \in 0, 1$, $x = \{x_1, x_2, \ldots, x_{n-1}, x_n\}$ i $x_i \in \{0, 1\}, \quad \forall i \in \{1, 2, \ldots, n\}$.

Ulazni vektor za prekidačku funkciju može da ima 2^n mogućih vrednosti. Ako se prihvati da ovi vektori predstavljaju binarne reprezentacije ekvivaletne celobrojnoj vrednosti k:

$$k = \sum_{i=1}^{n} x_i 2^{n-i}, \tag{2.1}$$

onda ulazni vektori mogu da imaju sve moguće vrednosti u opsegu od 0 do 2^{n-1}. Najjednostavniji način da se ovo ilustruje je da se prekidačka funkcija prikaže preko tablice istinitosti. Na primer tablica istinitosti za prekidačku funkciju zadatu sa $f(x_1, x_2) = \overline{x}_1 \vee x_2$ prikazana je na slici 2.1.

k	x_1	x_2	$f(x)$
0	0	0	1
1	0	1	1
2	1	0	0
3	1	1	1

SLIKA 2.1: Tablica istinitosti za prekidačku funkciju zadatu sa $f(x) = \overline{x}_1 \vee x_2$.

Operatori $\wedge$ - logičko "i" i $\vee$ - logičko "ili" se u oblasti prekidačke teorije obično označavaju sa "$\cdot$" i "+", respektivno. Vektor istinitosti za ovu prekidačku funkciju je $F = [1101]^T$. Vektor istinitosti (izlazi) prekidačke funkcije n promenljivih sadrži 2^n binarnih vrednosti.

Treba napomenuti da u nekim slučajevima za vrednosti ulaza i izlaza prekidačke funkcije koje su ograničene na Boole-ov domen $\{0, 1\}$ mogu da se koriste i vrednosti $\{+1, -1\}$.

Prekidačka funkcija prikazana u prethodnom primeru je poznata kao potpuno definisana prekidačka funkcija, gde su svi izlazi funkcije definisani za sve 2^n moguće ulazne kombinacije. Takođe su poznate i nepotpuno definisane prekidačke funkcije. To su prekidačke funkcije kod kojih vrednosti izlaza za neke ulazne kombinacije nisu definisane. Ovi izlazi se nazivaju "don't care" i označavaju se crticom "-".

Prekidačka funkcija prikazana u prethodnom primeru je poznata kao jednoizlazna prekidačka funkcija. Takođe su poznate i višeizlazne prekidačke funkcije. Na primer tablica istinitosti za višeizlaznu prekidačku funkciju sa dva izlaza kod koje je jedan izlaz nepotpuno definisana funkcija prikazana je na slici 2.2.

x_1	x_2	x_3	$f_1(x)$	$f_2(x)$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	-
0	1	1	1	1
1	0	0	0	0
1	0	1	0	-
1	1	0	1	-
1	1	1	1	1

SLIKA 2.2: Primer tablice istinitosti za nepotpuno definisanu višeizlaznu prekidačku funkciju sa dva izlaza.

2.2 Predstavljanje prekidačkih funkcija kubovima

Prekidačke funkcije, kao što je prikazano u prethodnom poglavlju, se obično definišu tablicama istinitosti koje na levoj strani sadrže sve moguće kombinacije ulaza, a na desnoj vrednosti funkcije (vektor istinitosti). Veličina tablice istinitosti kao i vektor istinitosti eksponencijalno rastu sa povećanjem broja promenljivih funkcije. Redukovane reprezentacije prekidačkih funkcija možemo dobiti korišćenjem njihovih specifičnih osobina. Pošto prekidačke funkcije mogu da uzmu samo dve različite vrednosti, nije neophodno da se pamti kompletna tablica istinitosti niti ceo vektor istinitosti. Dovoljno je označiti mesta gde data funkcija uzima vrednosti 0 ili 1 i pretpostaviti da u ostalim tačkama domena, funkcija ima drugu vrednost 1 ili 0, respektivno. Na ovaj način funkcija se predstavlja preko 0-polja ili 1-polja. U slučaju nepotpuno definisanih funkcija uvodi se i *-polje koje sadrži vektore nezavisno promenljivih na kojima funkcija nije definisana.

Polja se sažeto mogu predstaviti pomoću kubova. Kod kubova su neki elementi označeni sa "-" tj. sa "don't care" vrednostima ulaza, pri čemu se podrazumeva da "-" može biti jednako 0 ili 1, tako da je sa jednim kubom pokriveno više mogućih kobinacija ulaza. Na primer, na slici 2.3 prikazano je predstavljanje prekidačke funkcije f_1 iz primera sa slike 2.2 preko polja i preko kubova.

$1-polje$			$1-kubovi$		
x_1	x_2	x_3	x_1	x_2	x_3
0	0	0	0	0	-
0	0	1	0	-	1
0	1	1	1	1	0
1	1	0			

SLIKA 2.3: Primer predstavljanja prekidačke funkcije f_1 iz primera sa slike 2.2 preko polja i preko kubova.

Za dva kuba se kaže da su disjunktna [**?**] ako je presek skupova njihovih minterma prazan skup, gde se skup minterma kuba definiše kao skup svih ulaznih kombinacija vrednosti promenljivih koje taj kub pokriva.

Na primer, razmotrimo dva kuba "$01--$" i "$-0-1$". Skupovi njihovih minterma su $0100, 0101, 0110, 0111$ i $0001, 0011, 1001, 1011$ respektivno, gde se može zaključiti da je njihov presek prazan skup i na osnovu toga da su kubovi "$01--$" i "$-0-1$" disjunktni.

2.3 Dijagrami odlučivanja

U prethodnim poglavljima je opisno predstavljanje prekidačkih funkcija preko tablica istinitosti, vektora istinitosti i preko kubova. Međutim, način predstavljanja funkcija

preko tablica istinitosti je neefikasan jer one sadrže sve moguće kombinacije ulaza i vrednosti na tim ulazima. Slično važi i kod predstavljanja funkcije preko vektora istinitosti. Jedan način redukovane reprezentacije funkcije je predstavljanje preko kubova. U poslednje vreme u cilju rešavanja problema efikasnog predstavljanja sve više se koriste redukovane reprezentacije funkcija preko raznih vrsta dijagrama odlučivanja. Dijagrami odlučivanja se dobijaju redukcijom odgovarajućeg drveta odlučivanja radi kompaktnijeg predstavljanja.

2.3.1 Binarni dijagrami odlučivanja

Binarni dijagrami odlučivanja (BDD) predstavljaju strukturu podataka pogodnu za predstavljanje prekidačkih funkcija sa mnogo ulaznih promenljivih [**?**], [**?**]. Zahvaljujući tome, BDD su počeli da se masovno koriste kod različitih CAD aplikacija, uključujući simboličku simulaciju i verifikaciju kombinacionih i sekvencialnih prekidačkih mreža [**?**], [**?**], [**?**], [**?**].

Binarni dijagram odlučivanja se dobija redukcijom binarnog stabla odlučivanja (eng. BDT - Binary Decision Tree) [**?**], [**?**]. Redukcija se obavlja deljenjem izomorfnih podstabala i brisanjem redundantnih informacija iz BDT-a korišćenjem odgovarajuće definisanih redukcionih pravila. BDT može da se posmatra kao grafička reprezentacija rekurzivne primene Shannon-ove dekompozicije po svim promenljivama prekidačke funkcije [**?**]. Shannon-ova dekopozicija prekidačke funkcije po promenljivoj x_i je definisana kao:

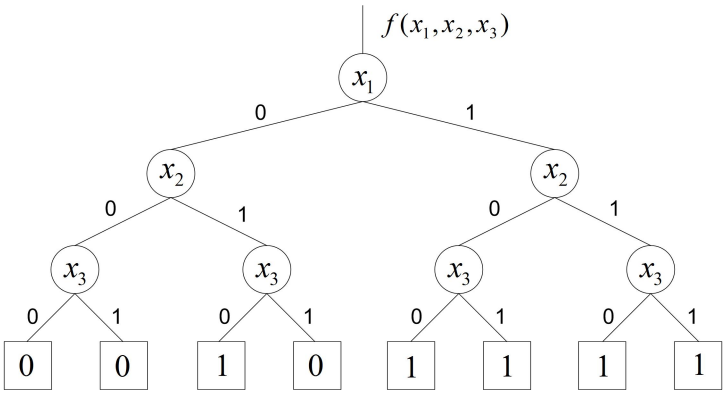
$$f(x_1, x_2, ..., x_i, ...) = \overline{x}_i f_0 * x_i f_1 \tag{2.2}$$

gde je f_0 vrednost funkcije f kada je $x_i = 0$ i f_1 vrednost funkcije f kada je $x_i = 1$. Ograničenja za kreiranje redukovanog uređenog BDD-a (eng. OBDD - Ordered BDD) su naknadno definisana u [**?**].

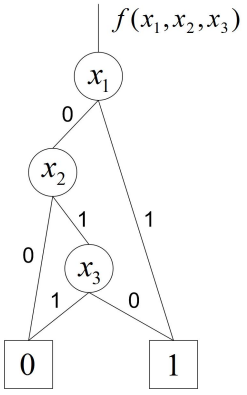
Primeri BDT-a i BDD-a za prekidačku funkciju definisanu sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ čiji je vektor istinitosti $F = [00101111]^T$ prikazani su na slikama 2.4 i 2.5. Očigledno je da BDD ima dosta manji broj čvorova u odnosu na BDT, s obzirom da u vektoru istinitosti funkcije f postoje konstantni podvektori nula dužine 2 i jedinica dužine 4. Ova osobina je esencijalna kod BDD reprezentacije prekidačkih funkcija i najviše se eksploatiše kod raznih izračunavanja preko BDD-a.

2.3.2 Razdeljeni binarni dijagrami odlučivanja

Višeizlazne prekidačke funkcije se predstavljaju preko razdeljenih BDD-ova (SBDD-ova) [**?**] gde postoji više polaznih čvorova (korena dijagrama) za svaki izlaz prekidačke



SLIKA 2.4: BDT za prekidačku funkciju čiji je vektor istinitosti $F = [00101111]^T$



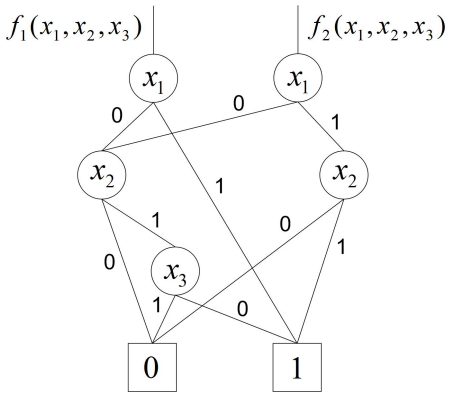
SLIKA 2.5: BDD za prekidačku funkciju čiji je vektor istinitosti $F = [00101111]^T$

funkcije posebno. Zbog toga se SBDD dobija deljenjem izomorfnih podstabala u BDD-ovima preko kojih su predstavljeni izlazi prekidačke funkcije, gde se izlazi tretiraju kao posebne prekidačke funkcije.

Primer SBDD-a za višeizlaznu prekidačku funkciju čiji su izlazi definisani sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$ prikazan je na slici 2.6. Vektor istinitosti funkcije f_1 je $F_1 = [00101111]^T$, a funkcije f_2 je $F_2 = [00100011]^T$. U prethodnom primeru SBDD sa dva polazna čvora može se iskoristiti za predstavljanje funkcija f_1 i f_2. Očigledno je da SBDD ima znatno manji broj čvorova u odnosu na dva odvojena BDD-a za funkcije f_1 i f_2, s obzirom da postoje deljivi podvektori u vektorima istinitosti funkcija f_1 i f_2.

2.3.3 Veza između kubova i dijagrama odlučivanja

Kod BDT-a svaki put u dijagramu od polaznog do krajnjeg nenultog čvora odgovara određenom članu sa potpunim proizvodima u potpunoj disjunktivnoj normalnoj formi prekidačke funkcije. Slično kod BDD-a svaki put u dijagramu odgovara određenom članu sa proizvodima u redukovanoj disjunktivnoj normalnoj formi.



SLIKA 2.6: SBDD za višeizlaznu prekidačku funkciju čiji su vektori istinitosti izlaza $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$

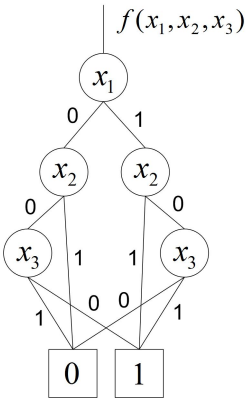
S obzirom da BDD predstavlja grafičku reprezentaciju disjunktivne normalne forme za prekidačku funkciju, određivanje disjunktivne forme iz BDD-a je krajnje jednostavno. Odgovarajuće metode za određivanje disjunktivne forme iz BDD-a su diskutovane u [**?**], [**?**]. Skup disjunktnih kubova za datu prekidačku funkciju se može odrediti obilaskom svih puteva u BDD-u date funkcije što je opisano sledećim algoritmom [**?**]:

1. Kreirati BDD za funkciju f i izvršiti obilazak svih puteva iz polaznog čvora do terminalnog čvora 1.
2. Izvršiti pamćenje labela na granama za svaki put kao sekvencu $(\tilde{x_i}, \tilde{x_j}, \ldots, \tilde{x_k})$, gde $\tilde{x} \in \{x, \bar{x}\}$ i $\bar{x}$ označava logički komplement promenljive x.
3. Proširiti sekvence tako da odgovaraju putevima dužine n, gde je n broj promenljivih funkcije, dodavanjem "-" oznaka, odnosno "don't care" vrednostima, na pozicije nepostojećih promenljivih.
4. Izvršiti zamenu u sekvencama promenljivih x i $\tilde{x}$ sa 0 i 1, respektivno, što generiše traženu listu disjunktnih kubova.

Primer BDD-a za prekidačku funkciju $f(x_1, x_2, x_3)$ definisanu sa $F = [10000111]^T$ prikazan je na slici 2.7. Primenom prethodno definisanog algoritma na ovaj BDD vrši se generisanje puteva u BDD-u određenih sledećim sekvencama $(\overline{x}_1, \overline{x}_2, \overline{x}_3), (x_1, \overline{x}_2, x_3), (x_1, x_2)$. Proširenjem ovih puteva u BDD-u dodavanjem "-" oznaka dobijaju se sledeće sekvence: $(\overline{x}_1, \overline{x}_2, \overline{x}_3), (x_1, \overline{x}_2, x_3), (x_1, x_2, -)$. Zamenom ovih sekvenci promenljivih x i $\tilde{x}$ sa 0 i 1, respektivno, vrši se generisanje sledeće liste disjunktnih kubova $000, 101, 11-$.

2.3.4 Multi-terminalni binarni dijagrami odlučivanja

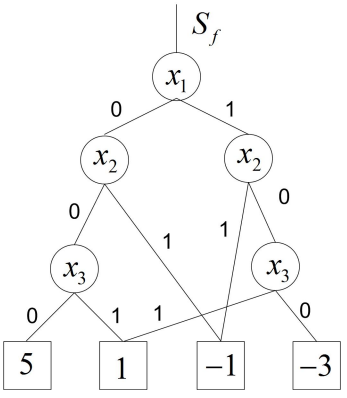
BDD tehnika može biti proširena za predstavljanje i celobrojnih funkcija. Ova tehnika se koristi u mnogim oblastima računarstva. MTBDD (Multi-terminalni BDD) [**?**],



SLIKA 2.7: Primer BDD-a za prekidačku funkciju $f(x_1, x_2, x_3)$ definisanu sa $F = [10000111]^T$

[**?**] predstavlja generalizaciju BDD-a dozvoljavajući postojanje terminalnih čvorova sa više različitih celobrojnih vrednosti. MTBDD se dobija redukcijom odgovarajućeg multi-terminalnog binarnog stabla odlučivanja (eng. MTBDT - Multi-terminal Binary Decision Tree). Slično kao kod BDD-a, kod $MTBDT(f)$ nivoi se sastoje od čvorova koji odgovaraju promenljivama u celobrojnoj funkciji $f(x_1, x_2, ..., x_n)$. Usvojeno je da se nivoima kod MTBDT-a pridružuju promenljive funkcije na isti način kao kod BDT-a ili BDD-a. Ista konvencija važi i za MTBDD. Kompletan skup aritmetičkih operacija može biti efikasno realizovan nad MTBDD-om, kao što su sabiranje, oduzimanje i množenje. Slično važi i za logičke operacije koje se implementiraju rekurzivnim algoritmima koji imaju linearno vreme izvršavanja u odnosu na veličinu dijagrama [**?**].

Walsh-ov spektar prekidačke funkcije (ako se faktor za skaliranje 2^{-n} izostavi) je jedan vektor sa celobrojnim vrednostima i može biti predstavljen preko MTBDD-a. Detaljniji opis definisanja Walsh-ovog spektra može se naći u sledećem poglavlju. Primer MTBDD-a za Walsh-ov spektar $S_f = [5, 1, -1, -1, -3, 1, -1, -1]^T$ prekidačke funkcije definisane sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$, čiji je vektor istinitosti $F = [00101111]^T$ je prikazan je na slici 2.8.

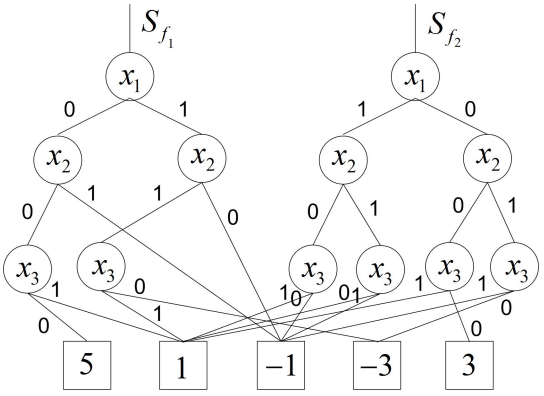


SLIKA 2.8: Primer MTBDD-a za Walsh-ov spektar $S_f = [5, 1, -1, -1, -3, 1, -1, -1]^T$ prekidačke funkcije definisane sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$

Evidentno je da je predstavljanje Walsh-ovog spektra iz primera preko MTBDD-a kompaktno, sobzorm da u vektoru Walsh-ovih koeficijenata postoje dva konstantna podvektora sa vrednostima -1 dužine 2. Ova osobina je esencijalna kod MTBDD reprezentacije prekidačkih funkcija i najviše se eksploatiše kod raznih izračunavanja preko MTBDD-a.

2.3.5 Razdeljeni multi-terminalni binarni dijagrami odlučivanja

Višeizlazne celobrojne funkcije mogu da se predstave preko razdeljenog MTBDD-a (SMTBDD-a) gde postoji više polaznih čvorova (korena dijagrama) za svaki izlaz celobrojne funkcije posebno. Primer SMTBDD-a za Walsh-ove spektre
$S_{f_1} = [5, 1, -1, -1, -3, 1, -1, -1]^T$ i $S_{f_2} = [3, 1, -3, -1, -1, 1, -1, -1]^T$ višeizlazne prekidačke funkcije definisane sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$ čiji su vektori istinitosti $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$ prikazan je na slici 2.9. Očigledno je da SMTBDD ima znatno manji broj čvorova u odnosu na dva odvojena MTBDD-a za funkcije f_1 i f_2, s obzirom da postoje deljivi podvektori u Walsh-ovim spektrima S_{f_1} i S_{f_2} funkcija f_1 i f_2.



Slika 2.9: Primer SMTBDD-a za Walsh-ove spektre $S_{f_1} = [5, 1, -1, -1, -3, 1, -1, -1]^T$ i $S_{f_2} = [3, 1, -3, -1, -1, 1, -1, -1]^T$ višeizlazne prekidačke funkcije definisane sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$

2.4 Walsh-ovi koeficijenti prekidačkih funkcija

Walsh-ovi koeficijenti se definišu preko diskretnih Walsh-ovih funkcija koje mogu da se posmatraju kao određeni primeri Walsh-ovih funkcija. Diskretne Walsh-ove funkcije reda n označene sa $wal(k, x), x, k \in \{0, 1, ..., 2^n\}$ su definisane kao kolone Walsh-ove matrice reda $2^n \times 2^n$ [**?**]:

$$W(n) = \bigotimes_{i=1}^{n} W(1), \tag{2.3}$$

gde $W(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ predstavlja bazičnu Walsh-ovu matricu prvog reda.

Primer Walsh-ove matrice za $n = 3$ je:

$$W(3) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \\ = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{2.4}$$

Iz strukture Kronecker-ovog proizvoda može da se dobije sledeća rekurentna relacija za Walsh-ovu matricu reda n:

$$W(n) = \begin{bmatrix} W(n-1) & W(n-1) \\ W(n-1) & -W(n-1) \end{bmatrix} \tag{2.5}$$

Za prekidačku funkciju f predstavljenu preko vektora istinitosti $F = [f(0), f(1), ..., f(2^n - 1)]^T$, Walsh-ov spektar u matričnoj notaciji, $S_f = [S_f(0), S_f(1), ..., S_f(2^n - 1)]^T$ je definisan kao:

$$S_f = W(n)F. \tag{2.6}$$

gde se $S_f(i), i = 0, ..., 2^n - 1$ nazivaju Walsh-ovim koeficijentima.

Kada se Walsh-ova transformacija primeni nad prekidačkom funkcijom podrazumeva se da se logičke vrednosti 0 i 1 posmatraju kao celobrojne vrednosti 0 i 1.

Primer Walshovog spektra za prekidačku funkciju f predstavljenu preko vektora istinitosti $F = [0, 0, 1, 0, 1, 1, 1, 1]^T$ je $S_f = [5, 1, -1, -1, -3, 1, -1, -1]^T$

Kod predstavljanja prekidačke funkcije preko Walsh-ovog spektra, neka poboljšanja mogu biti postignuta ako se za vrednosti ulaza prekidačke funkcije koristi $\{+1, -1\}$ kodiranje, što čini Walshovu transformaciju usaglašenijom [**?**]. Kod ovakvog načina kodiranja prekidačke funkcije nakon primene Walsh-ove transformacije dobijaju se Walsh-ovi koeficijenti čije su vrensoti parni brojevi, u opsegu -2^n do 2^n i čija je suma svih koeficijenata jednaka 2^n. Osim toga postoje ograničenja za kombinacije parnih celih brojeva koji mogu da se jave u Walsh-ovom spektru prekidačke funkcije. Na primer, za prekidačku funkciju

od $n = 3$ promenljivih, Walsh-ovi spektralni koeficijenti mogu da uzimaju vrednosti iz tri usaglašena skupa celobrojnih vrednosti $\{0, 8\}$, $\{0, 4, -4\}$ i $\{2, -2, 6\}$. Međutim, ne može se proizvoljno generisati Walsh-ov spektar iz jednog skupa celobrojnih vrdnosti. Na primer, Walsh-ov spektar $\frac{1}{8}[2, 2, -2, -2, 6, 2, 2, -2]^T$ ima zbir Walsh-ovih koeficijenata jednak $2^3 = 8$ i sve vredniosti koeficijenata pripadaju skupu $\{2, -2, 6\}$, ali spektar ne odgovara ni jednoj prekidačkoj funkciji. Walsh-ov spektar prekidačke funkcije koja koristi $\{+1, -1\}$ kodiranje se obeležava sa R_f.

Primer Walshovog spektra za prekidačku funkciju f koja koristi $\{+1, -1\}$ kodiranje predstavljenu preko vektora istinitosti $F = [0, 0, 1, 0, 1, 1, 1, 1]^T$ je $R_f = [2, -2, 2, 2, 6, -2, 2, 2]^T$.

2.4.1 Izračunavanje Walsh-ovih koeficijenata preko brze Walsh-ove transformacije

Brza Fourier-ova transformacija (eng. FFT - Fast Fourier Transform) je algoritam za efikasno izračunavanje diskretne Fourier-ove transformacije (eng. DFT - Discrete Fourier Transform), pri čemu se efikasnost odnosi kako na vreme izračunavanja, tako i na raspoložive memorijske resurse koji se koriste u procesu izračunavanja [**?**]. Proširenje FFT algoritma na Walsh-ovu transformaciju je jednostavno ako se Walsh-ova transformacija posmatra kao Fourier-ova transformacija na dijadičkoj grupi, tj. grupi binarnih sekvenci nad operacijom "suma po modulu 2" (eng. EXOR - Exlusive OR). U primenama vezanim za dijagrame odlučivanja koristi se Walsh-ova transformacija u Hadamard-ovom uređenju [**?**], [**?**]. Ovo uređenje se generiše Kronecker-ovim proizvodom osnovnih Walsh-ovih transformacionih matrica $W(1)$, što se može videti iz jednačine 2.3. Zbog toga se brza Walsh-ova transformacija (eng. FWT - Fast Walsh Transform) može izvesti iz takozvane Good-Thomas-ove faktorizacione teoreme za Walsh-ove matrice [**?**]. Ova teorema definiše osobinu da matrica koja se dobija Kronecker-ov proizvodom matrica $M(i), i = 1, ..., n$ može biti izražena običnim proizvodom retkoposednutih matrica $C(i), i = 1, ..., n$. Svaka retkoposednuta matrica $C(i)$ se ponovo predstavlja kao Kronecker-ov proizvod Walsh-ove matrice i jediničnih matrica, što predstavlja osnovu za definisanje "leptir" operacija u odgovarajućem koraku FWT algoritma.

Izračunavanje brze Walsh-ove transformacije koje se bazira na njenoj definiciji (videti jednačinu 2.3) je neefikasno. Razlog tome je eksponencijalna kompleksnost izračunavanja koja se može opisati sa $O(2^{2n})$, gde n predstavlja proj promenljivih prekidačke funkcije. Postojanje FWT algoritma poboljšava kompleksnost izračunavanja koja u ovom slučaju može opisati sa $O(2^n \log 2^n)$ [**?**]. Brza Walsh-ova transformacija se definiše sledećom faktorizacijom:

$$W(n) = \prod_{i=1}^{n} C_{w_i}(n) \tag{2.7}$$

gde je,

$$C_{w_i}(n) = \bigotimes_{j=1}^{n} C_{w_i}^{j}(1), \quad C_{w_i}^{j}(1) = \begin{cases} W(1), & i = j, \\ I(1), & i \neq j. \end{cases} \tag{2.8}$$

Matrica $C_{w_i}(n)$ definiše parcijalnu Walsh-ovu transformaciju i odgovara i-tom koraku FWT algoritma. Iz ove parcijalne faktorizacije se izvodi FFT algoritam Cooley-Tukey tipa [?].

Na primer, za prekidačku funkciju od dve promenljive zadatu preko vektora istinitosti $F = [1101]^T$, izračunavanje odgovarajućeg Walsh-ovog spektra zahteva 4 operacija sabiranja i 4 operacija oduzimanja, umesto 16 operacija množenja i 12 operacija sabiranja koliko je potrebno kod izračunavanja Walsh-ovog spektra preko definicije (jednačina 2.6). FWT algoritam za primer prekidačke funkcije se izvodi iz sledeće faktorizacije matrice W(2):

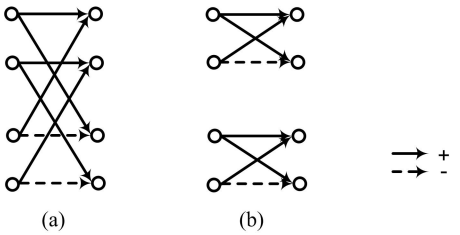
$$S_f = W(2)F = (C_1 C_2)F. \tag{2.9}$$

gde je

$$C_1 = W(1) \otimes I(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{2.10}$$

$$C_2 = I(1) \otimes W(1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{2.11}$$

Izračunavanja pomoću matrica C_1 i C_2 mogu biti izvršavana preko grafova toka operacija koji su prikazani na sledećoj slici. Na slici puna i isprekidana linija označavju operacije sabiranja i oduzimanja respektivno. Za prekidačku funkciju zadatu sa $F = [1101]^T$,

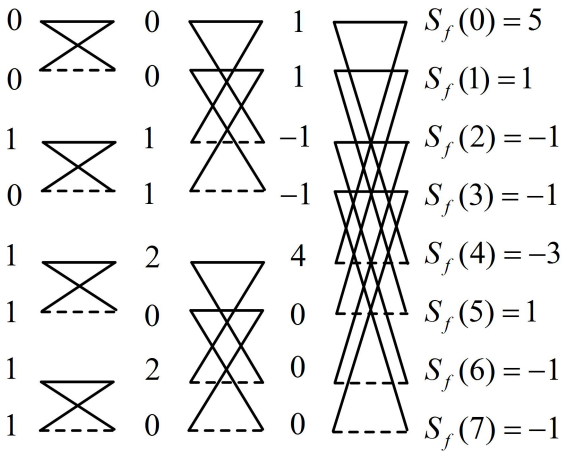


Slika 2.10: Graf toka operacija za izračunavanje pomoću matrica C_1 i C_2 u FWT algoritmu za prekidačku funkciju od dve promenljive: (a) tok operacija za matricu C_1, (b) tok operacija za matricu C_2

izračunavanje odgovarajućeg Walsh-ovog spektra prikazano je u sledećoj jednačini:

$$F = \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 2 \\ 2 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 3 \\ -1 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} S_f(0) \\ S_f(1) \\ S_f(2) \\ S_f(3) \end{bmatrix} = S_f \tag{2.12}$$

U praksi se u grafu toka operacija za izračunavanja u FWT algoritmu ne iscrtavaju strelice i kružići, već samo odgovarajuće linije za označavanje potrebnih operacija. Brzi algoritam za izračunavanje Walsh-ovog spektra prekidačke funkcije definisane sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ čiji je vektor istinitosti $F = [00101111]^T$ je prikazan je na slici 2.11. Očigledno



Slika 2.11: Brzi algoritam za izračunavanje Walsh-ovog spektra prekidačke funkcije definisane sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ čiji je vektor istinitosti $F = [00101111]^T$

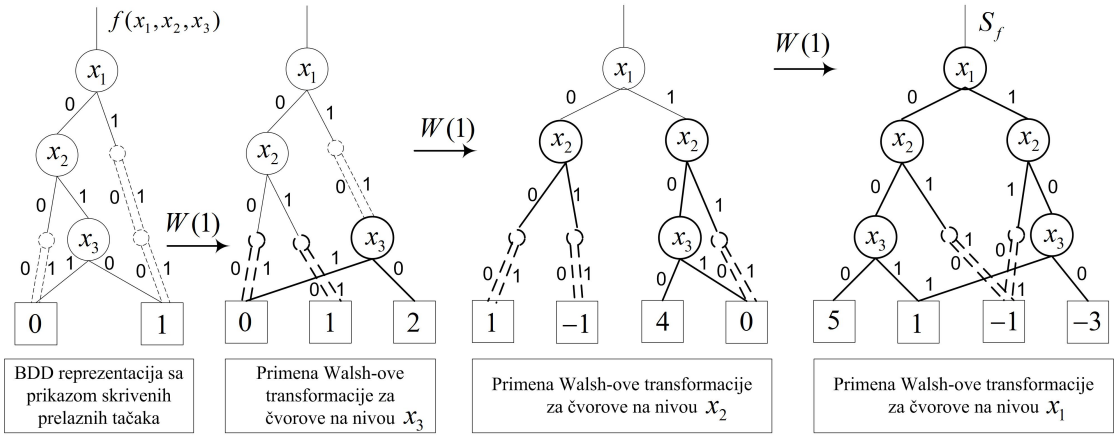
je da se izračunavanje Walsh-ovog spektra prekidačke funkcije $f(x_1, x_2, x_3)$ sastoji od mnogobrojnih ponavljanja primene iste "leptir"operacije, definisane na osnovu osnovne Walsh-ove transformacione matrice $W(1)$, koja se izvršava u svakom koraku FWT algoritma nad različitim podskupom podataka. Ova osobina je esencijalna kod predstavljanja funkcije preko MTBDD-a i najviše se eksploatiše kod izračunavanja Walshovog spektra preko brze Walsh-ove transformacije preko MTBDD-a. Više detalja o sličnim izračunavanjima preko algoritama FFT tipa mogu se naći u [**?**], [**?**].

2.4.2 Izračunavanje Walsh-ovih koeficijenata preko brze Walsh-ove transformacije i dijagrama odlučivanja

Algoritam izračunavanja brze Walsh-ove transformacije preko SMTBDD-a se bazira na brzom algoritmu za spektralne transformacije preko BDD-a. Nekoliko varijanti BDD-baziranih algoritma za izračunavanje Walsh-ove transformacije je razmatrano u brojnoj

literaturi [?], [?], [?], [?], [?], [?], [?]. U toj literaturi su takođe predstavljeni algoritmi izračunavanja preko BDD-a i za druge spektralne transformacije.

Ovaj algoritam se zasniva na faktorizaciji transformacionih matrica koje se koriste kod izvođenja FFT algoritma. "Leptir" operacije u algoritmu brze Walshove transformacije se implementiraju kao operacije sabiranja i oduzimanja u BDD-u koje koriste tehnike manipulacije grafom. Ovaj algoritam korisiti prednosti kompaknosti predstavljanja informacija preko MTBDD-a i može biti efikasniji za izračunavanje spektralnih transformacija prekidačkih funkcija u odnosu na tradicionalne načine izračunavanja. Na primer, detaljan prikaz brzog algoritma za izračunavanje Walsh-ovog spektra preko MTBDD-a prekidačke funkcije zadate sa $f(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$, čiji je vektor istinitosti $F = [00101111]^T$, je prikazan na slici 2.12.

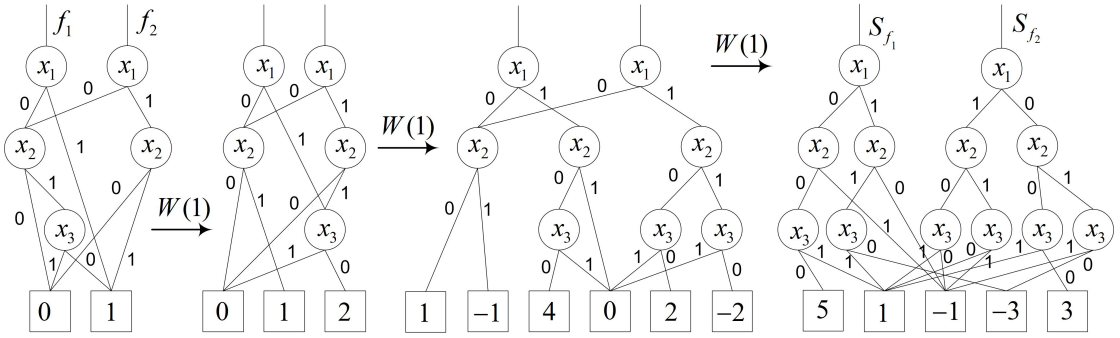


SLIKA 2.12: Primer brzog algoritma za izračunavanje Walsh-ovog spektra preko MTBDD-a prekidačke funkcije zadate sa $f(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$, čiji je vektor istinitosti $F = [00101111]^T$.

U ovom detaljnom prikazu primera brzog algoritma svi "skriveni" čvorovi ili prelazne tačke u MTBDD-u moraju da budu razmatrane kao postojeće kako bi bilo moguće lokalno primeniti Walsh-ovu transformaciju. Algoritam Walsh-ove transfomacije je "odozdo-nagore" tipa. Walah-ova transformacija koja se primenjuje lokalno na nivou u MTBDD-u koji odgovara promenljivoj x_3 ima efekat da se svaki čvor ili "skrivena" prelazna tačka zamenjuje poddijagramom gde "0" grana predstavlja sumu vrednosti oba poddijagrama tog čvora í "1" grana predstavlja razliku vrednosti oba poddijagrama tog čvora. Ista procedura se primenjuje korak po korak za sve čvorove i "skrivene" prelazne tačke na višim nivoima MTBDD-a.

Ova tehnika izračunavanja preko MTBDD-a može biti korišćena kod izračunavanja bilo koje spektralne transformacije koja se bazira na Kronecker-ovom proizvodu transformacionih matrica i može biti proširena i na razdeljene MTBDD-ove [?] za višeizlazne prekidačke funkcije. Na primer, brzi algoritam za izračunavanje Walsh-ovog spektra preko SMTBDD-a višeizlazne prekidačke funkcije definisane sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$

i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$, čiji su vektori istinitosti $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$, respektivno, je prikazan je na slici 2.13.



SLIKA 2.13: Primer brzog algoritma za izračunavanje Walsh-ovog spektra preko SMTBDD-a višeizlazne prekidačke funkcije definisane sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$, čiji su vektori istinitosti $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$.

Evidentno je da je izračunavanje i predstavljanje Walsh-ovog spektra iz prethodnog primera preko SMTBDD kompaktno, sobzorm da u vektorima međurezultata i Walsh-ovih koeficijenata postoje deljivi podvektori. Ova osobina je esencijalna kod efikasnog predstavljanja prekidačkih funkcija preko SMTBDD-a, kao i kod raznih izračunavanja preko SMTBDD-a. Očigledno je da je broj "leptir"operacija u SMTBDD-u mnogo manji u odnosu na broj "leptir" operacija u dva odvojena MTBDD-a za višeizlaznu prekidačku funkciju definisanu funkcijama f_1 i f_2. Raylog tome je postojanje deljivih poddijagrama u procesu izračunavanja preko MTBDD-a za funkcije f_1 i f_2.

Poglavlje 3

Autokorelacija prekidačkih funkcija

U ovom poglavlju biće razmatrane definicije, osobine i primene autokorelacije prekidačkih funkcija. Osim toga, biće razmatran problem izračunavanja autokorelacionih koeficijenata prekidačkih funkcija. Ovde su predstavljenji i diskutovani postojeći algoritmi za izračunavanje autokorelacionih koeficijenata (algoritam iscrpljivanja, algoritam preko disjunktnih kubova i Wiener-Khinchin algoritam preko spektralnih koeficijenatasu). Svaki od ovih algoritama je ilustrovan adekvatnim primerom izračunavanja autokorelacije za zadatu prekidačku funkciju.

3.1 Definicija autokorelacije prekidačkih funkcija

Opšta konvoluciona (kros-korelaciona) funkcija između dve prekidačke funkcije $f(X)$ i $g(X)$ na distanci u, gde je $X = x_1x_2...x_n$ sa ekvivalentnom celobrojnom vrednošću $X = \sum_{i=1}^{n} x_i 2^{n-i}$, i gde je n broj promenljivih funkcije, je definisana kao [**?**]:

$$B_{fg}(u) = \sum_{u=0}^{2^n-1} f(v)g(v \oplus u), \tag{3.1}$$

gde je $u = u_1u_2...u_n$ sa ekvivalentnom celobrojnom vrednošću $u = \sum_{i=1}^{n} u_i 2^{n-i}$ i $v = v_1v_2...v_n$ sa ekvivalentnom celobrojnom vrednošću $v = \sum_{i=1}^{n} v_i 2^{n-i}$ i gde znak $\oplus$ predstavlja XOR logičku operaciju nad bitovima.

Ako u prethodno definisanoj jednačini važi da je $f(X) = g(X)$, rezultujuća jednačina daje konvoluciju prekidačke funkcije sa samom sobom ili autokorelacionu funkciju na distanci u. Rezultujući koeficijenti se nazivaju autokorelacionim koeficijentima prekidačke

funkcije. Autokorelaciona funkcija je definisana kao [**?**]:

$$B(u) = \sum_{v=0}^{2^n-1} f(v)f(v \oplus u). \tag{3.2}$$

Ako se za vrednosti ulaza prekidačke funkcije koristi $\{+1, -1\}$ kodiranje, rezultujuća autokorelaciona funkcija se označava sa $C(u)$:

$$C(u) = \sum_{v=0}^{2^n-1} f(v)f(v \oplus u). \tag{3.3}$$

Autokorelaciona funkcija izračunava meru sličnosti između funkcije i iste te funkcije posmatrane na nekoj distanci (pomeraju). Celobrojne vrednosti u uzimaju vrednosti u opsegu 0 do $2^n - 1$. Autokorelaciona funkcija (ili transformacija) kada se primenjuje na izlaze funkcije $f(X)$ transformiše vrednosti izlaza iz Boole-ovog domena u celobrojni domen. Na primer izračunavanje 3. autokorelaciong koeficijenta za prekidačku funkciju definisanu sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ čiji je vektor istinitosti $F = [00101111]^T$ prikazan je u sledećoj jednačini:

$$\begin{aligned} B(3) = \sum_{v=0}^{7} f(v)f(v \oplus 3) =& f(000)f(011) + f(001)f(010) + f(010)f(001) + \\ &+ f(011)f(000) + f(100)f(111) + f(101)f(110) + \\ &+ f(110)f(101) + f(111)f(100) = \\ &= 0 \cdot 0 + 0 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 \\ &+ 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 4 \end{aligned} \tag{3.4}$$

Rezultujući uređeni skup autokorelacionih koeficijenata predstavlja autokorelacioni spektar prekidačke funkcije koji se označava sa B_f. Primer autokorelacionog spektra za prekidačku funkciju f predstavljenu preko vektora istinitosti $F = [0, 0, 1, 0, 1, 1, 1, 1]^T$ je $B_f = [5, 4, 4, 4, 2, 2, 2, 2]^T$.

Za višeizlazne prekidačke funkcije neophodno je uvođenje dodatnog koraka kako bi se kombinovale autokorelacione funkcije svakog izlaza posebno u totalnu autokorelacionu funkciju. Totalna autokorelaciona funkcija $B(u)$ višeizlazne prekidačke funkcije sa m izlaza, u oznaci $F = (f_0 f_1 ... f_{m-1})$, definisana je sledećom jednačinom [**?**]:

$$B(u) = \sum_{i=0}^{m-1} B_i(u) = \sum_{i=0}^{m-1} \sum_{v=0}^{2^n-1} f_i(v)f_i(v \oplus u). \tag{3.5}$$

Iz jednačina 3.3 i 3.4 je evidentno da izračunavanje autokorelacionih koeficijenata zahteva 2^n operacija za izračunavanje svakog od 2^n koeficijenta. Zbog toga su vreme izračunavanja i zahtevi za računarskim resursima potrebnih za izračunavanje eksponencijalni u odnosu na broj promenljivih funkcije n.

Autokorelacioni koeficijenti mogu da se izračunaju i preko Walsh-ovih spektralnih koeficijenata prekidačke funkcije. Wiener-Khinchin-ova teorema definiše vezu između autokorelacionog spektra i Walsh-ovog spektra prekidačke funkcije [**?**]:

$$B_f = 2^{-n} W(n) S_f^2. \tag{3.6}$$

Ako izvršimo smenu opisanu jednačinom 2.6 dobijamo direktnu vezu između autokorelacionog spektra i vektora istinitosti:

$$B_f = 2^{-n} W(n)(W(n)F)^2. \tag{3.7}$$

Primer izračunavanja autokorelacionog spektra za prekidačku funkciju f predstavljenu preko vektora istinitosti $F = [0,0,1,0,1,1,1,1]^T$ koristeći Wiener-Khinchin-ovu teoremu je prikazan u sledećoj jednačini:

$$\begin{aligned} B_f =& \frac{1}{8} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \cdot \\ & \cdot \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right)^2 = \begin{bmatrix} 5 \\ 4 \\ 4 \\ 4 \\ 2 \\ 2 \\ 2 \\ 2 \end{bmatrix} \end{aligned} \tag{3.8}$$

Iz jednačine 3.6 je evidentno da vrednosti Walsh-ovih koeficijenata u procesu izračunavanja autokorelacionih koeficijenta moraju da budu kvadrirane. U procesu kvadriranja

informacije o znaku Wlash-ovih koeficijenata se gube. Iz tog razloga autokorelaciona funkcija nije reverzibilna, odnosno iz autokorelacionog spektra prekidačke funkcije se ne može rekonstruisati prekidačka funkcija.

3.2 Osobine autokorelacije prekidačkih funkcija

Ako se indeksi autokorelacionih koeficijenata posmatraju kao binarne vrednosti, izračunavanje, na primer 3. autokorelacionog koeficijenta za prekidačku funkciju od 3 promenljivih, može da se prikaže sledećom jednačinom:

$$B(000) = f(000) \cdot f(000 \oplus 011) + \ldots + f(111) \cdot f(111 \oplus 011). \tag{3.9}$$

Iz prethodne jednačine je evidentno značenje autokorelacionih koeficijenata. Ono što se odmah može zaključiti je da svaka XOR operacija vrši invertovanje bitova ulazne vrednosti prekidačke funkcije. Vrši se invertovanje samo bitova na kojima binarna reprezentacija indeksa autokorelacionog koeficijenta ima vrednost 1. Kao što je naznačeno u prethodnoj sekciji uređenje ulaznih promenljivih je u redosledu $x_1 \ldots x_n$. Zbog toga može da se uvede i alternativno označavanje gde se za svaki koeficijent označava koje promenljive će biti invertovane, što je prikazano u sledećoj jednačini [**?**]:

$$\begin{bmatrix} B(0) \\ B(1) \\ B(2) \\ B(3) \\ B(4) \\ B(5) \\ B(6) \\ B(7) \end{bmatrix} = \begin{bmatrix} B(000) \\ B(001) \\ B(010) \\ B(011) \\ B(100) \\ B(101) \\ B(110) \\ B(111) \end{bmatrix} = \begin{bmatrix} B(0) \\ B(x_3) \\ B(x_2) \\ B(x_2x_3) \\ B(x_1) \\ B(x_1x_3) \\ B(x_1x_2) \\ B(x_1x_2x_3) \end{bmatrix} \tag{3.10}$$

Autokorelacioni koeficijenti mogu se podeliti u grupe u skladu sa brojem jedinica u binarnoj reprezentaciji broja u. Na primer, B(0) je autokorelacioni koeficijent nultog reda, B(1), B(2) i B(3) su autokorelacioni koeficijenti prvog reda, B(3), B(5) i B(6) su autokorelacioni koeficijenti drugog reda, i tako dalje. Autokorelacioni koeficijenti generalno predstavljaju korelaciju između vrednosti funkcije u tačkama na fiksnim distancama. Koeficijenti prvog reda predstavljaju korelaciju između vrednosti funkcije u tačkama na distanci 1. Koeficijenti k-tog reda predstavljaju korelaciju između vrednosti funkcije u tačkama na distanci k, itd. Preciznije, red autokorelacionog koeficijenta $B(u)$ se definiše kao težina $|u|$, ili broj jedinica u binarnoj reprezentaciji broja u.

Autokorelacioni koeficijenti funkcije su koeficijenti koji daju meru sličnosti između funkcije f koja se razmatra i iste te funkcije f koja je modifikovana na neki način. U ovom slučaju, koeficijenti prvog reda određuju sličnost između funkcija f i g gde je $g(x_1, \ldots, x_i, \ldots, x_n) = f(x_1, \ldots, \bar{x}_i, \ldots, x_n)$. Koeficijenti drugog reda određuju sličnost između funkcija f i h gde je h funkcija f gde su dve promenljive komplementirane [**?**]. Ovakvo razmatranje može da se nastavi do koeficijenta n-tog reda.

Postoje brojna ograničenja vrednosti autokorelacionih koeficijenata koja važe i kod $\{1, -1\}$ kodiranja prekidačke funkcije [**?**]. Poznavajući ova ograničenja može se veoma jednostavno ispitati korektnost izračunatih koeficijenata. Za autokorelacione koeficijente važe sledeće osobine:

- $B(u) \in \{0, \ldots, 2^n\}$ za $\forall u$.
- $B(u)$ je paran za $\forall u \neq 0$.
- $B(u) \leq B(0)$ za $\forall u \neq 0$ i $B(0) = k$ gde k označava broj minterma prekidačke funkcije.
- $C(u) \in \{-2^n, \ldots, 2^n\}$ i ujednačeno deljivo sa 2 za $\forall u$.
- Prekidačka funkcija može imati najviše 2^{n-1} negativnih $C(u)$ koeficijenata.
- $C(u) \leq C(0)$ za $\forall u \neq 0$ i $C(0) = 2^n$.
- $\sum_{i=0}^{2^{n-1}} B(u) = k^2$.
- $\sum_{i=0}^{2^{n-1}} C(u) = (2^n - 2k)^2$.

Primećuje se da autokorelacioni koeficijent može da ima maksimalnu vrednost 2^n. Na primer, maksimalna vrednost autokorelacionog koeficijenta prekidačke funkcije od 65 promenljivih može da bude $2^{65} = 36.893.488.147.419.103.232$. Zbog toga izračunavanje autokorelacionog koeficijenta na računaru za prekidačke funkcije sa velikim brojem promenljivih zahteva rad sa posebnim tipovima podataka koji podržavaju rad sa velikim celobrojnim vrednostima.

3.3 Primene autokorelacije prekidačkih funkcija

U ovoj sekciji su opisane neke tipične primene autokorelacionih koeficijenata prekidačkih funkcija.

Autokorelacioni koeficijenti prekidačke funkcije pokazuju koliko je funkcija slična samoj sebi kada je određeni broj promenljivih invertovan. Ova informacija može biti primenjena kod određivanja promenljivih koje treba grupisati u cilju dobijanja BDD-a sa što više deljivih poddijagrama. Ako dve ili više promenljivih koje su invertovane daju veliku apsolutnu vrednost autokorelacionog koeficijenta, grupisanje ovih promenljivih može rezultovati da slični delovi funkcije postanu deljivi u okviru BDD-a. Eksperimentalno se pokazalo da ako dve invertovane promenljive imaju veliku apsolutne vrednosti autokorelacionog koeficijenta prvog reda, grupisanje ovih promenljivih može rezultovati veću deljivost u okviru BDD-a. Detaljniji opis ovih primena autokorelacije i njena eksperimentalna evaluacija su prikazane u radovima [**?**], [**?**], [**?**], [**?**], [**?**], [**?**].

Linearizacija prekidačkih funkcija podrazumeva predstavljanje datog sistema prekidačkih funkcija kao superpoziciju sistema linearnih prekidačkih funkcija i na taj način stvaranje nelinearnih delova sa minimalnom kompleksnošću. Na ovaj način proizvedena logička mreža se sastoji od serijske veze linearnih i nelinearnih blokova. Linearni blokovi se sastoje samo od XOR logičkih elemenata. Kompleksnost linearnih blokova se može zanemariti, dok je kompleksnost nelinearnih blokova redukovana u odnosu na kompleksnost polazne funkcije. Kod rešavanja problema optimalne linearizacije, konstruisanje linearne transformacione matrice se može izvršiti na osnovu skupa nezavisnih promenljivih koje imaju maksimalne vrednosti autokorelacionih koeficijenta. Ovakva linearizacija, testirana na benčmark funkcijama sa velikim brojem promenljivih, pokazuje veliki stepen redukcije kompleksnosti polazne funkcije. Detaljniji opis ovih primena autokorelacije i njena eksperimentalna evaluacija su prikazane u radovima [**?**], [**?**], [**?**], [**?**].

S obzirom da ima 2^{2^n} mogućih Boolean-ovih funkcija od n promenljivih, postoji potreba za grupisanjem ovih funkcija na osnovu njihovih osobina. Ciljevi grupisanja omogućavaju jednostavnije upravljanje informacijama u okviru grupe funkcija, kao i lakše proučavanje prekidačkih funkcija sa sličnim osobinama u okviru jedne grupe. Osim toga, za svaku klasu se može odrediti standardna ili kanonička funkcija (prototip te klase) preko koje mogu da se realizuju sve funkcije u toj klasi, što se može koristiti u procesu sinteze i testiranja prekidačkih mreža. Autokorelacione klase se definišu na onovu četiri invarijantne operacije: permutacija bilo koje ulazne promenljive, negacija bilo koje ulazne promenljive, negacija izlaza prekidačke funkcije i zamena bilo koje ulazne promenljive x_i sa $x_i \oplus x_j$, $(i, j \in \{1 \dots n\}, i \neq j)$. Autokorelacione klase omogućavaju da se za prekidačke funkcije u istoj autokorelacionoj klasi može izvršiti sinteza iz kanoničkih funkcija dodavanjem komplementarne logike i permutovanjem promenljivih. Problem u primeni autokorelacionih klasa je što invarijantna operacija negacije ulaza ne utiče na autokorelacione koeficijente. Kod drugih invarijantnih operacija ispitivanjem promena u autokorelacionim koeficijentima može se odrediti koje operacije su bile primenjene na

originalnu funkciju. Detaljniji opis ove primene autokorelacije i njena eksperimentalna evaluacija je prikazana u radu [**?**].

3.4 Izračunavanje autokorelacionih koeficijenta preko algoritma iscrpljivanja

Formule za izračunavanje autokorelacijonih koeficijenata prekidačkih funkcija, što je detaljno opisano na početku ovog poglavlja, zahtevaju 2^n složenih operacija nad 2^n različitih vrednosti funkcije za izračunavanje svakog od 2^n autokorelacionog koeficijenta, gde n predstavlja proj promenljivih prekidačke funkcije. Ako se izvrši direktna implementacija formule, izračunavanje pojedinačnog autokorelacionog koeficijenta bez pokušaja da se poboljša efikasnost izračunavanja zahteva primenu 2^n operacija množenja i $2^n - 1$ operacija sabiranja. Ovakav način izračunavanja autokorelacije poznat je pod imenom "algoritam iscrpljivanja". Na primer, za izračunavanje jednog autokorelaciong koeficijenta prekidačke funkcije od $n = 30$ promenljivih potrebno je oko jedna miljarda operacija množenja i oko 500 miliona operacija sabiranja nad isto tolikim brojem različitih vrednosti funkcije. Takođe je potrebna oko jedna milijarda memorijskih lokacija za smeštanje izračunatih različitih vrednosti autokorelacionih koeficijenata (oko 3,72 GB memorijskog prostora za 32-bitni memorijski sistem).

Stoga, izračunavanje autokorelacije preko algoritma iscrpljivanja za prekidačke funkcije sa relativno malim brojem ulaznih promenljivih postaje neizvodljivo u realnom vremenu. Nagli porast dostupnosti računarskih resursa (povećanje kapaciteta memorije i brzine procesorskih jedinica) učinili su da izračunavanje autokorelacije do neke mere može da se izvrši u realnom vremenu. Izračunavanje autokorelacije može biti izvršavano korišćenjem višejezgarnih procesora [**?**], [**?**], [**?**] i mnogojezgarnih grafičkih kartica [**?**], [**?**] sa esencijalnim razlikama u hardverskim resursima ili kombinovano korišćenjem oba hardverska resursa istovremeno [**?**].

U pokušaju da se reši ovaj problem razvijene su nekoliko mogućnosti za poboljšanje efikasnosti algoritma iscrpljivanja. Postoji nekoliko mogućnosti za poboljšanje efiksnosti izračunavanja autokorelacionih koeficijenata kod algoritma iscrpljivanja [**?**].

Izračunavanje autokorelacionog koeficijenta nultog reda $B(0)$ ne mora da zahteva 2^n operacija množenja, već samo $2^n - 1$ operacija sabiranja. Autokorelacioni koeficijent nultog reda u oznaci $B(0)$ predstavlja broj nenultih elemenata u vektoru istinitosti prekidačke

funkcije i može se definisati sledećom jednačinom:

$$B(0) = \sum_{v=0}^{2^n-1} f(v)f(v \oplus 0) = \sum_{v=0}^{2^n-1} f(v)f(v) = \sum_{v=0}^{2^n-1} f(v). \tag{3.11}$$

Takođe se može pokazati da izračunavanje autokorelacionih koeficijenata ne mora da zahteva 2^n operacija množenja, već upola manje. Primer izračunavanja 3. autokorelacionog koeficijenta prekidačke funkcije sa tri promenljivih $f(x_1, x_2, x_3)$ pokazuje prethodno tvrdjenje:

$$\begin{aligned} B(3) &= \sum_{v=0}^{7} f(v)f(v \oplus 3) = f(000)f(011) + f(001)f(010) + f(010)f(001) + \\ &+ f(011)f(000) + f(100)f(111) + f(101)f(110) + f(110)f(101) + f(111)f(100) = \\ &= 2(f(000)f(011) + f(001)f(010) + f(100)f(111) + f(101)f(110)) \end{aligned} \tag{3.12}$$

Dvostruko manji broj operacija množenja je posledica osobine da je $f(v) = f(v \oplus u \oplus u)$, tako da se jednačina autokorelacione funkcije može napisati u sledećem obilku:

$$B(u) = 2 \sum_{v=0}^{2^{(n-1)-1}} f(v)f(v \oplus u). \tag{3.13}$$

Ako se prekidačka funkcija predstavi preko "1-polja" ili "1-kubova" (što je opisano u prethodnom poglavlju) moguće je dodatno smanjiti broj operacija množenja i sabiranja potrebnih za izračunavanje autokorelacionih koeficijenata. Razlog smanjenja broja operacija je taj što izračunavanje ne zahteva svih 2^n vrednosti prekidačke funkcije već samo vrednosti funkcije na kojima funkcija ima vrednost 1. U ovom slučaju broj operacija za izračunavanje svakog koeficijenta zavisi od broja nenultih vrednosti prekidačke funkcije.

Međutim u praksi se pokazalo da primenom prethodno opisanih poboljšanja kod algoritma iscrpljivanja ne doprinosi značajno poboljšanju efikasnosti izračunavanja [**?**]. Eksponencialno vreme izračunavanja autokorelacionih koeficijenata u odnosu na broj promenljivih funkcije je i dalje ograničavajući faktor. Zbog toga su razvijani algoritmi sa drugačijim pristupom izračunavanja u odnosu na pristup gde se definicija autokorelacione funkcije koristi direktno, kao što su algoritmi preko disjunktnih kubova i preko Wiener-Khinchin-ove teoreme i brze Walshove transformacije.

3.5 Izračunavanje autokorelacionih koeficijenta preko disjunktnih kubova

Algoritma za izračunavanje autokorelacioih koeficijenata preko disjunktnih kubova koristi predstavljanje prekidačkih funkcija preko kubova i zbog toga je kompaktnija u odnosu na algoritme za izračunavanje autokorelacionih koeficijenata preko vektora istinitosti. Kompaktnost ovog pristupa se ogleda u tome da za izračunavanje koeficijenata nije potrebno svih 2^n ulaznih kombinacija, gde je n broj ulaznih promenljivih funkcije. Osim toga, jedan kub može da predstavi više ulaznih kombinacija, pri čemu "don't care" vrednosti funkcije ne utiču na izračunavanje vrednosti autokorelacionih koeficijenata. Algoritam za izračunanje autokorelacionih koeficijenata preko skupa disjunktnih kubova se bazira na tehnici izračunavanja Rademacher-Walshovih koeficijenata preko kubova opisanoj u [**?**]. Ovaj algoritam je nadgradnja metode koja je prethodno objavljena u [**?**].

Kod ovog algoritma se zahteva generisanje skupa disjunktnih kubova. Lista disjunktnih 1-kubova može da se odredi nekom od postojećih metoda za generisanje [**?**], [**?**], [**?**].

Algoritam za izračunavanje pojedinačnog autokorelacionog koeficijenata preko disjunktnih kubova može da se formuliše u 3 koraka:

1. Generiše se skup disjunktnih kubova prekidačke funkcije.
2. U skupu disjunktnih kubova poredi se savaki kub sa svakim, rezultat poređenja $P_{a,b}$ kuba a i kuba b se dobija primenom $\oplus$ (XOR) logičke operacije nad bitovima kuba. Svakom rezultatu poređenja se pridružuje vrednost doprinosa $D_{a,b}$ koja se izračunava po sledećim pravilima: ako se poredi kub sa samim sobom $D_{a,b} = 2^d$, gde d predstavlja broj "don't-care" vrednosti u kubu. U slučaju da se kub poredi sa nekim drugim kubom $D_{a,b} = 2 * 2^d$, gde d predstavlja broj "dont-care" vrednosti na istim pozicijama u oba kuba koji se porede.
3. Vrednost i-tog autokorelacionog koeficijenta prekidačke funkcije se izračunava kao suma vrednosti doprinosa čija binarna vrednost rezultata poređenja ima dekadnu vrednost i. U analitičkom obliku: $B(i) = \sum_{(P_{a,b} \Leftrightarrow i)} D_{a,b}$.

Izračunavanje, na primer 1. autokorelacionog koeficijenata prekidačke funkcije zadate preko vektora istinitosti $F = [00011111]^T$ primenom prethodno opisanog algoritma ilustrovano je na slici i jednačinama u nastavku. Na slici 3.1 prikazani su skupovi 1-polja i 1-disjunktnih kubova zadate prekidačke funkcije (korak 1 u algoritmu). U jednačini 3.14 prikazani su rezultati poređenja kubova u skupu disjunktnih kubova gde se poredi savaki kub sa svakim i vrednosti odgovarajućih doprinosa (korak 2 u algoritmu). U jednačini 3.15 prikazan je postupak izračunavanja 1. autokorelacionog koeficijenta prekidačke

funkcije na osnovu rezultata poređenja disjunktnih kubova i njihovoih odogvarajućih doprinosa (korak 3 u algoritmu).

$1-polje$			$1-kubovi$		
x_1	x_2	x_3	x_1	x_2	x_3
0	1	1	0	1	1
1	0	0	1	0	-
1	0	1	1	1	-
1	1	0			
1	1	1			

SLIKA 3.1: 1-polje i 1-disjunktni kubovi prekidačke funkcije zadate preko vektora istinitosti $F = [00011111]^T$

$$\begin{aligned}
&P_{011,011} = \text{"}000\text{"} \quad D_{011,011} = 2^0 = 1 \\
&P_{011,10-} = \text{"}11-\text{"} \quad D_{011,10-} = 2 * 2^0 = 2 \\
&P_{011,11-} = \text{"}10-\text{"} \quad D_{011,11-} = 2 * 2^0 = 2 \\
&P_{10-,10-} = \text{"}00-\text{"} \quad D_{10-,10-} = 2^1 = 2 \\
&P_{10-,11-} = \text{"}01-\text{"} \quad D_{10-,11-} = 2 * 2^1 = 4 \\
&P_{11-,11-} = \text{"}00-\text{"} \quad D_{11-,11-} = 2^1 = 2
\end{aligned} \tag{3.14}$$

$$\begin{aligned}
&B(1) = \sum_{P_{a,b} \Leftrightarrow 1} D_{a,b} = D_{10-,10-} + D_{11-,11-} = 4 \\
&P_{10-,10-} = \text{"}00-\text{"} \Leftrightarrow 1 \\
&P_{11-,11-} = \text{"}00-\text{"} \Leftrightarrow 1
\end{aligned} \tag{3.15}$$

Broj operacija potreban za izračunavanje autokorelacionih koeficijenata prekidačke funkcije preko disjunktnih kubova linearno je zavisan od broja disjunktnih kubova funkcije. Potrebno je $n * \frac{p(p+1)}{2}$ operacija množenja po modulu 2, gde je n broj ulaznih promenljivih funkcije i p broj disjunktnih kubova funkcije. Eksponencijalno vreme izračunavanja autokorelacionih koeficijenata u odnosu na broj disjunktnih kubova prekidačke funkcije je i dalje ograničavajući faktor. Može se primetiti da je ovaj algoritam pogodan samo za izračunavanje pojedinačnih autokorelacionih koeficijenata.

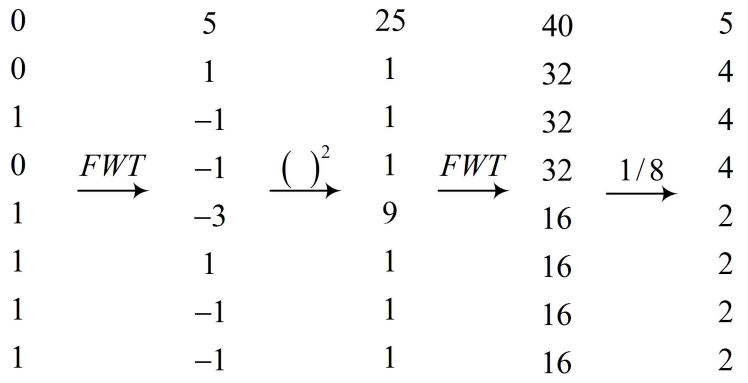
3.6 Izračunavanje autokorelacionih koeficijenta preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije

Algoritmi za izračunavanje autokorelacionih koeficijenata imaju eksponencijalnu kompleksnost u odnosu na broj promenljivih funkcije. U ovom poglavlju su prethodno predstavljeni algoritmi za izračunavanje pojedinačnih autokorelacionih koeficijenata čije strategije izračunavanja uglavnom zavise od strukture podataka kojom je predstavljena prekidačka funkcija. U slučaju da je potrebno izračunavanje kompletnih autokorelacionih koeficijenata, algoritmi za izračunavanje pojedinačnih autokorelacionih koeficijenata moraju da se ponove 2^n puta kako bi se izračunalo svih 2^n autokorelacionih koeficijenta, gde n predstavlja proj promenljivih prekidačke funkcije.

U slučaju da je potrebno efikasno izračunavanje kompletnih autokorelacionih koeficijenta (autokorelacioni spektar) neophodno je korišćenje Wiener-Khinchin-ove teoreme. Wiener-Khinchin-ova teorema definiše vezu između autokorelacionog spektra i Walsh-ovog spektra prekidačke funkcije, što je detaljno opisano na po;etku ovog poglavlja. Brza Walsh-ova transformacija može se iskoristititi za dodatno poboljšanje efikasnosti izračunavanja kompletnih autokorelacionih koeficijenata preko Wiener-Khinchin-ove teoreme.

Izračunavanje Walsh-ovog spektra može biti izvršena preko grafa toka operacija koji opisuje FWT algoritam koji je detaljno opisan u prethodnom poglavlju. Izračunavanje autokorelacionih koeficijenata preko Wiener-Khinchin-ove teoreme definisane jednačinom 3.7 može da se izvrši korišćenjem FWT algoritma. Korišćenje FWT algoritma umesto matričnog množenja sa Walsh-ovom matricom na dva mesta u jednačini značajno poboljšava efikasnost izračunavanja autokorelacionog spektra.

Primer izračunavanja autokorelacionog spektra preko Wiener-Khinchin-ove teoreme i FWT algoritma prekidačke funkcije definisane sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ čiji je vektor istinitosti $F = [00101111]^T$ je prikazan je na slici 3.2. Korak 1 prikazuje izračunavanje Walsh-ovog spektra $S_f = [5, 1, -1, -1, -3, 1, -1, -1]^T$ iz vektora istinitosti prekidačke funkcije $f(x_1, x_2, x_3)$. U koraku 2, Walsh-ovi koeficijenti se kvadriraju. Korak 3 generiše autokorelacioni spektar preko izračunavanja Walsh-ove transformacije kvadriranog Walsh-ovog spektra i može se označiti sa $S_{S_f{}^2}$. U posledenjem koraku autokorelacioni spektar se množi faktoromn normalizacije koji ima vrednost 1/8. Može se primetiti da nakon izračunavanja druge Walsh-ove transformacije vrednosti autokorelacionih koeficijenta mogu da imaju maksimalnu vrednost 2^{2n}. Na primer, maksimalnu vrednost autokorelacionog koeficijenta pre normalizacije prekidačke funkcije od 32 promenljivih može da bude $2^{32} \approx 3.6 \cdot 10^{19}$. Zbog toga izračunavanje autokorelacionog koeficijenta na računaru za prekidačke funkcije sa velikim brojem promenljivih zahteva rad sa posebnim tipovima podataka koji podržavaju rad sa velikim celobrojnim vrednostima. Kod izračunavanja



SLIKA 3.2: Izračunavanje autokorelacionog spektra prekidačke funkcije definisane sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ čiji je vektor istinitosti $F = [00101111]^T$ preko Wiener-Khinchin-ove teoreme i FWT algoritma

Walshovog spektra, kao što je prikazano na slici 2.11, redosled izračunavanja u svakom koraku može biti preuređen tako da odgovara strukturi BDD-a kojim je predstavljena prekidačka funkcija. Brza Walsh-ova transformacija preko BDD-a može se iskoristititi za dodatno poboljšanje efikasnosti izračunavanja kompletnih autokorelacionih koeficijenata preko Wiener-Khinchin-ove teoreme.

Poglavlje 4

Izračunavanje autokorelacionih koeficijenta preko dijagrama odlučivanja

U ovom poglavlju biće razmatran problem izračunavanja autokorelacionih koeficijenata prekidačkih funkcija preko raznih vrsta dijagrama odlučivanja. Ovde su predstavljeni i diskutovani novi razvijeni algoritmi za izračunavanje autokorelacionih koeficijenata na osnovu postojećih algoritama (SBDD algoritam sa permutovanim labelama, SBDD algoritam sa bočnim kretanjem, SMTBDD algoritam preko Wiener-Kinchin-ove teoreme i brze Walsh-ove transformacije i SMTBDD algoritam preko BDD paketa sa dinamičkim terminalnim čvorovima). Svaki od ovih algoritama je ilustrovan adekvatnim primerom izračunavanja za zadatu prekidačku funkciju.

4.1 Algoritmi za izračunavanje pojedinačnih autokorelacionih koeficijenata preko dijagrama odlučivanja

4.1.1 SBDD algoritam sa permutovanim labelama

Algoritmi za izračunavanje autokorelacionih koeficijenata prekidačkih funkcija koji se baziraju na dijagramima odlučivanja predstavljeni su u [?], [?] i [?]. Analiza ovih algoritama pokazala je veću efikasnost u odnosu na standardne algoritme (opisane u prethodnom poglavlju), pri čemu se efikasnost odnosi kako na vreme izračunavanja tako i na raspoložive memorijske resurse koji se koriste u procesu izračunavanja [?], [?], [?] i [?]. Na osnovu postojećeg algoritma za izračunavanje autokorelacionih

koeficijenata preko BDD-a sa permutovanim labelama na potezima, u ovom radu se predlaže novi algoritam za izračunavanje autokorelacionih koeficijenata preko razdeljenog BDD-a sa permutovanim labelama na granama za prekidačke funkcije sa velikim brojem promenljivih [**?**].

U matričnoj notaciji, ako se prekidačka funkcija f sa n promenljivih, predstavljena preko vektora istinitosti $F = [f(0), f(1), ..., f(2^n - 1)]^T$ i njen autokorelacioni spektar preko autokorelacionog vektora B_f (definisan u jednačini 3.6), tada se izračunavanje autokorelacionog spektra prekidačke funkcije može predstaviti sledećom jednačinom:

$$B_f = B(n)F, \tag{4.1}$$

gde je $B(n)$ dijadička autokorelaciona matrica za funkciju f [**?**]. Dijadička autokorelaciona matrica može da se definiše preko vrednosti u vektoru istinitosti $F = [f(0), f(1), ..., f(2^n - 1)]^T$:

$$B(n) = \begin{bmatrix} f(0) & f(1) & \dots & f(2^n-2) & f(2^n-1) \\ f(1) & f(0) & \dots & f(2^n-1) & f(2^n-2) \\ \vdots & \vdots & \dots & \vdots & \vdots \\ f(2^n-2) & f(2^n-1) & \dots & f(0) & f(1) \\ f(2^n-1) & f(2^n-2) & \dots & f(1) & f(0) \end{bmatrix} \tag{4.2}$$

Na osnovu dijadičke autokorelacione matrice vektor autokorelacionih koeficijenata se može predstaviti u obliku:

$$B_f = \begin{bmatrix} f(0)^2 + f(1)^2 + \ldots + f(2^n-2)^2 + f(2^n-1)^2 \\ 2(f(0)f(1) + f(2)f(3) + \ldots + f(2^n-3)f(2^n-4) + f(2^n-1)f(2^n-2)) \\ \vdots \\ f(0)f(2^n-2) + f(1)f(2^n-1) + \ldots + f(2^{n/2})f(2^{n/2+2}) + f(2^{n/2+1})f(2^{n/2+3}) \\ f(0)f(2^n-1) + f(1)f(2^n-2) + \ldots + f(2^{n/2})f(2^{n/2+3}) + f(2^{n/2+1})f(2^{n/2+2}) \end{bmatrix} \tag{4.3}$$

Izračunavanje autokorelacionih koeficijenata za višeizlaznu prekidačku funkciju korišćenjem prethodne jednačine zahteva $m * 2^{2n}$ operacija gde je m broj izlaza prekidače funkcije. Algoritam za izračunavanje autokorelacionih koeficijenata preko BDD-a sa permutovanim labelama na potezima koristi redukovanu reprezentaciju prve vrste dijadičke autokorelacione matrice, gde se ostale vrste matrice dobijaju iz prve vrste permutacijom labela na potezima BDD-a. Kod ovog algoritma koristi se osobina da je BDD reprezentacija funkcije $f(v \oplus u)$ (definisana u jednačini 2.9 ekvivalentna BDD reprezentaciji funkcije $f(v)$ sa permutovanim labelama na potezima BDD-a za sve čvorove koji odgovaraju promenljivoj v_i za koju važi $u_i = 1$, gde u_i predstavlja i-ti bit u binarnoj reprezentaciji za $u = (u_1, u_2, \ldots, u_n)$.

SBDD reprezentacija višeizlaznih prekidačkih funkcija (opisana u 2. poglavlju), jednačina za totalnu autokorelaciju (opisana u 3.5) i prethodna osobina za predstavljanje dijadičke autokorelacione matrice preko BDD-a koristi se za kreiranje sledećeg algoritma:

1. Za svaki totalni autokorelacioni koeficijent u kreira se SBDD za funkciju f, u oznaci $SBDD(f(v))$, i SBDD za funkciju f sa permutovanim labelama na potezima SBDD-a, gde se permutacija labela definiše na osnovu u, u oznaci $SBDD(f(v \oplus u))$.
2. Izvršava se množenje $SBDD(f(v))$ sa $SBDD(f(v \oplus u))$ korišćenjem standardne procedure za množenje predstavljene preko BDD-a [**?**], gde se dobija rezultujući $SBDD(f(v)f(v \oplus u))$.
3. Sumiranjem svih minterma preko SBDD-a (svih terminalnih čvorova na svim putevima u SBDD-u) dobija se vrednost totalnog autokorelacionog koeficijenta u.

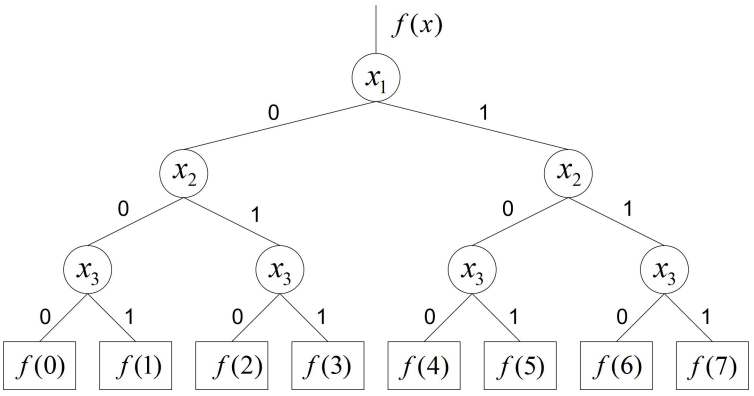
Primer izračunavanja 3. autokorelacionog koeficijenta za prekidačku funkciju sa 3 promenljive prikazan je u nastavku. Za $n = 3$, dijadička autokorelaciona matrica je:

$$B(3) = \begin{bmatrix} f(0) & f(1) & f(2) & f(3) & f(4) & f(5) & f(6) & f(7) \\ f(1) & f(0) & f(3) & f(2) & f(5) & f(4) & f(7) & f(6) \\ f(2) & f(3) & f(0) & f(1) & f(6) & f(7) & f(4) & f(5) \\ f(3) & f(2) & f(1) & f(0) & f(7) & f(6) & f(5) & f(4) \\ f(4) & f(5) & f(6) & f(7) & f(0) & f(1) & f(2) & f(3) \\ f(5) & f(4) & f(7) & f(6) & f(1) & f(0) & f(3) & f(2) \\ f(6) & f(7) & f(4) & f(5) & f(2) & f(3) & f(0) & f(1) \\ f(7) & f(6) & f(5) & f(4) & f(3) & f(2) & f(1) & f(0) \end{bmatrix} \tag{4.4}$$

i vektor autokorelacionih koeficijenata je:

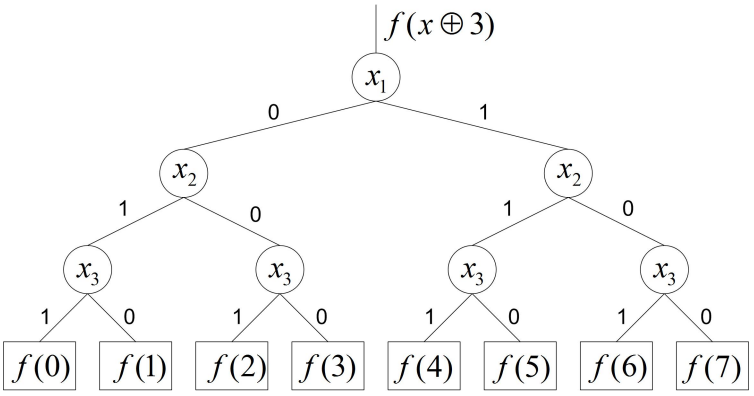
$$B_f = \begin{bmatrix} f(0)^2 + f(1)^2 + f(2)^2 + f(3)^2 + f(4)^2 + f(5)^2 + f(6)^2 + f(7)^2 \\ 2(f(0)f(1) + f(2)f(3) + f(4)f(5) + f(6)f(7)) \\ 2(f(0)f(2) + f(1)f(3) + f(4)f(6) + f(5)f(7)) \\ 2(f(0)f(3) + f(1)f(2) + f(4)f(7) + f(5)f(6)) \\ 2(f(0)f(4) + f(1)f(5) + f(2)f(6) + f(3)f(7)) \\ 2(f(0)f(5) + f(1)f(4) + f(2)f(7) + f(3)f(6)) \\ 2(f(0)f(6) + f(1)f(7) + f(2)f(4) + f(3)f(5)) \\ 2(f(0)f(7) + f(1)f(6) + f(2)f(5) + f(3)f(4)) \end{bmatrix} \tag{4.5}$$

Za $n = 3$, dijadička autokorelaciona matrica može da se predstavi preko BDT-a, $BDT(f)$, koji je prikazan na slici 4.1.



SLIKA 4.1: $BDT(f)$ kojim je predstavljena dijadička autokorelaciona matrica $B(3)$

Za predstavljanje i-te vrste matrice $B(3)$, odgovarajuće labele na granama na i-tom nivou BDT-a trebaju da budu permutovane. Na primer, za predstavljanje 3. vrste matrice $B(3)$, $u = 3 = (0, 1, 1)$ označava nivoe BDT-a na kojima labele na granama treba da budu permutovane. Labele na granama se permutuju kako bi obilaskom BDT-a dobili elemente 3. vrste matrice $B(3)$, što je prikazano na slici 4.2. Treba napomenuti da vrste matrice $B(3)$ imaju indekse od 0 do 7, tako da postoji i 0-ta vrsta.

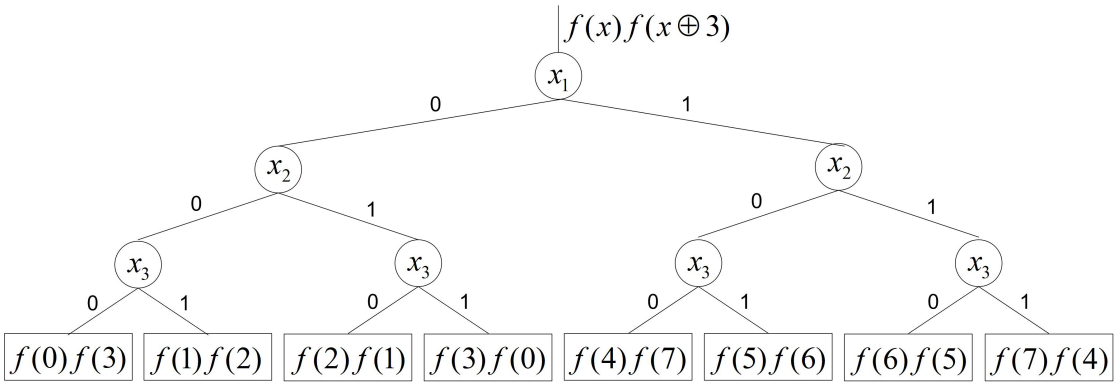


SLIKA 4.2: $BDT(f)$ kojim je predstavljena 3. vrsta dijadičke autokorelacione matrice $B(3)$

BDT kojim se predstavlja 3. autokorelacioni koeficijent se dobija množenjem $BDT(f)$ i $BDT(f)$ sa permutovanim labelama na granama za predstavljanje 3. vrste matrice $B(3)$ i prikazan je na slici 4.3.

Suma svih terminalnih čvorova koji se nalaze u $BDT(f(x)f(x \oplus 3))$ određuje vrednost 3. autokorelacionog koeficijenta prekidačke funkcije f:

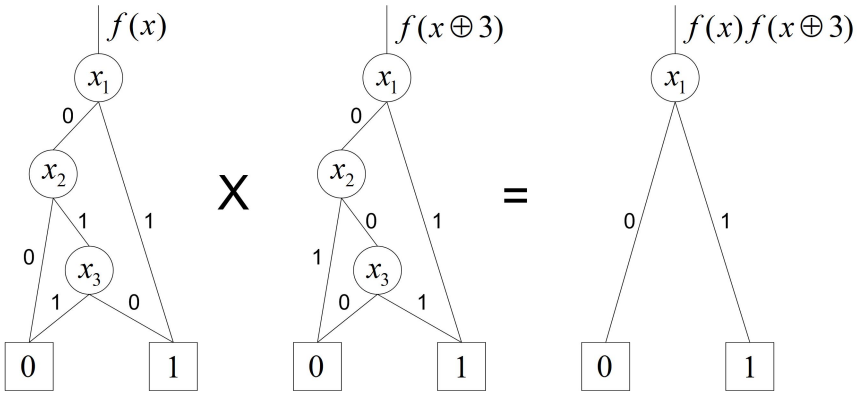
$$B_3 = f(0)f(3)+f(1)f(2)+f(2)f(1)+f(3)f(0)+f(4)f(7)+f(5)f(6)+f(6)f(5)+f(7)f(4) \tag{4.6}$$



SLIKA 4.3: $BDT(f(x)f(x \oplus 3))$ kojim se predstavlja 3. autokorelacioni koeficijent funkcije f

Na prethodnim slikama, dijadička autokorelaciona matrica $B(3)$ je predstavljena preko BDT-a. U praksi, izračunavanja se uvek izvršavaju nad BDD-om i prednosti izračunavanja se postižu iz kompaktnosti reprezentacije preko BDD-a u odnosu na BDT.

Primer izračunavanja 3. autokorelacionog koeficijenta za prekidačku funkciju definisanu sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ čiji je vektor istinitosti $F = [00101111]^T$ preko BDD-a sa permutovanim labelama na granama je prikazan je na slici 4.4. Suma svih vrednosti



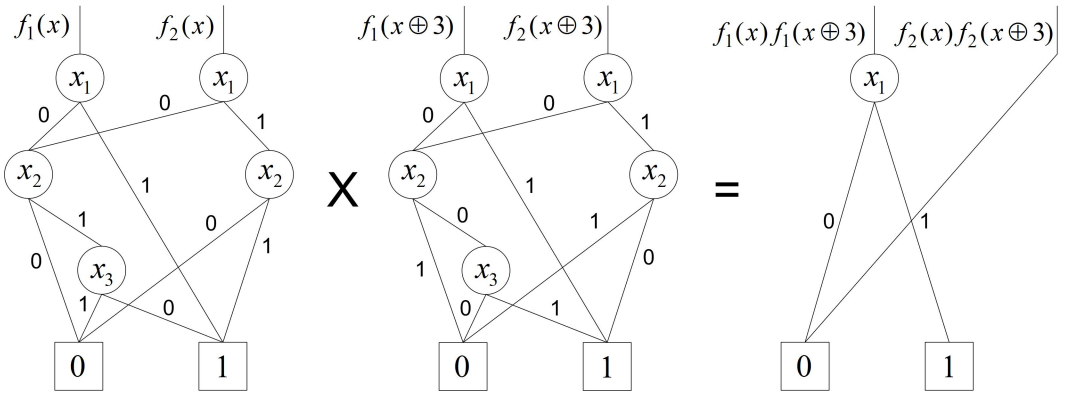
SLIKA 4.4: Primer izračunavanja 3. autokorelacionog koeficijenta za funkciju $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ preko BDD-a sa permutovanim labelama na granama

vektora istinitosti prekidačke funkcije predstavljene preko $BDD(f(x)f(x \oplus 3))$ određuje vrednost 3. autokorelacionog koeficijenta prekidačke funkcije f:

$$B_3 = 0 + 0 + 0 + 0 + 1 + 1 + 1 + 1 = 4 \tag{4.7}$$

Kod višeizlaznih prekidačkih funkcija za izračunavanje totalnih autokorelacionih koeficijenata se koristi SBDD-a. Zbog toga se u svakom koraku algoritma sva predstavljanja funkcija i sve operacije izvršavaju preko SBDD-a. Prednosti izračunavanja preko SBDD-a se postižu deljenjem izomorfnih podstabala u BDD-ovima preko kojih su predstavljene višeizlazne funkcije $f(x)$, $f(x \oplus 3)$ i $f(x)f(x \oplus 3)$. Primer izračunavanja 3. totalnog

autokorelacionog koeficijenta za višeizlaznu prekidačku funkciju čiji su izlazi definisani sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$ čiji su vektori istinitosti $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$ preko SBDD-a sa permutovanim labelama na granama je prikazan je na slici 4.5. Suma svih vrednosti vektora istinitosti višeizla-



Slika 4.5: Primer izračunavanja 3. totalnog autokorelacionog koeficijenta za višeizlaznu funkciju $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$ preko SBDD-a sa permutovanim labelama na granama

zne prekidačke funkcije predstavljene preko $SBDD(f(x)f(x \oplus 3))$ određuje vrednost 3. totalnog autokorelacionog koeficijenta funkcije $f(f_1, f_2)$:

$$B_3 = (0+0+0+0+1+1+1+1) + (0+0+0+0+0+0+0+0) = 4 \qquad (4.8)$$

Ovaj algoritam za izračunavanje totalnih autokorelacionih koeficijenata prekidačkih funkcija je pogodan samo za izračunavanje pojedinačnih autokorelacionih koeficijenata. Izračunavanje pojedinačnih totalnih autokorelacionih koeficijenata korišćenjem ovog algoritma za višeizlaznu prekidačku funkciju f zahteva $O(m*k)$ operacija, gde je m označava broj izlaza višeizlazne prekidačke funkcije f i k broj čvorova u $SBDD(f)$.

Opis implementacije ovog algoritma razmatran je i diskutovan u narednom poglavlju ove disertacije. U cilju ispitivanja efikasnosti predloženog algoritma, u slučaju ograničenih raspoloživih memorijskih resursa, izvršena je implementacija i eksperimentalno ispitivanje ovog algoritma što je detaljno opisano u narednim poglavljima ovog rada. Prednosti i mane ovog algoritma u slučaju izračunavanja za prekidačke funkcije sa velikim brojem promenljivih su takođe diskutovane.

4.1.2 SBDD algoritam sa bočnim kretanjem

Dijagrami odlučivanja predstavljaju strukturu podataka koja se godinama koristi kod efikasnih rekurzivnih predstavljanja podataka [**?**]. U radovima [**?**] i [**?**] ukazano je na mogućnost da se rekurzivna struktura dijadičke autokorelacione matrice može povezati

sa rekurzivnom strukturom BDD-a. U skladu sa tim, za eleminisanje eksponencijalne prostorne i vremenske kompleksnosti, razvijeni su algoritmi za izračunavanje autokorelacionih koeficijenata preko BDD-a. U radu [**?**] je pokazano da je moguće izračunavanje pojedinačnih dijadičkih autokorelacionih koeficijenata na dva principijalna načina. Prvi način, nazvan prost BDD metod, osim kreiranja inicijalnog BDD-a za predstavljanje prekidačke funkcije zahteva kreiranje novog BDD-a kod izračunavanja svakog autokorelacionog koeficijenta. Drugi način, nazvan rekurzivni BDD metod, u odnosu na prethodni zahteva kreiranje samo jednog BDD-a. Za oba metoda je izvršena eksperimentalna evaluacija korišćenjem CUDD BDD-paketa [**?**]. U [**?**] su predložena četiri različita metoda za izračunavanje dijadičkih autokorelacionih koeficijenata preko MTBDD-a i ternarnog dijagrama odlučivanja (eng. TDD - Ternary Decision Diagram) [**?**]. Izračunavanje autokorelacije preko BDD-a je takođe razmatrano u [**?**], gde je opisna metoda pod nazivom "in-place BDD" koja je veoma slična metodi "rekurzivni BDD". Svaka od ovih metoda je ograničena na izračunavanje autokorelacionih koeficijenata za jednoizlazne prekidačke funkcije. Zbog toga kod ovih metoda izračunavanje totalne autokorelacije zahteva izračunavanje pojedinačnih autokorelacionih koeficijenata za svaki izlaz a potom naknadno sumiranje svih izračunatih koeficijenata. Osim toga, ista procedura za izračunavanje pojedinačnih autokorelacionih koeficijenata se ponavlja m puta, gde je m broj izlaza prekidačke funkcije.

Ovde se može uočiti da prethodno razvijeni metodi umesto izračunavanja pojedinačnih koeficijenata preko BDD-a za svaki izlaz posebno mogu biti modifikovani za izračunavanje totalnih autokorelacionih koeficijenata preko SBDD-a. Sa ovom motivacijom, u ovom poglavlju predložen je algoritam za "in-place" izračunavanje pojedinačnih totalnih autokorelacionih koeficijenata preko SBDD-a korišćenjem efikasnog obilaska poddijagrama SBDD-a. Izračunavanje totalne autokorelacije jednim obilaskom SBDD-a trebalo bi da obezbedi ubrzanje, dok "in-place" organizacija redukuje potrebe za memorijskim prostorom. Osim toga, na osnovu efikasnih programskih tehnika kod BDD paketa [**?**], uvedena je dodatna heš tabela za ubrzavanje rekurzivnog izračunavanja preko SBDD-a. Dodatna heš tabela se koristi u cilju izbegavanja ponavljanja izračunavanja prilikom obilaska izomorfnih poddijagrama u okviru SBDD-a. Ovo predstavlja esencijalnu razliku u poređenju sa prethodnim metodama baziranih na izračunavanjima preko BDD-a. Predloženi algoritam iskorišćava rekurzivnu strukuturu autokorelacione operacije i SBDD-a [**?**], [**?**] i [**?**]. U cilju ispitivanja efikasnosti predloženog algoritma sa ograničenim memorijskim resursima izvršena je njegova implementacija proširenjem standardne tehnike za BDD-pakete, što je detaljno opisano u narednom poglavlju.

Kod višeizlaznih funkcija $f = (f_0 f_1 \dots f_{m-1})$, autokorelacione funkcije za svaki pojedinačni izlaz $B_i, i = 0 \dots m-1$ moraju da se sumiraju u totalnu autokorelacionu funkciju $B(\tau)$, što je definisano u jednačini 3.5. U matričnoj notaciji ako se funkcije f_i i njihovi

autokorelacioni koeficijenti B_i predstave kao vektori $F_i = [f_i(0), f_i(1), \ldots, f_i(2^n - 1)]^T$ i $B_i = [B_i(0), B_i(1), \ldots, B_i(2^n - 1)]^T$ respektivno, operacija totalne autokorelacije može da se definiše preko dijadičke autokorelacione matrice $B(n)$ kao:

$$B_i = B_i(n)F_i, \tag{4.9}$$

gde se matrica $B_i(n)$ sastoji od:

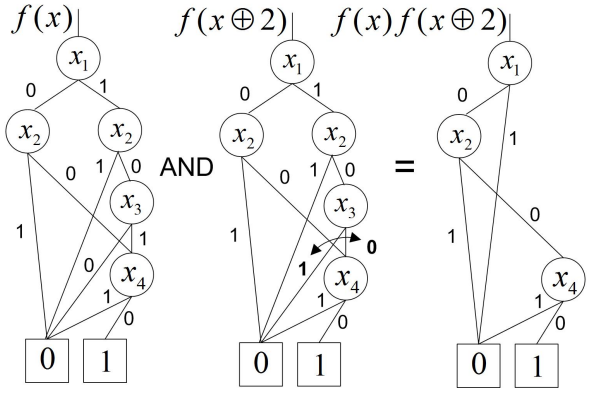
$$\begin{bmatrix} f_i(0 \oplus 0) & \cdots & f_i(0 \oplus 2^n - 1) \\ f_i(1 \oplus 0) & \cdots & f_i(1 \oplus 2^n - 1) \\ \vdots & \vdots & \vdots \\ f_i(2^n - 1 \oplus 0) & \cdots & f_i(2^n - 1 \oplus 2^n - 1) \end{bmatrix} \tag{4.10}$$

Primećuje se da se $B_i(n)$ matrice sastoje od vrsta u kojima su vrednosti funkcije u različitim permutacionim uređenjima. Ova osobina je esencijalna kod SBDD reprezentacije autokorelacionih matrica i ona se višestruko iskorišćava kod izračunavanja totalne autokorelacije preko SBDD-a. Izračunavanja autokorelacionih koeficijenata za zadatu prekidačku funkciju zahteva eksponencijalni broj operacija u odnosu na broj promenljivih funkcije. Međutim kompleksnost izračunavanja preko BDD-a je proporcionalna veličini dijagrama odlučivanja [**?**], [**?**].

Kod prostog BDD metoda [**?**] ili BDD metoda sa permutovanim labelama na granama [**?**], koristi se osobina da je BDD reprezentacija funkcije $f(x \oplus \tau)$ ekvivalentna BDD reprezentaciji funkcije $f(x)$ sa permutovanim labelama na granama za sve čvorove u dijagramu koji odgovaraju promenljivama na pozicijama bita 1 u binarnoj reprezentaciji za $u = (u_1, u_2, \ldots, u_n)$. Izračunavanje autokorelacionih koeficijenata se izvršava kreiranjem jednog BDD-a za originalnu funkciju i jednog za funkciju koja je pomerena za u. U sledećem koraku izvršava se operacija množenja, odnosno AND operacija nad dijagramima $BDD(f(x))$ i $BDD(f(x \oplus u))$, a potom se sumiraju mintermi u rezultujućem dijagramu $BDD(f(x)f(x \oplus u))$. Primer izračunavanja 2. autokorelacionog koeficijent ($u = 0010$) preko BDD metoda sa permutovanim labelama na granama za funkciju definisanu preko vektora istinitosti $F = [1010\,0000\,0010\,0000]^T$ je prikazan na slici 4.6.

Suma svih vrednosti u vektoru istinitosti koji je predstavljen preko $BDD(f(x)f(x \oplus u))$ određuje 2. autokorelacioni koeficijent:

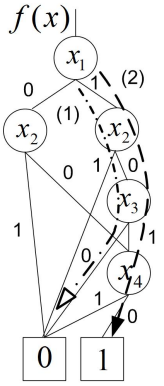
$$\begin{aligned} B(2) \;=\; & 1+0+1+0+0+0+0+0+0+ \\ & +0+0+0+0+0+0+0+0+0 = 2 \end{aligned} \tag{4.11}$$



SLIKA 4.6: Primer izračunavanja 2. autokorelacionog koeficijenta preko BDD metoda sa permutovanim labelama na granama za funkciju definisanu preko vektora istinitosti $F = [1010\,0000\,0010\,0000]^T$

Rekurzivni BDD metod [**?**] ili "in-place" BDD metod [**?**], zahteva kreiranje samo jednog BDD-a. $BDD(f(x))$ i $BDD(f(x \oplus u))$ se razlikuju samo na labelama na granama tako da izračunavanja mogu biti izvršavana preko jednog BDD-a. Kod ovog metoda vrši se reprezentacija $BDD(f(x))$ preko BDD-a koji se potom obilazi na takav način da se izvršava AND operacija između vrednosti terminalnih čvorova $BDD(f(x))$ i $BDD(f(x \oplus u))$ i na kraju se vrši sumiranje tih vrednosti. Obilazak BDD-a se određuje na osnovu pozicija 1 bitova u binarnoj reprezentaciji za $u = (u_1, u_2, \ldots, u_n)$. Više detalja za "in-place" BDD metod je prezentovano u [**?**].

Primer izračunavanja $f(1000)f(1000 \oplus 0010)$ kod izračunavanja 2. autokorelacionog koeficijenta ($u = 0010$) preko "in-place" BDD metode za prekidačku funkciju definisanu vektorom istinitosti $F = [1010\,0000\,0010\,0000]^T$ je prikazan na slici 4.7.



SLIKA 4.7: Primer izračunavanja $f(1000)f(1000 \oplus 0010)$ kod izračunavanja 2. autokorelacionog koeficijenta preko "in-place" BDD metode za prekidačku funkciju definisanu vektorom istinitosti $F = [1010\,0000\,0010\,0000]^T$

Može se primetiti da se izračunavanje $f(1000)f(1000 \oplus 0010)$ preko BDD-a izvršava kao logička operacija između vrednosti terminalnih čvorova koje predstavljaju krajeve puteva označenih sa (1) i (2) na slici 4.7. Put označen sa (1) odgovara vrednosti funkcije

$f(1000)$, dok put označen sa (2) odgovara vrednosti funkcije $f(1000 \oplus 0010)$. Može se takođe primetiti da se putevi (1) i (2) razlikuju samo u različitoj labeli na grani koja odgovara promenljivoj x_3 i koja je određena pozijom bita 1 u binarnoj reprezentaciji vrednosti u. Kompleksnost izračunavanja ove metode je proporcijalna broju čvorova u dijagramu odlučivanja kojim je predstavljena prekidačka funkcija.

Svi algoritmi za izračunavanje autokorelacionih koeficijenata su razvijeni za izračunavanje pojedinačnih koeficijenata jednoizlaznih prekidačkih funkcija. Za dobijanje totalnih autokorelacionih koeficijenata neophodno je dodatno sumiranje izračunatih pojedinačnih koeficijenata. "In-place" BDD algoritam može biti modifikovan za izračunavanje totalnih autokorelacionih koeficijenata preko SBDD-a za višeizlazne prekidačke funkcije. Može se pretpostaviti da će kompaktnija reprezentacija preko SBDD-a ubrzati izračunavanja autokorelacionih koeficijenta. Sa ovom motivacijom, u ovom poglavlju predloženo je proširenje "in-place" BDD metode u SBDD algoritam sa bočnim kretanjem koji je namenjen efikasnom izračunavanju totalnih autokorelacionih koeficijenata za prekidačke funkcije sa mnogo promenljivih i mnogo izlaza.

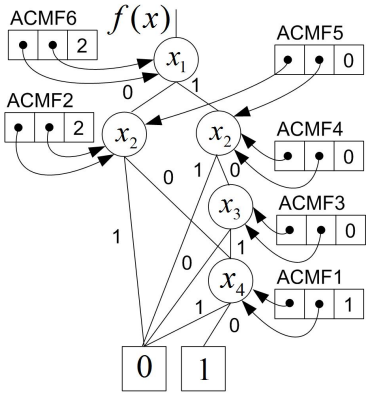
SBDD reprezentacija višeizlaznih prekidačkih funkcija (opisana u 2. poglavlju), jednačina za totalnu autokorelaciju (opisana u 3.5) i osobina da je BDD reprezentacija funkcije ekvivalentna BDD reprezentaciji iste te funkcije sa permutovanim labelama na granama koristi se za kreiranje sledećeg algoritma:

1. Kreira se SBDD za funkciju f u oznaci $SBDD(f(v))$.
2. Za svaki totalni autokorelacioni koeficijent u vrši se markiranje čvorova sa permutovanim labelama na granama u $SBDD(f(v))$ za sve čvorove $x_i, i = 1, \ldots, n$ u pozicijama koje odgovaraju 1 bitovima u binarnoj reprezentaciji za $u = (u_1, u_2, \ldots, u_n)$.
3. Izvršava se množenje $SBDD(f(v))$ sa $SBDD(f(v \oplus u))$ korišćenjem rekurzivnog obilaska inicijalnog SBDD-a i istog tog SBDD-a sa permutovanim labelama na granama gde se permutovane labele određuju markiranjem čvorova u prethodnom koraku. Rekurzivnim obilaskom poddijagrama u SBDD-u se dobija suma rezultata množenja SBDD-a i SBDD-a sa permutovanim labelama na granama što daje vrednost totalnog autokorelacionog koeficijenta.

Efikasnost izračunavanja totalnih autokorelacionih koeficijenata preko SBDD-a omogućena je zahvaljujući rekurzivnom obilasku dijagrama koji iskorišćava postojanje visokog nivoa rendundanse između izlaza funkcije (ukoliko ona zaista postoji). Na osnovu efikasnih tehnika za programiranje jezgra BDD paketa, kao što je predstavljeno na primer u [**?**], uvodi se dodatna heš tabela za ubrzanje rekurzivnih obilazaka SBDD-a (3. korak u prethodno definisanom algoritmu). Ideja se zasniva na osnovu razmatranja da

memorijske funkcije za rekurzivni ITE algoritam koriste heš tablicu kako bi smanjile obim korišćenja operativne memorije. U slučaju ovog algoritma, memorijske funkcije pamte rezulzate autokorelacija između dva poddijagrama u SBDD-u i one se čuvaju u posebnoj heš tabeli. Ove memorijske funkcije se predstavljaju pomoću uređene trojke $(original, shifted, result)$, gde oznaka $result$ označava celobrojni rezultat autokorelacije izvršene nad dva ulazna argumenta $original$ (poddijagram $SBDD(f(x))$) i $shifted$ (poddijagram $SBDD(f(x \oplus \tau))$). Svaka jedinstvena operacija između dva poddijagrama u SBDD-u se unosi u kreiranu heš tabelu. Pre izračunavanja nove vrednosti autokorelacije između dva poddijagrama u SBDD-u, u heš tabeli se proverava da li rezultat autokorelacije između ova dva poddijagrama već postoji. Ako postoji, umesto izračunavanja rezultata koristi se vrednost koja je preuzeta iz heš tabele.

Primer memorijske funkcije za rezultate autokorelacije je prikazan na slici 4.8. Zbog lakšeg razumevanja prikazane su memorijske funkcije za BDD jednoizlazne prekidačke funkcije.



Slika 4.8: Primer memoriske funkcije za rezultate kod izračunavanja 2. autokorelacionog koeficijenta preko SBDD algoritma sa bočnim kretanjem za jednoizlaznu prekidačku funkciju definisanu preko vektora istinitosti $F = [1010\,0000\,0010\,0000]^T$

Može se primetiti da autokorelaciona memorijska funkcija, u oznaci ACMF ima argumente $original$ i $shifted$ u kojima se čuvaju adrese na polazne čvorove poddijagrama u BDD-u za koje se izračunava autokorelacija. Na primer, autokorelaciona memorijska funkcija $ACMF1$ označava da vrednost autokorelacije za poddijagrame koji odgovaraju promenljivoj x_4 je 1 i ona poništava pokušaj ponovljenog izračunavanja autokorelacije za te poddijagrame. Takođe se može primetiti da nenulte vrednosti autokorelacione operacije za poddijagrame odgovaraju postojećim čvorovima na rezultujućem BDD-u na slici 4.6. Generalno, kompleksnost ovog algoritma je proporcijalna broju čvorova u BDD-u za jednoizlazne funkcije ili u SBDD-u za višeizlazne funkcije.

Opis implementacije ovog algoritma razmatran je i diskutovan u narednom poglavlju ove disertacije. U cilju ispitivanja efikasnosti predloženog algoritma, u slučaju ograničenih

raspoloživih memorijskih resursa, izvršena je implementacija i eksperimentalno ispitivanje ovog algoritma što je detaljno opisano u narednim poglavljima ovog rada. Prednosti i mane ovog algoritma u slučaju izračunavanja za prekidačke funkcije sa velikim brojem promenljivih su takođe diskutovane.

4.1.3 Ocena složenosti SBDD algoritama sa permutovanim labelama i SBDD algoritma sa bočnim kretanjem

Izračunavanje autokorelacionih koeficijenata preko SBDD algoritma sa permutovanim labelama na granama vrši prvo generisanje dva dijagrama odlučivanja u memoriji računara: $SBDD(f(x))$ i $SBDD(f(x \oplus u))$, a potom na osnovu nih generiše dijagram $SBDD(f(x)f(x \oplus u))$. Broj čvorova u dijagramima $SBDD(f(x))$ i $SBDD(f(x \oplus u))$ su isti, tako da je prostorna kompleksnost ovog algoritma izražena sa $O(2{*}size(SBDD(f)))+$ $O(size(SBDD(f(x)f(x \oplus u))))$, gde funkcija $size$ označava ukupan broj čvorova u dijagramu. Budući da se za izračunavanje svakog autokorelacionog koeficijenta izvršava množenje terminalnog čvora u dijagramu $SBDD(f(x))$ i terminalnog čvora u dijagramu $SBDD(f(x \oplus u))$ i odgovarajuće sabiranje tih terminalnih čvorova, broj množenja i broj sabiranja ima kompleksnost $O(size(SBDD(f)))$. Zbog toga ukupna kompleksnost SBDD algoritma sa permutovanim labelama na granama je $O(size(SBDD(f)))$. S obzirom da je ovaj algoritam namenjen za izračunavanje pojedinačnih autokorelacionih koeficijenata, kod izračunavanja različitih koeficijenata se vrši samo korekcija pokazivača na granama koje treba da se permutuju. Zbog toga SBDD algoritam sa permutovanim labelama na granama kod izračunavanja bilo kog koeficijenta ima kompleksnost $O(size(SBDD(f)))$.

Izračunavanje autokorelacionih koeficijenata preko SBDD algoritma sa bočnim kretanjem vrši generisanje samo jednog dijagrama odlučivanja u memoriji računara $SBDD(f(x))$. Dijagram $SBDD(f(x \oplus u))$ se određuje korekcijom prilikom obilaska dijagrama $SBDD(f(x))$, a potom se rekurzivnim paralelnim obilaskom $SBDD(f(x))$ određuje $f(x)f(x \oplus u)$. Prostorna kompleksnost ovog algoritma izražena je sa $O(size(SBDD(f)))$. Budući da se za izračunavanje svakog autokorelacionog koeficijenta izvršava množenje terminalnog čvora u dijagramu $SBDD(f(x))$ i terminalnog čvora u istom dijagramu i odgovarajuće sabiranje tih terminalnih čvorova, broj množenja i broj sabiranja ima kompleksnost $O(size(SBDD(f)))$. Zbog toga ukupna kompleksnost SBDD algoritma sa bočnim kretanjem je $O(size(SBDD(f)))$. S obzirom da je ovaj algoritam namenjen za izračunavanje pojedinačnih autokorelacionih koeficijenata, kod izračunavanja različitih koeficijenata se vrši samo korekcija obilazaka istog dijagrama. Zbog toga SBDD algoritam sa bočnim kretanjem kod izračunavanja bilo kog koeficijenta ima kompleksnost $O(size(SBDD(f)))$.

4.2 Algoritmi za izračunavanje kompletnih autokorelacionih koeficijenata preko dijagrama odlučivanja

Efikasnost izračunavanja kompletnih autokorelacionih koeficijenata je direktno određena vremenom potrebnim za njihovo izračunavanje i kompleksnošću odgovarajućih struktura podataka koje se koriste kod reprezentacije ulaza i izlaza funkcije [?], [?], [?], [?]. Kompletni autokorelacioni koeficijenti ili autokorelacioni spektar prekidačke funkcije od n promenljivih se predstavlja preko vektora dužine 2^n. Zbog toga metodi za izračunavanje kompletnih autokorelacionih koeficijenata imaju eksponencijalnu vremensku i prostornu kompleksnost u zavisnosti od broja promenljivih funkcije. Postoje brojni algoritmi za izračunavanje autokorelacionih koeficijenata za zadatu prekidačku funkciju u zavisnosti od strukture podatka koja se koristi za memorisanje funkcije i njenih autokorelacionih koeficijenata.

U prethodnim sekcijama su predstavljeni algoritmi za efikasno izračunavanje pojedinačnih totalnih autokorelacionih koeficijenata. Ovim algoritmima se mogu izračunati i kompletni autokorelacioni koeficijenti ali je prostorna i vremenska kompleksnost izračunavanja veoma velika [?], [?], [?]. Nekoliko algoritama za efikasno izračunavanje autokorelacionih koeficijenata je predstavljeno u prethodnom poglavlju. Međutim svi postojeći algoritmi za izračunavanje ovih koeficijenata su u praksi primenljivi samo za funkcije sa malim brojem promenljivih [?], [?], [?]. Posebnu grupu algoritama čine oni zasnovani na postupcima izračunavanja autokorelacije na osnovu Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije na odgovarajućim algebarskim strukturama [?]. Efikasnost izračunanja kompletne totalne autokorelacije se može povećati korišćenjem mogućnosti da se Walsh-ov spektar može izračunati preko brze Walshove transformacije i dijagrama odlučivanja.

4.2.1 SMTBDD algoritam preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije

Kompletni autokorelacioni koeficijenti ili autokorelacioni spektar prekidačke funkcije može biti izračunat iz Walsh-ovih spektralnih koeficijenata prekidačke funkcije korišćenjem Wiener-Khinchin-ove teoreme i brzog algoritma za Walsh-ovu transformaciju preko multi-terminalnog BDD-a. Ovaj algoritam je teorijski razmatran u radu [?]. Njegova implementacija i eksperimentalno razmatranje izvršeno je u radovima [?], [?], [?]. Predloženi algoritam se takođe može upotrebiti kod izračunavanja autokorelacionih koeficijenata karakterističnih funkcija višeizlaznih prekidačkih funkcija [?], [?] i [?]. Ovaj

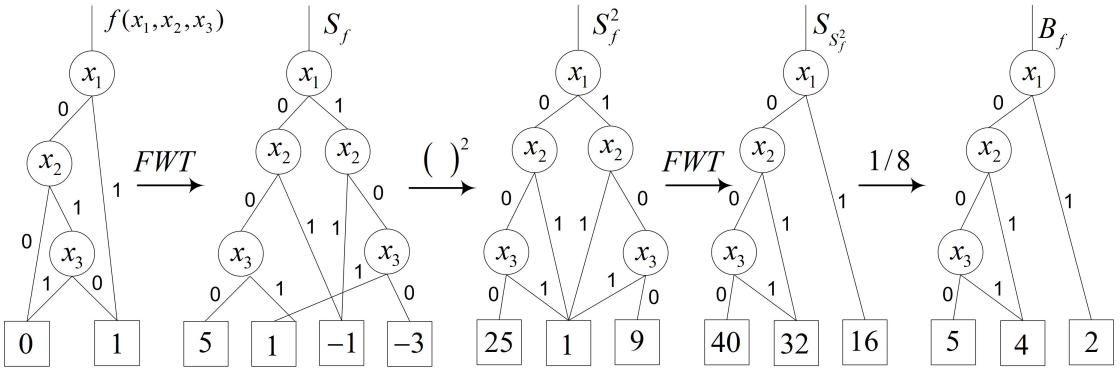
algoritam se može proširiti na izračunavanje autokorelacione transformacije za višeizlazne prekidačke funkcije što je predstavljeno u ovom radu. S obzirom da ovaj algoritam izračunavanja generiše veoma velike celobrojne vrednosti (vrednosti do 2^{2n}, gde je n broj promenljivih funkcije), trenutno dostupne tehnike implementacije ograničavaju primenu algoritma za prekidačke funkcije sa ne više od 32 promenljive. Zbog toga je u narednoj sekciji ovog rada predstavljeno proširenje ovog algoritma sa izračunavanjima preko BDD paketa sa dinamičkim terminalnim čvorovima.

U ovom poglavlju predstavljen je algoritam za izračunavanje kompletnih totalnih autokorelacionih koeficijenata preko razdeljenih MTBDD-ova za višeizlazne prekidačke funkcije sa ne više od 32 promenljive. Izračunavanje se izvršava u spektralnom domenu gde se koristi Wiener-Khinchin-ova teorema i brzi algoritam za spektralnu transformaciju preko SMTBDD-a. Izračunavanje Walsh-ovog spektra preko grafa toka operacija koji opisuje brzu Walsh-ovu transformaciju (FWT) je detaljno opisano u 2. poglavlju.

SMTBDD reprezentacija višeizlaznih prekidačkih funkcija (opisana u 2. poglavlju), jednačina za totalnu autokorelaciju (opisana u 3.5) i mogućnost za izračunavanje Walsh-ovog spektra preko brze Walsh-ove transformacije i dijagrama odlučivanja (opisana u 2. poglavlju) koristi se za kreiranje sledećeg algoritma:

1. Za višeizlaznu prekidačku funkciju $f(v)$ kreira se SMTBDD za funkciju f, u oznaci $SMTBDD(f(v))$.
2. Vrši se konverzija $SMTBDD(f(v))$ u $SMTBDD(S_f)$, gde S_f označava Walsh-ov spektar funkcije f.
3. Vrši se množenje $SMTBDD(S_f)$ samim sobom korišćenjem standardne procedure za množenje prekidačkih funkcija predstavljenih dijagramima odlučivanja (na primer u [**?**]).
4. Vrši se konverzija $SMTBDD({S_f}^2)$ u $SMTBDD(S_{{S_f}^2})$.
5. Vrši se normalizacija celobrojnih vrednosti u terminalnim čvorovima u $SMTBDD(S_{{S_f}^2})$ i generisanje rezultujućeg dijagrama $SMTBDD(B_f)$, s obzirom da je Walsh-ova matrica samoinverzna do konstante 2^n, gde je n broj promenljivih u funkciji f.
6. Vrši se sumiranje odgovarajućih vrednosti u terminalnim čvorovima u $SMTBDD(S_{{S_f}^2})$ kako bi se generisali totalni autokorelacioni koeficijenti.

Na primer, izračunavanje kompletnih autokorelacionih koeficijenta korišćenjem Wiener-Khinchin-ove teoreme i brze Walshove transformacije preko MTBDD-a jednoizlazne prekidačke funkcije definisane sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$, čiji je vektor istinitosti $F = [00101111]^T$, prikazano je na slici 4.9.

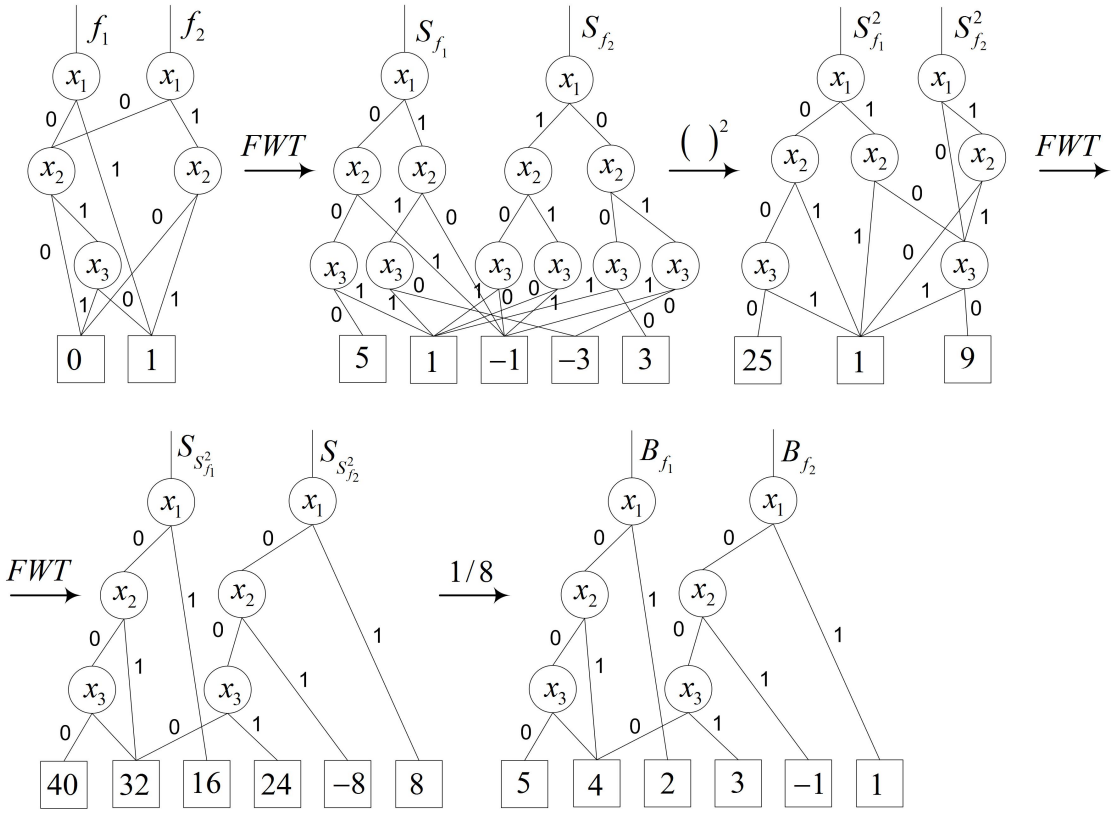


SLIKA 4.9: Primer izračunavanja kompletnih autokorelacionih koeficijenata korišćenjem Wiener-Khinchin-ove teoreme i brze Walshove transformacije preko MTBDD-a za funkciju definisanu sa $f(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$, čiji je vektor istinitosti $F = [00101111]^T$.

Prvi korak kod izračunavanja autokorelacije odnosi se na izračunavanje Walsh-ovog spektra $S_f = [5, 1, -1, -1, -3, 1, -1, -1]^T$ predstavljenim preko MTBDD korišćenjem algoritma za izračunavanje Walsh-ovog spektra preko brze Walsh-ove transformacije i MTBDD-a. U drugom koraku, vektor spektralnih koeficijenata ${S_f}^2 = [25, 1, 1, 1, 9, 1, 1, 1]^T$ koji se dobija kvadriranjem Walsh-ovog spektra $S_f = [5, 1, -1, -1, -3, 1, -1, -1]^T$ predstavljen je takođe preko MTBDD-a, gde se rezultujući MTBDD dobija množenjem MTBDD-a iz prethodnog koraka samim sobom. Očigledno je da broj terminalnih čvorova znatno manji od broja vrednosti u vektoru ${S_f}^2$. Korak 3 dovodi do autokorelacionog spektra predstavljenog preko MTBDD-a korišćenjem brze Walsh-ove transformacije kvadriranog Walsh-ovog spektra predstavljenog preko MTBDD-a. Poslednji korak se koristi za množenje autokorelacionog spektra, predstavljenog preko MTBDD-a, sa faktorom normalizacije (u slučaju funkcije od 3 promenljive vrednsot faktora je 1/8).

Ova tehnika u slučaju višeizlaznih prekidačkih funkcija korišćenjem SMTBDD-a može da bude efikasnija u odnosu na izračunavanja preko pojedinačnih MTBDD-ova. Na primer, izračunavanje kompletnih autokorelacionih koeficijenta korišćenjem Wiener-Khinchin-ove teoreme i brze Walshove transformacije preko SMTBDD-a višeizlazne prekidačke funkcije definisane sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$, čiji su vektori istinitosti $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$, je prikazan je na slici 4.10.

Evidentno je da je izračunavanje i predstavljanje kompletnih autokorelacionih koeficijenta iz prethodnog primera preko SMTBDD kompaktno, sobzorm da u MTBDD-ima kojima su predstavljeni Walsh-ovi spektri, kvadrirani Walsh-ovi spektri i autokorelacioni spektri postoje deljivi poddijagrami. Ova osobina je esencijalna kod efikasnog predstavljanja prekidačkih funkcija preko SMTBDD-a, kao i kod raznih izračunavanja preko SMTBDD-a. Očigledno je da je broj "leptir" operacija u SMTBDD-u mnogo manji u odnosu na broj "leptir" operacija u dva odvojena MTBDD-a za višeizlaznu prekidačku funkciju definisanu



SLIKA 4.10: Primer SMTBDD algoritma preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije višeizlazne prekidačke funkcije definisane sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$, čiji su vektori istinitosti $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$.

funkcijama f_1 i f_2, s obzirom da postoje deljivi poddijagrami u procesu izračunavanja preko MTBDD-a za funkcije f_1 i f_2.

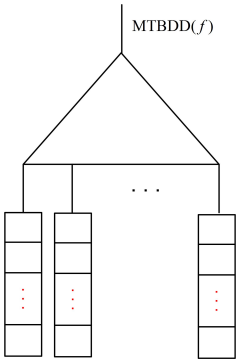
Opis implementacije ovog algoritma razmatran je i diskutovan u narednom poglavlju ove disertacije. Takođe su u jednom od narednih poglavlja prikazani i eksperimentalni rezultati testiranja efikasnosti predloženog algoritma na skupu benchmark prekidačkih funkcija.

4.2.2 SMTBDD algoritam preko BDD paketa sa dinamičkim terminalnim čvorovima

Iz jednačine za izračunavanje autokorelacionog koeficijenta 3.2 evidentno je da maksimalna vrednost koeficijenta koji se formira iz sume 2^n Bool-ovih vrednosti može da bude 2^n, gde je n broj promenljivih prekidačke funkcije. U skladu sa tim iz jednačine 3.6 koja opisuje Wiener-Khinchin-ovu teoremu takođe je evidentno da maksimalna vrednost autokorelacionog koeficijenta pre množenja sa faktorom normalizacije 2^{-n} može da bude 2^{2n}. Na primer, maksimalna vrednost autokorelacionog koeficijenta prekidačke funkcije

od 65 promenljivih može da bude $2^{130} = 1,36 * 10^{39}$. Zbog toga izračunavanje autokorelacionog koeficijenta na računaru za prekidačke funkcije sa velikim brojem promenljivih zahteva rad sa posebnim tipovima podataka koji podržavaju rad sa velikim celobrojnim vrednostima.

U slučaju da se autokorelacioni koeficijenti izračunavaju SMTBDD algoritmom preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije maksimalna vrednost terminalnih čvorova u SMTBDD-u, pre množenja sa faktorom normalizacije od 2^{-n}, može da bude 2^{2n}, gde je n broj promenljivih prekidačke funkcije. Zbog toga, za izračunavanja preko SMTBDD-a, korišćenje klasičnih dijagrama odlučivanja u okviru BDD paketa sa 32-bitnim ili 64-bitnim celobrojnim terminalnim čvorovima ograničava izračunavanja za prekidačke funkcije sa relativno malim brojem promenljivih (najviše 16 ili 32 promenljivih, respektivno). Sa motivacijom za uklanjanje ovog ograničenja, u ovom radu je predložen novi tip dijagrama odlučivanja, MTBDD sa dinamički redimenzionisanim terminalnim čvorovima, koji dozvoljava izračunavanje autokorelacionih koeficijenata za prekidačke funkcije sa velikim brojem promenljivih. Na slici 4.11 je prikazana generalna struktura MTBDD-a sa dinamički redimenzionisanim terminalnim čvorovima. Ovaj koncept se veoma jednostavno može proširiti i na SMTBDD.



Slika 4.11: Generalna struktura MTBDD-a sa dinamički redimenzionisanim terminalnim čvorovima.

Predložena struktura podataka se zasniva na ideji prezentovanoj u radu [**?**], gde se dijagrami odlučivanja sa celobrojnim terminalnim čvorovima posmatraju kao binarni vektori reprezentacija celobrojnih vrednosti. Ti binarni vektori se mogu proširiti i upotrebiti za reprezentaciju kao binarna reprezentacija velikih celobrojnih vrednosti. Veličine tih binarnih vektora u terminalnim čvorovima mogu biti modifikovane u skladu sa zahtevima i predstavljene preko dinamički redimenzionisanih terminalnih čvorova. Dinamički redimenzionisani terminalni čvorovi kod dijagrama odlučivanja nisu standardno uključeni u BDD pakete. U tom smislu neophodno je razviti specijalizovan BDD paket sa dinamičkim terminalnim čvorovima što predstavlja proširenje standardnih BDD tehnika. Opis implementacije BDD paketa sa dinamičkim terminalnim čvorovima je razmatran i diskuztovan u narednom poglavlju ove disertacije.

U ovom poglavlju predstavljen je novi algoritam za izračunavanje kompletnih totalnih autokorelacionih koeficijenata preko razdeljenih MTBDD-a za višeizlazne prekidačke funkcije sa više od 16 ili 32 promenljive u zavisnosti od tipa BDD paketa kojim se implementira rad sa dijagramima. Osnovni principi koji se koriste kod implementacije standardnih BDD paketa opisani su u narednom poglavlju. Treba napomenuti da BDD paketi korišćenjem trenutnih računarskih tehnologija mogu podržati rad sa 32-bitnim ili 64-bitnim celobrojnim vrednostima. Izračunavanje se izvršava u spektralnom domenu gde se koristi Wiener-Khinchinova teorema i brzi algoritam za spektralnu transformaciju preko SMTBDD-a sa dinamički redimenzionisanim terminalnim čvorova. Ovaj algoritam je veoma sličan prethodno definisanom algoritmu za izračunavanje kompletnih totalnih autokorelacionih koeficijenata preko razdeljenih MTBDD-a (opisanog u prethodnom poglavlju) sa razlikom u samo jednom dodatnom koraku. Izračunavanje Walsh-ovog spektra preko grafa toka operacija koji opisuje brzu Walsh-ovu transformaciju (FWT) je detaljno opisano u 2. poglavlju.

Implemntacija BDD paketa sa dinamičkim terminalnim čvorovima (opisana u sledećem poglavlju), jednačina za totalnu autokorelaciju (opisana u 3.5) i mogućnost za izračunavanje Walsh-ovog spektra preko brze Walsh-ove transformacije i dijagrama odlučivanja koristi se za kreiranje sledećeg algoritma:

1. Za višeizlaznu prekidačku funkciju $f(v)$ vrši se inicijalizacija SMTBDD-a sa dinamičkim terminalnim čvorovima, gde se dužina terminala postavlja na vrednost 2^{2n}, gde je n broj promenljivih prekidačke funkcije.

2. Za višeizlaznu prekidačku funkciju $f(v)$ kreira se SMTBDD za funkciju f, u oznaci $SMTBDD(f(v))$.

3. Vrši se konverzija $SMTBDD(f(v))$ u $SMTBDD(S_f)$, gde S_f označava Walsh-ov spektar funkcije f.

4. Vrši se množenje $SMTBDD(S_f)$ samim sobom korišćenjem standardne procedure za množenje prekidačkih funkcija predstavljenih dijagramima odlučivanja (na primer u [**?**]).

5. Vrši se konverzija $SMTBDD({S_f}^2)$ u $SMTBDD(S_{{S_f}^2})$.

6. Vrši se normalizacija terminalnih čvorova u $SMTBDD(S_{{S_f}^2})$ i generiše rezultujući dijagram $SMTBDD(B_f)$, s obzirom da je Walsh-ova matrica samo-inverzna do konstante 2^n, gde je n broj promenljivih u funkciji f.

7. Vrši se sumiranje odgovarajućih vrednosti u terminalnim čvorovima u $SMTBDD(S_{{S_f}^2})$ kako bi se generisali totalni autokorelacioni koeficijenti.

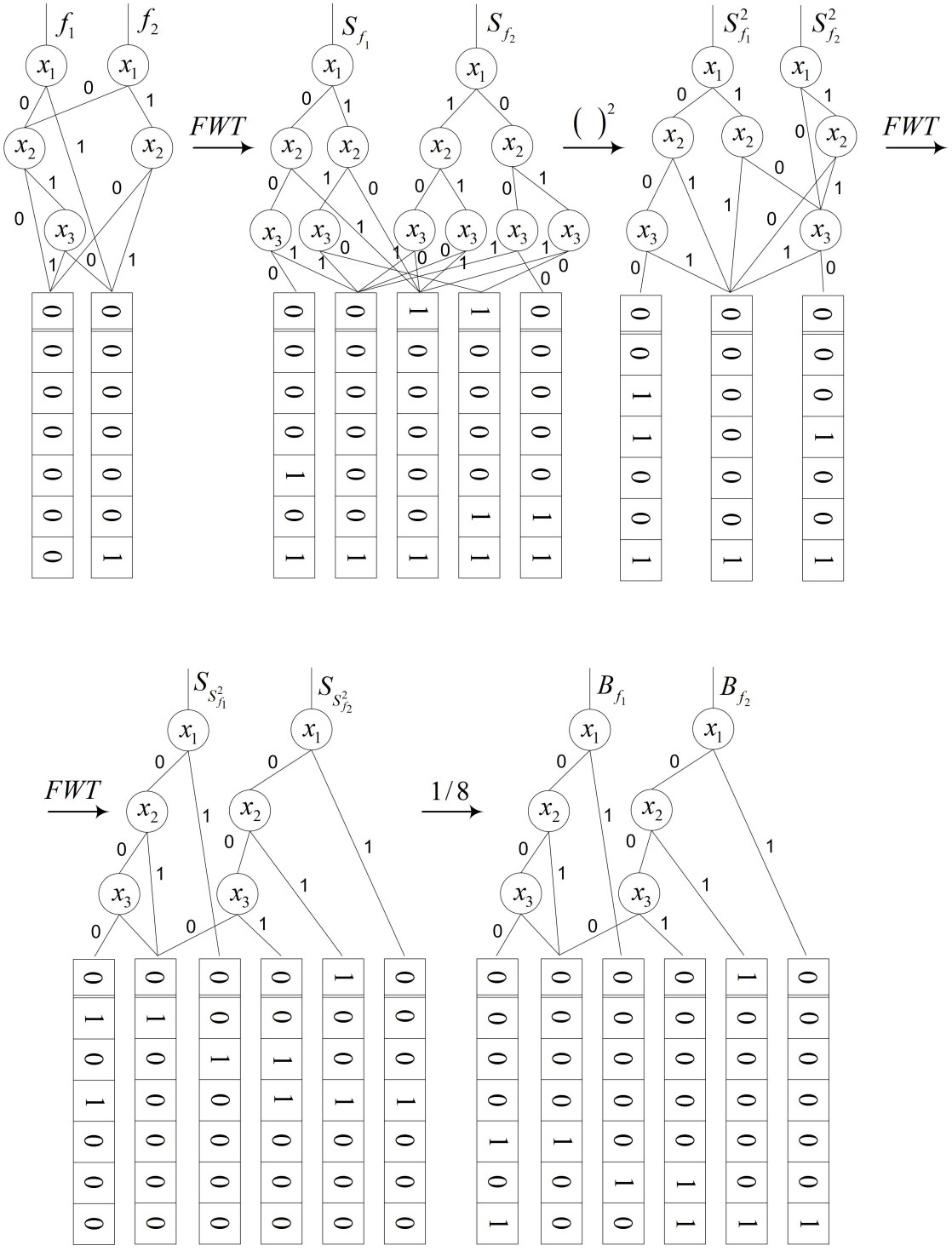
Na primer, izračunavanje kompletnih autokorelacionih koeficijenta korišćenjem Wiener-Khinchin-ove teoreme i brze Walshove transformacije preko SMTBDD-a sa dinamičkim terminalnim čvorovima višeizlazne prekidačke funkcije definisane sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$, čiji su vektori istinitosti $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$, je prikazano na slici 4.12. U ovom primeru se za višeizlaznu prekidačku funkciju $f(f_1, f_2)$ sa 3 promenljive vrši inicijalizacija SMTBDD-a sa dinamičkim terminalnim čvorovima, gde se maksimalna dužina dinamičkih terminalnih čvorova postavlja na vrednost 2^{2*3}, odnosno dužina vektora kojim se predstavlja dinamički terminalni čvor se postavlja na $2*3 = 6$. Vektoru kojim se predstavlja terminalni čvor se pridružuje i jedan dodatni element za predstavljanje znaka celobrojne vrednosti. Može se primetiti da dužina vektora od 6 elemenata (i jedan dodatni element za znak) kojim se predstavlja dinamički terminalni čvor omogućava smeštanje binarne reprezentacije maksimalne vrednosti terminalnog čvora (vrednost 40 u 4. SMTBDD-u na slici 4.12).

Evidentno je da izračunavanje i predstavljanje kompletnih autokorelacionih koeficijenta iz prethodnog primera preko SMTBDD-a sa dinamičkim terminalnim čvorovima za prekidačke funkcije sa malim brojem promenljivih (manje od 16 ili 32 promenljivih) nije efikasno sobzorm da se celobrojne vrednosti terminala u SMTBDD-u predstavljaju preko nizova. Takođe su neefikasne i osnovne aritmetičke i logičke operacije nad celobrojnim vrednostima predstavljenih preko nizova. Međutim ovakav način predstavljanja terminala uklanja ograničenje prethodno definisanog algoritma (opisanog u prethodnoj sekciji) i omogućava izračunavanja za prekidačke funkcije sa velikim brojem promenljivih (više od 16 ili 32 promenljivih). Efikasnost izračunavanja korišćenjem Wiener-Khinchin-ove teoreme i brze Walshove transformacije preko SMTBDD-a opisana je u prethodnom poglavlju.

Opis implementacije ovog algoritma razmatran je i diskutovan u narednom poglavlju ove disertacije. Takođe su u jednom od narednih poglavlja prikazani i eksperimentalni rezultati testiranja efikasnosti predloženog algoritma na skupu benchmark prekidačkih funkcija.

4.2.3 Ocena složenosti SMTBDD algoritama preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije i SMTBDD algoritma preko BDD paketa sa dinamičkim terminalnim čvorovima

Izračunavanje autokorelacionih koeficijenata preko SMTBDD algoritma preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije vrši generisanje četiri dijagrama odlučivanja u memoriji računara: $SMTBDD(f)$, $SMTBDD(S_f)$, $SMTBDD({S_f}^2)$ i



Slika 4.12: Primer SMTBDD algoritma sa dinamičkim terminalnim čvorovima višeizlazne prekidačke funkcije definisane sa $f_1(x_1, x_2, x_3) = x_1 + x_2\overline{x}_3$ i $f_2(x_1, x_2, x_3) = x_1x_2 + x_2\overline{x}_3$, čiji su vektori istinitosti $F_1 = [00101111]^T$ i $F_2 = [00100011]^T$.

$SMTBDD(B_f)$. Broj čvorova u ovim dijagramima su različiti, tako da je prostorna kompleksnost ovog algoritma izražena sa $O(size(SMTBDD(f)))+O(size(SMTBDD(S_f)))+O(size(SMTBDD({S_f}^2)))+O(size(SMTBDD(B_f)))$, gde funkcija *size* označava ukupan broj čvorova u dijagramu. Budući da se za izračunavanje kompletnih autokorelacionih koeficijenta izvršavaju operacije množenja, sabiranja i oduzimanja u neterminalnim čvorovima u sva četiri dijagrama, broj množenja i broj sabiranja ima kompleksnost $O(size(SBDD(F)))$, što važi za sve dijagrame. Zbog toga ukupna kompleksnost SMTBDD algoritma preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije je $O(size(SMTBDD(f)))+O(size(SMTBDD(S_f)))+O(size(SMTBDD({S_f}^2)))+O(size(SMTBDD(B_f)))$.

Izračunavanje autokorelacionih koeficijenata preko SMTBDD algoritam preko BDD paketa sa dinamičkim terminalnim čvorovima ima iste korake kao i SMTBDD algoritma preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije. Razlika ovih algoritama je u predstavljanju terminalnih čvorova u dijagramima, gde SMTBDD algoritam preko BDD paketa sa dinamičkim terminalnim čvorovima korisiti nizove definisane dužine u svakom terminalnom čvoru. Zbog toga prostorna kompleksnost ovog algoritma izražena je sa $O(size(SMTBDD(f))+sizet(SMTBDD(f)))+O(size(SMTBDD(S_f))+sizet(SMTBDD(S_f)))+O(size(SMTBDD({S_f}^2))+sizet(SMTBDD({S_f}^2))+O(size(SMTBDD(B_f))+sizet(SMTBDD(B_f))$, gde funkcija *sizet* označava ukupan broj terminalnih čvorova u dijagramu pomnožen sa brojem elemenata niza kojim se pamti vrednost terminala. Budući da se za izračunavanje kompletnih autokorelacionih koeficijenta izvršavaju operacije množenja, sabiranja i oduzimanja u neterminalnim čvorovima gde se prilkom svake operacije pristupa nizovima kojima su predstavljeni terminali, operacije u dijagramu imaju kompleksnost $O(size(SBDD(F))*sizet(SBDD(F)))$, što važi za sve dijagrame. Zbog toga je ukupna kompleksnost SMTBDD algoritma preko BDD paketa sa dinamičkim terminalnim čvorovima $O(size(SMTBDD(f))*sizet(SMTBDD(f)))$
$+O(size(SMTBDD(S_f))*sizet(SMTBDD(S_f)))$
$+O(size(SMTBDD({S_f}^2))*sizet(SMTBDD({S_f}^2))$
$+O(size(SMTBDD(B_f))*sizet(SMTBDD(B_f))$.

Poglavlje 5

Implementacija algoritama za izračunavanje autokorelacionih koeficijenata

U ovom poglavlju biće opisane implementacije algoritama za izračunavanje autokorelacionih koeficijenata preko dijagrama odlučivanja, koji su predloženi u ovom radu. Svi predloženi algoritmi su u prethodnom poglavlju detaljno opisani i ilustrovani na primerima višeizlaznih prekidačkih funkcija. Implementacije ovih algoritama predstavljaju nadgradnju implementacije BDD paketa. Zbog toga su u ovom poglavlju opisani i osnovni principi koji se koriste kod implementacije standardnih BDD paketa (implementacija čvorova dijagrama, jedinstvene tablice, tablice operacija, sakupljača otpada i operacija nad dijagramima odlučivanja).

5.1 BDD paketi

U prethodnim poglavljima diskutovane su razne vrste dijagrama odlučivanja i njihove osobine iz različitih aspekata. Međutim, kako bi oni mogli biti primenjivani u praksi, dijagrami odlučivanja moraju da budu implementirani u nekom programskom jeziku. Programske implementacije dijagrama odlučivanja, takozvani BDD paketi, moraju da obezbede sve njihove osobine u različitim kontekstima.

Binarni dijagrami odlučivanja su postali dominantna struktura podatka za predstavljanje i manipulaciju prekidačkih i višeznačnih logičkih funkcija u CAD aplikacijama [**?**]. Oni se široko koriste u različitim oblastima CAD-a: logička sinteza, testiranje, simulacija,

projektovanje i verifikacija [?] [?], [?]. Različiti BDD algoritmi se obično kreiraju kao nadogradnja BDD paketa.

U poslednjih nekoliko godina rayvijeni su mnogi BDD paketi koji predstavljaju interfejs za manipulaciju prekidačkih funkcija. Neki od ovih paketa su kreirani u okviru akademskih institucija, dok neki u okviru industrijskih razvojnih centara. Komercijalni BDD paketi su u većem broju slučaja restriktivni i nisu javno dostupni. Međutim, s obzirom da je razvoj BDD paketa većim delom vođen u okviru akademskih intitucija, javno dostupni akademski BDD paketi obezbeđuju sagledavanje tehnologija koje se koriste u paketima. Većina implementacija BDD paketa je kreirana u programskim jezicima C, C++, Lisp i Java [?], [?], [?], [?].

Prva efikasna implementacija BDD paketa je bio programski sistem pod nazivom "OBDD package" koji je razvijen na "Carnegie Mellon" univerzitetu [?]. Ovaj paket postao je javno dostupan od 1990. godine. Paket je razvijen sa ciljem proširenja granica primenljivosti softvera za verifikaciju pod nazivom "Tranalyze". Paket "Tranalyze" se koristio za verifikaciju tranzistorskih kola u MOS (Metal-Oxide-Semiconductor) tehnologiji na prekidačkom nivou. Ovde se tranzistori na apstraktnom nivou modeluju prekidačima [?]. Mnoge implementacione tehnike opisane u ovom poglavlju su inicijalno razvijene i primenjene u BDD paketu "OBDD package". Efikasnost implementacionih tehnika predstavljenih u "OBDD package" paketu je potvrđena činjenicom da ove tehnike još uvek koriste svi savremeni BDD paketi. Iskustvo za kreiranje "OBDD package" paketa upotrebljeno je par godina kasnije za kreiranje Long-ovog OBDD paketa, takođe na "Carnegie Mellon" univerzitetu [?]. Ovaj paket postao je javno dostupan 1993. godine. Paket je bio integrisan u SIS [?] programski sistem za sintezu sekvencijalnih logičkih kola koji je razvijan na Univerzitetu Kalifornija u Berkliju. Najvažnija novina u Long-ovom BDD paketu je bila uvođenje implementacione tehnike za dinamičko određivanje "optimalnog" uređenja promenljivih u BDD-u. Sledeći značajniji napredak u razvoju implementacionih tehnika BDD paketa doneo je "CUDD" paket (eng. CUDD - California University Decision Diagram) [?], koji je razvijen na Univerzitetu Kalifornija u Boulderu 1996. godine. U ovom paketu su pored poboljšanja performansi prethodnih paketa razvijena i kolekcija algoritama za poboljšanje "optimalnog" uređenja promenljivih u BDD-u. Osim ovih algoritama u CUDD paket je ugrađena podrška za druge tipove dijagrama odlučivanja MTBDD i ZBDD (eng. ZBDD - Zero-suppressed Decision Diagram) [?]. Osim toga kod ovog paketa iz razloga efikasnosti je uveden brojač referenci na novokreirane čvorove dijagrama odlučivanja. Na ovaj način se efikasnije može izbeći proces referenciranja i dereferenciranja za međurezultate koji su potrebni veoma kratko u procesu izvršavanja operacija nad dijagramima. Neke implementacione tehnike opisane u ovom poglavlju se baziraju na tehnikama korišćenih u CUDD BDD paketu.

Većina BDD paketa su besplatno dostupni u javnom domenu na internetu. Paketi CAL (University of California, Berkeley) [**?**], CMU (ATT, USA), i CUDD su postali popularni zbog njihovog prisustva u programskim sistemima SIS i VIS [**?**]. BuDDy [**?**] (IT University of Copenhagen) BDD paket je zbog svoje jednostavnosti postao veoma popularan u naučnim krugovima. Paketi IBM [**?**] (IBM Watson, USA), Tiger [**?**] (Bull/DEC/Xorix, USA) i MONA [**?**] (ATT/BRICS, USA) su popularni u komercijalnim krugovima i zaštićeni su autorskim pravima. BDD paket TUD (Darmstadt, Germany) je interesantan jer podržava rad sa različitim tipovima dijagrama odlučivanja (BDD, FDD, KFDD, KDD, MTBDD, EVBDD, BMD, HDD i ZDD [**?**]). Veliki broj različitih tipova dijagrama odlučivanja podržava i BDD paket razvijen od strane naučnika M. Fujita i S. Minato [**?**]. Izbor BDD paketa može da se posmatra iz sledećih aspekata: funkcionalnost, korisnički interfejs, robusnost, pouzdanost, portabilnost, podrška i performanse. Detaljan opis i poređenja dostupnih BDD paketa se mogu naći u [**?**].

Osnovni principi na kojima se bazira arhitektura jednog BDD paketa za rad sa dijagramima odlučivanja je uvedena u [**?**]:

1. Prekidačke funkcije se predstavljaju kao usmereni aciklični graf sa više početnih čvorova.
2. Čvorovi se čuvaju u jedinstvenoj tablici (unique table).
3. Rezultati operacija se pamte u tablici računanja (operation table) i na taj način se ubrzavaju buduće operacije.
4. Uređenje promenljivih u dijagramu mora da se menja kako bi se redukovao ukupan broj čvorova u dijagramu.
5. Čvorovi koriste strukturu C jezika i sadrže sledeće informacije: identifikator promenljive "index"(na kom nivou se u dijagramu nalazi čvor), “then” i “else” pokazivače na svoje čvorove potomke i “next” pokazivač na sledeći čvor na istom nivou.
6. Reciklaža čvorova se implementira brojanjem referenci na svaki čvor.

U ovom poglavlju su u nastavku detaljnije opisani osnovni koncepti implementacije većine BDD paketa. Razumevanje ovih koncepata je važno kod realizacije modifikacije i proširenja BDD paketa za izračunavanje autokorelacionih koeficijenata preko dijagrama odlučivanja.

5.1.1 Implementacija čvorova

Na osnovu strukture BDD-a, koja je prikazana u primeru na slici 2.5, potpuno je očigledno šta treba uzeti u obzir kod kreiranje strukture podataka kojim se predstavlja čvor u BDD paketu. Osnovni elementi te strukture podataka su:

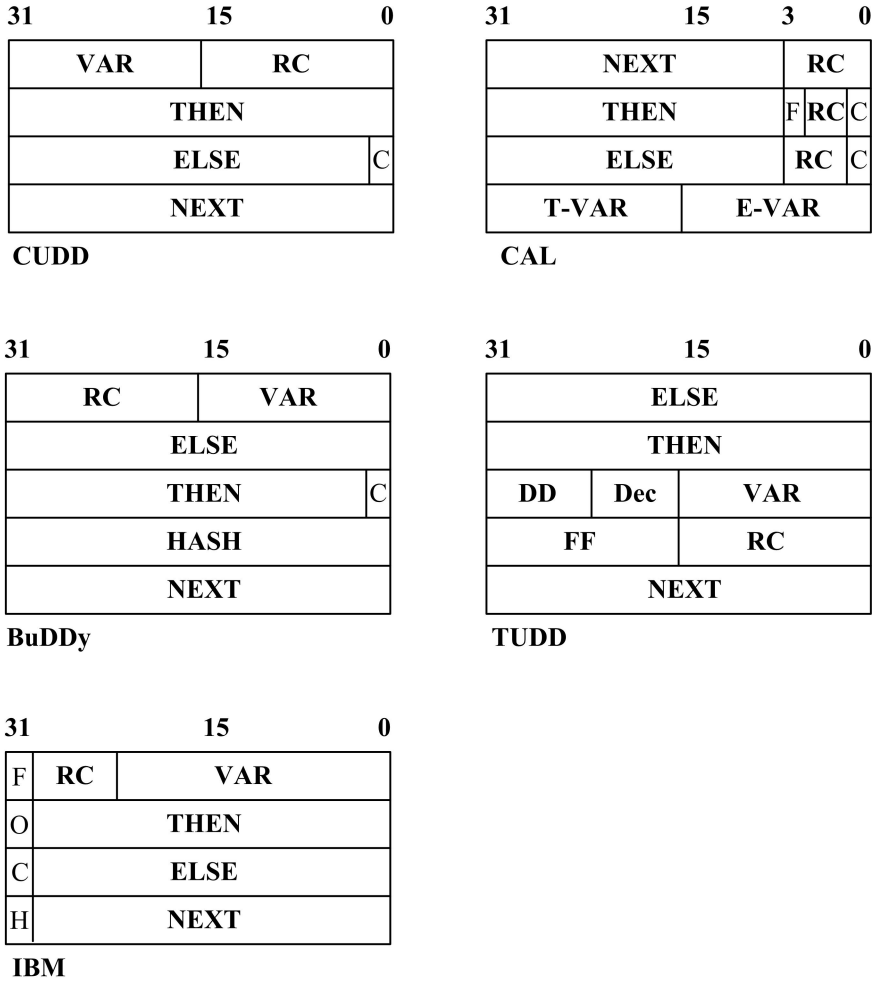
- Element `Index` predstavlja indeks i promenljive prekidačke funkcije x_i ili predstavlja nivo na kom se nalazi čvor u dijagramu.
- Element `Then` predstavlja pokazivač na čvor sledbenik koji se nalazi na grani sa labelom 1 (sadrži memorijsku adresu strukture podataka 0-čvora sledbenika).
- Element `Else` predstavlja pokazivač na čvor sledbenik koji se nalazi na grani sa labelom 0 (sadrži memorijsku adresu strukture podataka 1-čvora sledbenika).

Veličina memorijskog prostora za predstavljanje elementa `Index` najčešće se postavlja na 16 bitova što omogućava rad sa prekidačkim funkcijama do 65536 promenljivih. Veličina elemenata `Then` i `Else` je zavisna od veličine prostora koji se koristi u arhitekturi računara za adresiranje memorije (najčešće 32 ili 64 bita). U slučaju 32-bitne arhitekture adresni prostor je veličine 2^{32}. Razlikovanje terminalnih od neterminalnih čvorova kod implementacija BDD paketa se rešava postavljanjem specijalnih vrednosti u pokazivače `Then` i `Else` ili specijalnih vrednosti u element u elenekim `Index`.

Međutim, u praksi se pokazalo da je iz razloga efikasnosti implementacije potrebno proširenje osnovne strukture čvora BDD paketa. Naravno postoji mnogo različitih izbora za proširenje osnovne strukture čvora. Kod različitih BDD paketa se sreću različiti tipovi proširenja. U nastavku je data lista nekih najčešćih elemenata za proširenje osnovne strukture čvora:

- Element `C` za markiranje posećenosti čvora koji se koristi kod raznih algoritama za obilazak dijagrama odlučivanja.
- Element `RC` za brojanje referenci na čvor koji se koristi kod raznih algoritama za sakupljanje otpadnih čvorova, kao i za recikliranje čvorova.
- Elemant `Next` predstavlja pokazivač za različita ulančavanja čvorova (na primer, ulančavanje svih čvorova na istom nivou u dijagramu odlučivanja ili ulančavanje čvorova u heš tabeli).

Svi elementi koji ulaze u strukturu čvora zauzimaju određenu veličinu memorijskog prostora i moraju da budu efikasno smešteni u jednu strukturu. Ovaj postupak zahteva



Slika 5.1: Struktura BDD čvora kod 5 najpoznatijih BDD paketa.

pažljivo balansiranje memorijske veličine elemenata strukture čvora. Na slici 5.1 prikazana je kako izgleda struktura BDD čvora kod 5 najpoznatijih BDD paketa [**?**].

Na osnovu strukture čvorova pet BDD paketa može se uočiti da veličina meorijskog prostora potrebnog za smeštanje čvora iznosi 16 ili 20 bajtova (u jednom slučaju). CUDD paket predstavlja prototip strukture BDD paketa koji se bazira na pokazivačima. Struktura čvora ovog paketa zauzima 16 bajtova: indeks promenljive (VAR), brojač referenci (RC), pokazivači (Then, Else i Next). Bit za komplementiranje (C) je bit najmanje težine (Else) pokazivača. Očigledno je da smanjenjem memorijskog prostora za smeštanje brojača referenci se štedi na ukupnom memorijskom prostoru za smeštanje čvora. U slučaju CUDD paketa za brojač referenci se koristi 16 bitova, odnosno maksimalan broj ulaznih potega u jedan čvor može da bude $2^{16} = 65536$. CAL paket u strukturi BDD čvora smešta indekse promenljivih čvorova potomaka. Na ovaj način se informacije o čvorovima potomcima dobijaju bez prethodnog pristupa njima. Ušteda memorijskog prostora u strukturi BDD čvora se postiže distribuiranjem brojača referenci na bitove

najmanje težine u okviru tri različita pokazivača (Then, Else i Next). Ova ušteda memorijskog prostora ima za posledicu smanjenje performansi jer je na nivou bitova potrebno dodatno vreme za sastavljanje i rastavljanje brojača referenci. Struktura čvora u BuDDy paketu izgleda vrlo slično CUDD paketu stom razlikom što ova struktura ne koristi pokazivače. Dodatni element (Hash) se koristi za smeštanje početne adrese čvora u lančanoj listi čvorova koji dele isti heš ključ u jedinstvenoj tablici čvorova. IBM paket koristi veliki memorijski prostor za smeštanje indeksa promenljive. 2-bitni brojač referenci se ne koristi za sakupljanje otpadnih čvorova već sprečava smeštanje nepotrebnih čvorova u tablicu operacija. TUD paket koristi 3 dodatna elementa (jedan 16-bitni i dva 8-bitna) za markiranje čvorova sa ciljem da se ubrza vreme pristupa elementima čvora (nema dodatnog vremena za izdvajanje odgovarajućih bitova).

Proširenjem osnovne strukture čvora BDD paketa performanse određenog algoritma koji koristi dijagram odlučivanja kao osnovnu strukturu mogu znatno da se poboljšaju. Proširenja strukture čvora moraju da se vrše veoma oprezno s obzirom da utiču na rapidno smanjenje memorijskog prostora za smeštanje čvorova dijagrama. Ograničeni memorijski resursi računara u praktičnim primenama za prekidačke funkcije, čiji dijagrami odlučivanja imaju milione čvorova, mogu u mnogome da utiču na upotrebljivost određenog BDD paketa. Zbog toga je jedan od ključnih aspekata kod implementacije BDD paketa pronalaženje balansa između proširenja strukture čvora (efikasnosti izračunavanja) i praktične primenljivosti BDD paketa. Pronalaženje ovog balansa za efikasnu implementaciju izračunavanja autokorelacijonih koeficijenata za prekidačke funkcije sa velikim brojem promenljivih je bio glavni motiv za kreiranje specijalizovanog BDD paketa proširenjem standardnih BDD tehinka.

Strukture podataka koje se koristi za implementaciju čvorova u BDD paketima se razlikuju u zavisnosti od implementacije BDD paketa, programskog okruženja koje se koristi i namene BDD paketa [**?**].

Na slici 5.2 prikazan je primer implementacije BDD čvora u CUDD paketu korišćenjem C programskog jezika [**?**].

Slika 5.2 prikazuje spisak promenljivih i njihovu organizaciju u strukturu koja opisuje čvor dijagrama u CUDD paketu. Struktura `struct DdChildren` se sastoji iz dva elementa `*T` i `*E` koji predstavljaju pokazivače na adrese čvorova sledbenika u memoriji. Ova struktura se koristi samo kod definisanja neterminalnih čvorova u dijagrama. Opšta struktura čvora (koja se korisiti kod neterminalnih i terminalnih čvorova) `struct DdNode` se sastoji od četiri elementa: `index`, `ref`, `*next` i `type`. Promenljiva `index` definiše indeks i promenljive prekidačke funkcije kojoj pripada čvor, `ref` definiše brojač referenci na čvor (broj čvorova koji imaju poteg ka ovom čvoru), `*next` definiše pokazivač na sledeći čvor u jedinstvenoj tablici i `type` predstavlja strukturu tipa unije koja se sastoji iz dva elementa

```
struct DdChildren
{
    struct DdNode *T;
    struct DdNode *E;
}
struct DdNode
{
    DdHalfWord index;
    DdHalfWord ref;
    DdNode next;
    union{
        CUDD VALUE TYPE value;
        DdChildren kids;
    } type;
}

}
```

Slika 5.2: Primer implementacije BDD čvora u CUDD paketu korišćenjem C programskog jezika

`value` koji definiše vrednost terminalnog čvora i `kids` tipa strukture `struct DdChildren`. Samo jedan element strukture tipa unije je aktivan u istom trenutku i elementi unije dele istu lokaciju u memoriji računara.

Na slici 5.3 prikazan je primer implementacije BDD čvora u PUMA paketu korišćenjem C++ programskog jezika [**?**]. Paket PUMA osim podrške za rad sa binarnim dijagramima odlučivanja je prilagođen i za rad sa višeznačnim dijagramima odlučivanja (eng. MDD - Multiple-place Decision Diagram).

Slika 5.3 prikazuje spisak promenljivih i njihovu organizaciju u strukturu koja opisuje čvor dijagrama u CUDD paketu. Struktura `struct DdChildren` se sastoji iz dva elementa `*T` i `*E` koji predstavljaju pokazivače na adrese čvorova sledbenika u memoriji. Ova struktura se koristi samo kod definisanja neterminalnih čvorova u dijagrama. Opšta struktura čvora (koja se korisiti kod neterminalnih i terminalnih čvorova) `struct DdNode` se sastoji od četiri elementa: `index`, `ref`, `*next` i `type`. Promenljiva `index` definiše indeks i promenljive prekidačke funkcije kojoj pripada čvor, `ref` definiše brojač referenci na čvor (broj čvorova koji imaju poteg ka ovom čvoru), `*next` definiše pokazivač na sledeći čvor u jedinstvenoj tablici i `type` predstavlja strukturu tipa unije koja se sastoji iz dva elementa `value` koji definiše vrednost terminalnog čvora i `kids` tipa strukture `struct DdChildren`. Samo jedan element strukture tipa unije je aktivan u istom trenutku i elementi unije dele istu lokaciju u memoriji računara.

```
class MddNode
{
    public :
    unsigned v;
    pMddNode G,H,I,J,K;
    unsigned Key;
}

typedef struct node *DDedge;
typedef struct node *DDlink;
typedef struct node
{
char value, flag;
DDlink next;
DDedge edge[0];
} node;

}
```

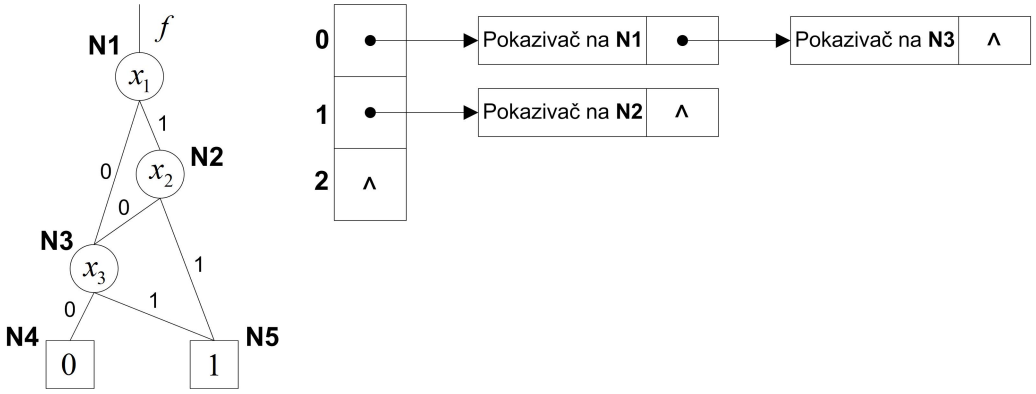
SLIKA 5.3: Primer implementacije BDD čvora u PUMA paketu korišćenjem C++ programskog jezika

5.1.2 Implementacija jedinstvene tablice

Redukciona pravila kod dijagrama odlučivanja služe za eliminaciju izomorfnih poddijagrama u cilju smanjenja memorijskog prostora potrebnog za pamćenje dijagrama. Umesto ponavljanja poddijagrama koji odgovaraju istim podfunkcijama kod implementacija BDD paketa koriste se pokazivači za eliminisanje izomorfnih poddijagrama u dijagramima odlučivanja. Isti koncept se koristi i kod memorisanja razdeljenih dijagrama odlučivanja gde se eliminacija izomorfnih poddijagrama sagledava nad svim izlazima funkcije istovremeno. U procesu konstrukcije dijagrama odlučivanja kod BDD paketa praksa je da se ne ponavlja konstruisanje izomorfnih poddijagrama u dijagramu odlučivanja. Ovo je obezbeđeno korišćenjem posebne strukture podataka koja se naziva jedinstvena tablica čvorova u dijagramu odlučivanja [**?**].

Prilikom dodavanja novog čvora u dijagram odlučivanja, njegov indeks i promenljive prekidačke funkcije x_i, *Then* pokazivač na postojeći čvor sledbenik koji se nalazi na grani sa labelom 1 i *Else* pokazivač na postojeći čvor sledbenik koji se nalazi na grani sa labelom 0 moraju da budu specificirani. Da bi se dijagram odlučivanja održao u redukovanoj formi mora se proveriti da li u dijagramu čvor sa takvom specifikacijom već postoji. Ako ne postoji, novi čvor se konstruiše, u suprotnom koristi se već postojeći čvor.

Svaki čvor u dijagramu odlučivanja predstavlja prekidačku funkciju i može se označiti slovom N. Takođe, čvor N se može definisati uređenom trojkom $N = (x_i, T, E)$, gde x_i predstavlja promenljivu koja odgovara čvoru N, T je čvor koji je povezan na granu sa labelom 1 čvora N i E je čvor koji je povezan na granu sa labelom 0 čvora N. Odluka o tome da li je uređenom trojkom $N = (x_i, T, E)$ već predstavljen neki čvor u dijagramu se donosi na osnovu jedinstvene tablice. U cilju efikasnog određivanja poklapanja uređene trojke sa već postojećim čvorom koji je takođe predstavljen preko uređene trojke, jedinstvena tablica se implementira kao heš tabela [**?**], [**?**], [**?**]. Svaki čvor u dijagramu odlučivanja mora da bude prisutan u jedinstvenoj tablici. Uređena trojka (x_i, T, E) čvora N se preslikava u heš ključ (vrednost) $h(x_i, T, E)$. Na poziciji heš ključa se u nizu pod nazivom *uniquetable* smešta pokazivač na čvor N. Više različitih uređenih trojki može da ima istu vrednost heš ključa. Zbog toga se niz *uniquetable* implementira kao niz lančanih listi, koje se nazivaju kolizione liste [**?**]. Ako se veličina niza dovoljno velika i funkcija za kreiranje heš ključa $h(x)$ dobro izabrana, onda svaka lančana lista u nizu lančanih listi nema mnogo elemenata. U tom slučaju je pretraživanje čvorova u okviru lančane liste (uređenih trojki) efikasno. Primer smeštanja čvorova BDD-a za prekidačku funkciju definisanu sa $f_1(x_1, x_2, x_3) = x_1x_2 + x_3$ čiji je vektor istinitosti $F = [01010111]^T$ u jedinstvenu tablicu veličine 3, čiji su indeksi elemenata u opsegu od 0 do 2, je prikazan na slici 5.4.



Slika 5.4: Primer smeštanja čvorova BDD-a za prekidačku funkciju definisanu sa $f_1(x_1, x_2, x_3) = x_1x_2 + x_3$ u jedinstvenu tablicu veličine 3.

Na slici 5.4 je prikazan BDD funkcije $f_1(x_1, x_2, x_3) = x_1x_2 + x_3$ gde su interno označeni čvorovi $N1, N2, ..., N5$. Oznake Ni služe za jednoznačno obeležavanje čvorova BDD-a i takođe mogu biti i memorijske adrese čvorova. Jedinstvena tablica je predstavljena nizom dužine 3 sa indeksima elemenata 0, 1 i 2. Heš funkcija za smeštanje čvora predstavljenim uređenom trojkom $h(N_i, N_j, N_k)$ može da bude definisana kao $h(N_i, N_j, N_k) = (i + j + k) mod 2$. Na primer, heš funkcija čvora $N2$ na osnovu prethodno definisane formule ima vrednost $h(N_2, N_3, N_5) = (2 + 3 + 5) mod 2 = 1$.

Na ovaj način se BDD redukuje pre konačnog kreiranja. Podfunkcija koja se predstavlja uređenom trojkom (x_i, T, E) već postoji u BDD-u isključivo ako postoji čvor u lančanoj kolizionoj listi u okviru jedinstvene tablice koji se može predstaviti pomoću iste uređene trojke. Jedinstvena tablica funkcioniše isto za BDD i za SBDD.

Radi veće fleksibilnosti kod jedinstvene tablice najčešće se koristi koncept otvorenog heširanja sa kolizionim listama. Kolizione liste mogu da se održavaju sortirane kako bi se smanjio prosečan broj memoriskih pristupa koji se zahteva prilikom pretraživanja heš tabele. U poslednje vreme predlaže se i koncept zatvorenog heširanja u cilju smanjenja memorijskog prostora za pamćenje čvorova, odnosno eliminisanje *Next* pokazivača za ulančavanje čvorova u kolizionim listama [**?**].

Jedinstvena tablica se često deli na jedinstvene podtabele (jedna tabela za svaku promenljivu funkcije, odn. BDD-a) [**?**]. Ovakva organizacija dozvoljava brži pristup čvorovima koji pripadaju određenom nivou BDD-a. Podela na podtabele je izrazito korisna kod promena uređenja promenljivih u BDD-u. Za terminalne čvorova se mogu koristiti dodatne tabele što je posebno efikasno kod multiterminalnih tipova dijagrama odlučivanja. Kod korišćenja jedinstvenih podtabela element `Index` u strukturi čvora se koristi za selekciju podtabele, dok se heš ključ kreira primenom heš funkcije na pokazivače čvorova potomaka. Veličina svake podtabele je prost broj ili broj stepen dvojke. Heš funkcije najčešće ima oblik [**?**]: $((T \cdot p_1 + E) \cdot p_2)/2^{b-k}$, gde su T i E pokazivači na čvorove potomke, p_1 i p_2 su odgovarajuće izabrani prosti brojevi, b je dužina reči koja se koristi kod izračunavanja heš funkcije i 2^k je veličina jedinstvene tablice. Podtabele su na početku kreiranja dijagrama prazne i one se sve više pune kako se sve više čvorova dodaje u BDD. Zbog toga faktor opterećenja heš tabele [**?**] ne seme da pređe zadatu vrednost. U slučaju da faktor opterećenja bude veći od zadate vrednosti, iz odgovarajuće podtabele moraju da se uklone svi neaktivni čvorovi (primena sakupljača otpada). Operacija sakupljača otpada kod implementacije BDD paketa biće detaljnije opisana kasnije u ovom poglavlju. U slučaju da je broj neaktivnih čvorova u okviru podtabele mali primena sakupljača otpada na jedinstvenu podtabelu neće biti efektivna za smanjenje faktora opterećenja. Česta primena sakupljača otpada može smanjiti performanse paketa, zato se koriste i strategije akumuliranja neaktivnih čvorova u podtabelama.

Na slici 5.5 prikazan je primer implementacije strukture jedinstvene tablica u CUDD paketu korišćenjem C programskog jezika sa odgovarajućim komentarima [**?**]. Primer realizacije implementacije strukure jedinstvene tablica u okviru strukture `struct DdManager` u CUDD paketu korišćenjem C programskog jezika sa odgovarajućim komentarima je prikazan na slici 5.6.

Vrednosti veličine i faktora opterećenja jedinstvene tablice u mnogome zavise od raspoloživih hardverskih resursa za izvršavanje BDD paketa. Generalno, porast jedinstvene

```
typedef struct DdSubtable {      /* podtabele za jedan index */
     DdNode **nodelist;          /* heš tabela */
     int shift;                  /* pomeraj za heš funkciju */
     unsigned int slots;         /* veličina heš tabele */
     unsigned int keys;          /* broj čvorova smešten u ovu tabelu */
     unsigned int maxKeys;       /* maksimalna gustina podtabele */
     unsigned int dead;          /* broj neaktivnih čvorova u ovoj tabeli */
     unsigned int next;          /* indeks sledeće promenljive u grupi */
     int bindVar;                /* indikator koji povezuje promenljivu */
                                 /* sa njenim nivoom */
     /* elementi kod primene preuređenja promenljivih */
     Cudd_VariableType varType;  /* tip promenljive */
     int pairIndex;              /* odgovarajući indeks promenljive */
     int varHandled;             /* indikator: 1 znači da je promenljiva */
                                 /* već obrađena */
     Cudd_LazyGroupType varToBeGrouped; /* označava koja grupa se primenjuje */
} DdSubtable;
```

Slika 5.5: Primer implementacije strukture jedinstvene tabele u CUDD paketu korišćenjem C programskog jezika sa odgovarajućim komentarima.

tablice mora de se uspori u slučaju da je za izvršavanje paketa ostalo malo slobodne fizičke memorije.

5.1.3 Implementacija tablice operacija

Prilikom konstruisanja dijagrama odlučivanja može da se desi da se nekoliko puta ista operacija izvršava nad istim poddijagramima u dijagramu odlučivanja. Ovo nepotrebno ponavljanje operacija treba izbeći i kod BDD paketa se ovaj problem rešava kreiranjem memorijskih funkcija za prethodno izračunate rezultate. U cilju efikasnosti rada sa ovim memorijskim funkcijama, one se smeštaju u specijalnu strukturu koja se naziva tablica operacija. Svaka operacija u dijagramu je jednoznačno određena sa pokazivačima na odgovarajuće čvorove dijagrama operanada i jedan pokazivač na čvor rezultat koji treba sačuvati. Na ovaj način pre izvršavanja izračunavanja u okviru dijagrama, pretraživanje tablice operacija može da skarti vreme izračunavanja preuzimanjem već prethodno izračunatog rezultata iz ove tablice. Ukoliko u tablici operacija ne postoji odgovarajući rezultat izvršava se algoritam izračunavanja i rezultat se smešta u ovu tablicu.

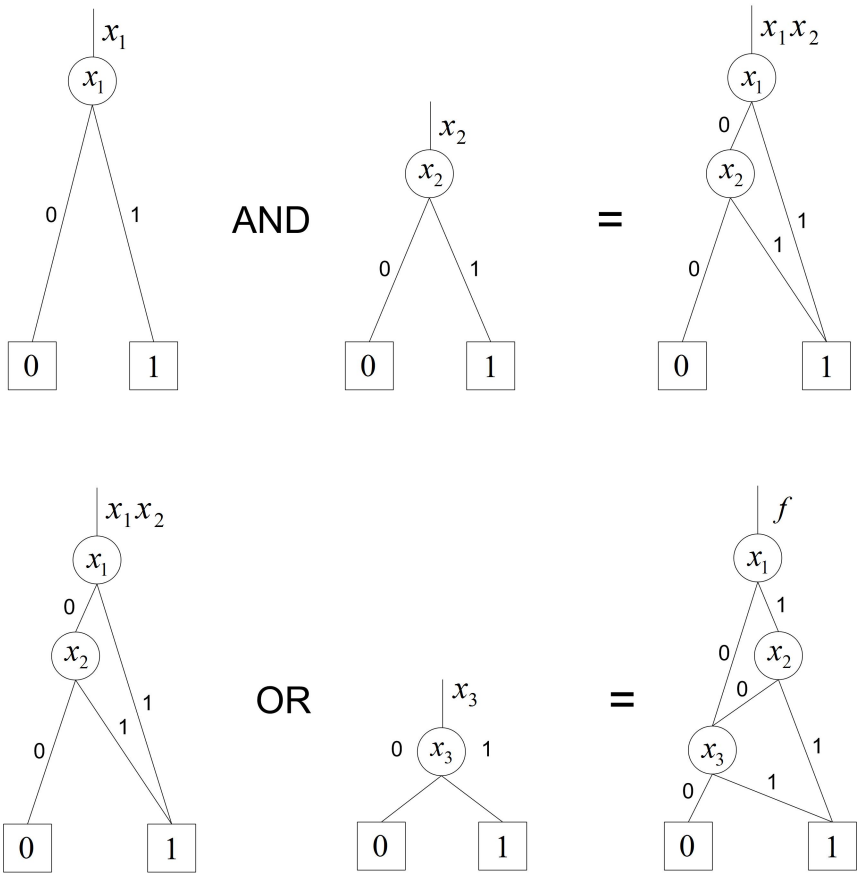
Dijagrami odlučivanja se konstruišu rekurzivnom primenom skupa operacija u zavisnosti od: domena i opsega funkcija koje se predstavljaju preko dijagrama, specifikacija funkcija i operacija koje se koriste kod konstruisanja dijagrama odlučivanja. Kod procesa konstruisanja dijagrama odlučivanja primarno se konstruišu bazični dijagrami promenljivih a potom se vrši njihovo rekurzivno kombinovanje u poddijagrame. Primer rekurzivnog

```
struct DdManager {      /* struktura koja upravlja manipulacijama sa DD-om */
     . . .
     /* Unique Table */
     int size;                  /* broj jedinstvenih podtabela */
     int sizeZ;                 /* za ZDD */
     int maxSize;               /* maksimalni broj podtabela pre */
                                /* redimenzionisanja */
     int maxSizeZ;              /* za ZDD */
     DdSubtable *subtables;     /* niz jedinstvenih podtabela */
     DdSubtable *subtableZ;     /* za ZDD */
     DdSubtable constants;      /* jedinstvena podtabela za konstante */
     unsigned int slots;        /* ukupan broj elemenata u hešu */
     unsigned int keys;         /* ukupan broj BDD i ADD čvorova */
     unsigned int keysZ;        /* ukupan broj ZDD čvorova */
     unsigned int dead;         /* ukupan broj neaktivnih BDD i ADD čvorova */
     unsigned int deadZ;        /* ukupan broj neaktivnih ZDD čvorova */
     unsigned int maxLive;      /* maksimalni broj aktivnih čvorova */
     unsigned int minDead;      /* minimalni broj neaktivnih čvorova */
                                /* za primenu sakupljača otpada */
     double gcFrac;             /* aktiviraj sakupljač otpada u slučaju */
                                /* da je ovaj identifikator neaktivan */
     int gcEnabled;             /* sakupljač otpada je aktiviran */
     unsigned int looseUpTo;    /* koristi spor rast tabele do ove granice */
                                /* (meri se broj upisa u elemente */
                                /* podtabela, ne broj heš ključeva) */
     unsigned int initSlots;    /* inicijalna veličina jedne podtabele */
     DdNode **stack;            /* stek za iterativne procedure */
     double allocated;          /* broj čvorova koji je alociran */
                                /* (ne u toku preuređenja promenljivih) */
    . . .
}
```

SLIKA 5.6: Primer realizacije implementacije strukure jedinstvene tablica u CUDD paketu korišćenjem C programskog jezika sa odgovarajućim komentarima.

konstruisanja BDD-a za prekidačku funkciju definisanu sa $f_1(x_1, x_2, x_3) = x_1x_2 + x_3$ čiji je vektor istinitosti $F = [01010111]^T$ prikazan je na slici 5.7.

Na slici 5.7 je prikazan korak po korak postupak konstruisanja BDD-a funkcije $f_1(x_1, x_2, x_3) = x_1x_2 + x_3$. U prvom koraku su prvo kreirani binarni dijagrami odlučiavanja za funkcije (promenljive) x_1 i x_2. Potom su ova dva dijagrama kombinovana u dijagram za funkciju x_1x_2 primenom logičke AND operacije. U drugom koraku je kreiran binarni dijagram odlučiavanja za funkciju (promenljivu) x_3. Potom je ovaj dijagram kombinovan sa dijagramom kreiranim u prethodnom koraku za funkciju x_1x_2 primenom logičke OR operacije. Na kraju je prikazan rezultujući dijagram za funkciju $x_1x_2 + x_3$. U ovom primeru operacije koje se koriste za konstruisanje dijagrama odlučivanja predstavljaju logičke AND i OR operacije koje se primenjuju nad dekompozicionim pravilima (Shanon-ovim



Slika 5.7: Primer rekurzivnog konstruisanja BDD-a za prekidačku funkciju definisanu sa $f_1(x_1, x_2, x_3) = x_1x_2 + x_3$ čiji je vektor istinitosti $F = [01010111]^T$.

pravilima) koji se koriste kod definicije BDD-a.

Kod izračunavanja BDD reprezentacije prekidačke funkcije $f * g$, gde se binarna operacija $*$ izvršava nad dva operanda $BDD(f)$ i $BDD(g)$ i gde su f i g dve prekidačke funkcije, a $BDD(f)$ i $BDD(g)$ predstavljaju BDD reprezentacije ovih funkcija, korisiti se Shanon-ov razvoj u odnosu na promenljivu x_i [**?**]:

$$f * g = \overline{x}_i(f(x_i = 0) * g(x_i = 0)) + x_i(f(x_i = 1) * g(x_i = 1)) \tag{5.1}$$

U cilju da se sve binarne operacije tretiraju na isti način kreiran je takozvani "ITE operator"(eng. ITE - If Then Else) [**?**]. ITE operator se definiše kao nova prekidačka funkcija sa tri promenljive x, y i z koja izračunava vrednost sledećeg izraza: If x, Then y, Else z. Analitički ITE operator može da se definiše kao:

$$ITE(x, y, z) = x \cdot y + \overline{x} \cdot z \tag{5.2}$$

Tabela 5.1: Realizacija svih 16 binarnih operacija preko ITE operatora.

Binarna operacija	ITE operator
0	$ITE(0,0,0)$
$f \cdot g$	$ITE(f,g,0)$
$f \not\Rightarrow g$	$ITE(f,\overline{g},0)$
f	$ITE(f,1,0)$
$f \not\Leftarrow g$	$ITE(f,0,g)$
g	$ITE(g,1,0)$
$f \oplus g$	$ITE(f,\overline{g},g)$
$f + g$	$ITE(f,1,g)$
$f \mp g$	$ITE(f,0,\overline{g})$
$f \equiv g$	$ITE(f,g,\overline{g})$
$\overline{g}$	$ITE(g,0,1)$
$f \Leftarrow g$	$ITE(f,1,\overline{g})$
$\overline{f}$	$ITE(f,0,1)$
$f \Rightarrow g$	$ITE(f,g,1)$
$f \barwedge g$	$ITE(f,\overline{g},0)$
1	$ITE(1,1,1)$

ITE operator je primenljiv u kontekstu operacija koje se izvršavaju nad BDD-om i one oslikavaju Shanon-ovo dekompoziciono pravilo na čvoru BDD-a. Postoje 16 mogućih različitih binarnih operacija i sve one se mogu izraziti preko ITE operatora. Realizacija svih 16 binarnih operacija preko ITE operatora je prikazana u tabeli 5.1 [**?**].

Kod izračunavanja operatora $ITE(f,g,h)$ gde su f, g i h prekidačke funkcije i x_i sledeća rekurzivna dekompozicija može biti definisana [**?**]:

$$\begin{aligned} ITE(f,g,h) &= f \cdot g + \overline{f} \cdot h \\ &= x_i \cdot (f \cdot g + \overline{f} \cdot h)_{x_i} + \overline{x}_i \cdot (f \cdot g + \overline{f} \cdot h)_{\overline{x}_i} \\ &= x_i \cdot (f_{x_i} \cdot g_{x_i} + \overline{f}_{x_i} \cdot h_{x_i}) + \overline{x}_i \cdot (f_{\overline{x}_i} \cdot g_{\overline{x}_i} + \overline{f}_{\overline{x}_i} \cdot h_{\overline{x}_i}) \\ &= ITE(x_i, ITE(f_{x_i}, g_{x_i}, h_{x_i}), ITE(f_{\overline{x}_i}, g_{\overline{x}_i}, h_{\overline{x}_i})) \\ &= (x_i, ITE(f_{x_i}, g_{x_i}, h_{x_i}), ITE(f_{\overline{x}_i}, g_{\overline{x}_i}, h_{\overline{x}_i})) \end{aligned} \tag{5.3}$$

Uređena trojka koja se javlja u poslednjem koraku rekurzivne dekompozicije ITE operatora ima značenje Shanon-ovog razvoja u odnosu na promenljivu x_i. Čvor sa indeksom x_i i njegovi rekurzivno definisani potomci se kreiraju u slučaju da takav čvor ne postoji u dijagramu. Rekurzija $ITE(f,g,h)$ se stopira ako je prvi argument ovog operatora konstanta, odnosno $ITE(1,f,g) = f$ ili $ITE(0,f,g) = g$. Osim toga, rekurzija može da

se stopira i u sledećim slučajevima: $ITE(f,1,0) = f$ ili $ITE(f,g,g) = g$. Opis ITE algoritma u pseudo kodu [**?**] prikazan je u nastavku.

Algoritam 5.1 ITE algoritam

```
1: ITE(F,G,H)
2: INPUT: F, G i H kao BDD reprezentacije funkcija f, g i h
3: OUTPUT: BDD reprezentacija funkcije ITE(f,g,h)
4: if terminalnislucaj then
5:    Return(rezultatterminalnogslucaja)
6: else
7:    if (F,G,H) ∈ OperationTable then
8:       ReturnOperationTable(F,G,H)
9:    end if
10: else
11:    Neka je promenljiva x_i prva u uređenju BDD reprezentacija F, G i H
12:    T = ITE(F_{x_i}, G_{x_i}, H_{x_i})
13:    E = ITE(F_{x̄_i}, G_{x̄_i}, H_{x̄_i})
14:    nađi ili dodaj u R = UniqueTable(v,T,E)
15:    dodaj u OperationTable((F,G,H),R,)
16:    Return(rezultatterminalnogslucaja)
17: end if
```

U opisu ITE algoritma koristi se tablica operacija za smeštanje već izračunatih rezultata. U algoritmu se koriste dve operacije u vezi sa tablicom: pronalaženje i dodavanje određene uređene trojke u tablicu. Funkcija za pronalaženje određenog rezultata operacije, ukoliko je rezultat pronađen u tablici, vraća pokazivač na čvor (rezultat) u dijagramu, u sprotnom se kreira novi čvor i vraća se pokazivač na taj novokreirani čvor. Funkcija za dodavanje u tablicu operacija smešta uređenu trojku koja predstavlja izračunatu operaciju na odgovarajuće mesto u tablici operacija.

Kod praktičnih primena ITE algoritma efikasnost izvršavanja ovog algoritma je u čvrstoj vezi sa veličinom rezultujućeg BDD-a. Vremenska kompleksnost primene $ITE(F,G,H)$ se može izraziti formulom $O(size(F) \cdot size(G) \cdot size(H))$. Zbog toga u slučaju dijagrama sa velikim brojem čvorova u smislu memorijskog prostora nije preporučljivo korišćenje tablice operacija. Ista konstatacija važi i za korišćenje jedinstvene tablice.

Tablica operacija se kod BDD paketa implementira, slično kao i jedinstvena tablica, korišćenjem heš tabele sa kolizionim lisrtama. Kod korošćenja heš tabela za tablicu operacija može da se dodgodi da već izračunati rezultati ne budu više potrebni kod budućih izračunavanja. U tom slučaju kod kreiranja tablice operacija postoje razni kompromisi u njenoj implementaciji. Pregled raznih tehnika za implemntaciju tablice operacija može se naći u radu [**?**].

Na slici 5.8 prikazan je primer implementacije strukture tablice operacija u CUDD paketu korišćenjem C programskog jezika [**?**]. Primer realizacije implementacije tablice

```
/* Generic hash item. */
typedef struct DdHashItem {
    struct DdHashItem *next;
    ptrint count;
    DdNode *value;
    DdNode *key[1];
} DdHashItem;

/* Local hash table */
typedef struct DdHashTable {
    unsigned int keysize;
    unsigned int itemsize;
    DdHashItem **bucket;
    DdHashItem *nextFree;
    DdHashItem **memoryList;
    unsigned int numBuckets;
    int shift;
    unsigned int size;
    unsigned int maxsize;
    DdManager *manager;
} DdHashTable;
```

Slika 5.8: Primer implementacije strukture tablice operacija u CUDD paketu korišćenjem C programskog jezika.

```
struct DdManager {      /* struktura koja upravlja manipulacijama sa DD-om */
     . . .
     /* Computed Table */
     DdCache *acache;            /* adresa alocirane memorije za heš tabelu */
     DdCache *cache;             /* tablica operacija kao heš tabela */
     unsigned int cacheSlots;    /* veličina heš tabele */
     int cacheShift;             /* pomeraj za generisanje heš funkcije */
     double cacheMisses;         /* broj promašaja u heš tabeli */
     double cacheHits;           /* broj pogodaka u heš tabeli */
     double minHit;              /* procenat pogodaka u heš tabeli za */
                                 /* aktiviranje preuređenja */
     int cacheSlack;             /* broj praznih mesta u heš tabeli */
                                 /* za preuređivanje */
     unsigned int maxCacheHard;  /* maksimalna granica za veličinu heš tabele */
     . . .
}
```

Slika 5.9: Primer realizacije implementacije strukure tablice operacija u CUDD paketu korišćenjem C programskog jezika sa odgovarajućim komentarima.

operacija u okviru strukture `struct DdManager` u CUDD paketu korišćenjem C programskog jezika sa odgovarajućim komentarima je prikazan na slici 5.9.

Vrednosti veličine i faktora opterećenja tablice operacija u mnogome zavise od raspoloživih hardverskih resursa za izvršavanje BDD paketa. Ako se tablica operacija realizuje kao heš tabela sa kolizionim listama, iz razloga memorijske efikasnosti potrebno je sa vremena na vreme ukloniti elemente koji su dugo prisutni u tablici operacija.

5.1.4 Sakupljač otpada

Primenom raznih manipulacija sa BDD-ovima može da rezultuj kreiranjem mnogo BDD-ova kao međurezultata. Zbog toga, implementacija BDD paketa treba da obezbedi mehanizam za oslobađanje od BDD-ova međurezultata koji više nisu potrebni. Ova mehanizam je u literaturi poznat pod imenom "sakupljač otpada" Kod razdeljenih BDD-a je situacija još složenija pošto međurezultati moraju da budu dostupni svim izlazima funkcije.

Najjednostavniji način za praćenje aktuelnosti međurezultata je tehinka brojanja referenci na svaki čvor u BDD paketu (eng. - reference-counter) [**?**]. Najveći broj BDD paketa koristi ovu tehniku kao sakupljača otpada. Ako se za sve čvorove u BDD paketu vodi evidencija o broju referenci na čvorove, onda se jednostavnom proverom mogu eliminisati svi čvorovi koji čiji je broj referenci jednak nuli. To znači da je čvor neaktivan i da u budućim manipulacijama verovatno više neće biti potrebreban. Međutim, broj referenci na čvor ne određuje jednoznačno neaktivnost nekog čvora. Čvor može biti referenciran tablicom operacija, odnosno može biti operand ili rezultat neke izračunate operacije. Zbog toga se pre eliminacije nekog čvora moraju proveriti svi pokazivači kod elemenata u tablici operacija.

Efikasnost primene sakupljača otpada je veoma mala kada je u memoriji veliki broj čvorova. Zbog toga se koriste razne tehnike kod kojih se kreiraju sortirane lančane liste pokazivača neaktivnih čvorva, takozvane "free-list" [**?**]. Ove liste mogu biti organizovane kao stranice u kojima se smešta po hiljadu neaktivnih čvorova. Ovakav način organizacije smanjuje lokalnost pristupa u memoriji.

Osim "reference-counter" tehnike za sakupljača otpada se koristi i "mark-sweep" tehnika, gde se prvo vrši faza markiranje svih neaktivnih čvorova a potom faza njihove eliminacije. U praksi se pokazalo da se posle primene faze eliminacije čvorova javlja mnogo malih lokacija slobodne memorije što znatno može da uspori performanse BDD paketa. Iz tog razloga razvijena je i mark-sweep-update-sweep tehnika [**?**]. Ova tehnika se bazira na "mark-sweep" tehnici gde se nakon primene faze eliminacije čvorova primenljuje faza premeštanja čvorova u memoriji iz njihovih originalnih lokacija u male slobodne lokacije memorije. Nakon ove faze se primenjuje faza elimiacija neaktivnih čvorova u jedinstvenoj tablici i tablici operacija koja je nastala usled odgovarajućih premeštanja u memoriji.

Sakupljač otpada se kod BDD paketa aktivira periodično. Kod implementacije BDD paketa mora da postoji kompromis oko periode aktiviranja sakupljača otpada. Kod nekih operacija se dešava da se veliki broj neaktivnih čvorova u toku izvršavanja operacije ponovo reciklira. U slučaju da se sakupljač otpada aktivira prilično često može rezultovati smanjenju performansi izvršavanja BDD paketa.

Opis tipičnog algoritma za sakupljanje otpada koji koristi "reference-count" tehinku u pseudo kodu prikazan je u nastavku [**?**].

Algoritam 5.2 Sakupljač otpada (reference-count)

1: iz tablice operacija se brišu svi elementi koji imaju pokazivače na neaktivne čvorove
2: iz jedinstvene tablice se brišu svi elementi koji imaju pokazivač na neaktivan čvor
3: prilikom obilaska jedinstvene tablice svi pokazivači na neaktivne čvorove se ubacuju u "free-list"
4: brišu se svi čvorovi čiji se pokazivači nalaze u "free-list"

Algoritm sakupljača otpada se najčešće primenjuje kada ima dovoljno neaktivnih čvorova kako bi se isplatila njegova primena. Kod BDD paketa koji koriste "reference-count" tehinku u svakom trenutku izvršenja programa su poznati ukupan broj čvorova i ukupan broj neaktivnih čvorova. Kombinacija ova dva parametra se kod BDD paketa korisit za kreiranje uslova za aktivaciju sakupljača otpada. Treba napomenuti da korišćenje "reference-count" tehinke zahteva dodatni memorijski prostor u strukturi za pamćenje čvorova što je detaljno opisano ranije u ovom poglavlju.

5.1.5 Implementacija operacija preko dijagrama odlučivanja

Izračunavanja preko dijagrama odlučivanja se sastoje od izvršavanja nekih matematičkih operacija preko funkcija koje su predstavljene preko dijagrama. Izvršenjem matematičke operacije nad funkcijama koje su predstavljene preko dijagrama odlučivanja zahteva konstrukciju odgovarajućeg rezultujućeg dijagrama. Rezultujući dijagram se konstruiše primenom redukcionih pravila (eliminacijom redundantnih čvorova i deljenjem čvorova). Eliminacija redundantnih čvorva se implementira poređenjem vrednosti THEN i ELSE pokazivača na čvorove potomke, dok se deljenje čvorova implementira iskorišćavanjem već kreiranih podgrafova u dijagramu.

Opis tipičnog algoritma za izvršavanje matematičkih operacija sa dva operanda preko BDD-a u pseudo-kodu prikazan je u nastavku [**?**].

Čvorovi BDD-a predstavljeni su promenljivama kao što su *p.index*, *p.then* i *p.else* koje predstavljaju čvor *p*. Promenljiva *p.index* označava indeks promenljive prekidačke funkcije koja odgovara čvoru u dijagramu, *p.then* predstavlja pokazivač na čvor sledbenik

Algoritam 5.3 Pseudo-kod algoritma za izvršavanje operacije preko BDD-a

```
1: BDDop(p,q)
2: if terminal(p) or terminal(q) then
3:    r ← terminalFunc(p, q)
4:    return r
5: else
6:    if p = q then
7:       return p
8:    else
9:       if p.index > q.index then
10:         r1 ← BDDop(p.then,q)
11:         r0 ← BDDop(p.else,q)
12:         r ← GETNODE(p.index,r1,r0)
13:         return r;
14:      else
15:         if p.index < q.index then
16:            r1 ← BDDop(p,q.then)
17:            r0 ← BDDop(p,q.else)
18:            r ← GETNODE(p.index,r1,r0)
19:            return r;
20:         else
21:            r1 ← BDDop(p.then,q.then)
22:            r0 ← BDDop(p.else,q.else)
23:            r ← GETNODE(p.index,r1,r0)
24:            return r;
25:         end if
26:      end if
27:   end if
28: end if
```

koji se nalazi na grani sa labelom 1 i *p.else* predstavlja pokazivaL na čvor sledbenik koji se nalazi na grani sa labelom 0. Funkcija $terminal(p)$ je tačna ukoliko je p terminalni čvor, u suprotnom je netačna. Funkcija $terminalFunc(p, q)$ zavisi od matematičke operacije koja se primenjuje na terminalnim čvorovima dijagrama. Na primer kod AND logičke operacije ova funkcija vraća 0-terminalni čvor u slučaju da je jedan od čvorova p ili q 0-terminalni čvor, u suprotnom vraća 1-terminalni čvor. Osim toga, funkcija $GETNODE(index, then, else)$ kreira novi BDD čvor p gde se vrednosti $index$, $then$ i $else$ dodeljuju vrednostima $p.index$, $p.then$ i $p.else$ čvora p. Funkcija $GETNODE(index, then, else)$ takođe implementira eliminaciju redundantnih čvorova i deljenje čvorova (redukciona pravila). Opis tipičnog algoritma za izvršavanje funkcije $GETNODE$ u pseudo-kodu prikazan je u nastavku [**?**].

Eliminacija redundantnih čvorova se realizuje u drugoj liniji koda. Za deljenje čvorova p i q u BDD-u mora da važi pravilo da je $p.index = q.index$, $p.then = q.then$ i $p.else = q.else$. Funkcija $SearchNodeTable$ proverava da li čvor sa podacima $index$, $then$, $else$ već postoji u jedinstvenoj tablici. U slučaju da posotji funkcija $GETNODE$ vraća postojeći čvor iz jedinstvene tablice umesto kreiranja novog čvora. U suprotnom funkcija

Algoritam 5.4 Pseudo-kod algoritma funkcije *GETNODE*

```
1: GETNODE(index,then,else)
2: if then = else then
3:    return then;
4: end if
5: r ← SearchNodeTable(index,then,else);
6: if r exists then
7:    return r;
8: end if
9: r ← CreateNode(index,then,else);
10: AddNodeTable(r);
11: return r;
```

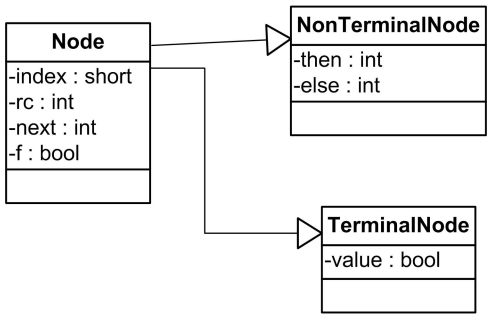
CreateNode kreira čvor sa podacima *index*, *then*, *else* i pomoću funkcije *AddNodeTable* ga smešta u jedinstvenu tablicu.

5.2 Implementacija SBDD algoritama sa permutovanim labelama i SBDD algoritma sa bočnim kretanjem

U ovom poglavlju, segmenti pseudo koda ilustruju jednu moguću implementaciju predloženog SBDD algoritma sa permutovanim labelama na granama za izračunavanje pojedinačnih totalnih autokorelacionih koeficijenata. Opis SBDD algoritma sa permutovanim labelama može se naći u poglavlju 4. Predložena implementacija može biti realizovana kao nadgradnja jezgra BDD paketa. Prethodno u ovom poglavlju su opisani osnovni principi i preporuke na kojima se bazira implementacija jezgra BDD paketa (jedinstvena tablica, tablica operacija, sakupljač otpada, itd.). Zbog toga su opisi ovih stavki u ovom poglavlju izostavljeni i celokupan fokus je usmeren ka implementaciji SBDD algoritma sa permutovanim labelama kao nadgradnje jezgra BDD paketa. U te svrhe u ovom poglavlju su predstavljene četiri ilustracije koje opisuju detalje implementacije predloženog algoritma: dve definicije struktura podataka u jezgru BDD paketa, opis procedure za permutovanje labela na granama i opis procedure za izračunavanje sumiranjem poddijagrama SBDD-a.

Jezgro BDD paketa kod ovog algoritma korisiti strukturu čvora koja je prikazana na slici 5.10.

Ova struktura čvora je veoma slična strukturi koja se korisiti u većini BDD paketa. Razlika u odnosu na većinu paketa je mehanizam za označavanje čvorova koji korisiti poseban element u strukturi čvora. Označavanje čvorova se korisiti kod implementacije algoritma za obilazak SBDD-a, a posebno je koristan kod implementacije algoritma sa bočnim kretanjem. Konkretno označavanje čvorova se korisiti prilikom kreiranja bočnog



Slika 5.10: Struktura čvora kod implementacije SBDD algoritma sa permutovanim labelama na granama.

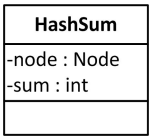
obilaska SBDD-a (funkcija $f(x \oplus u)$) za generisanje vrednosti autokorelacionog koeficijenta. Opis strukture čvora je realizovan korišćenjem UML model dijagrama.

Čvor kod implementacije SBDD algoritama za izračunavanje pojedinačnih autokorelacionih koeficijenata se sastoji od sledećih elemenata:

1. `index` predstavlja nivo na kom se nalazi čvor u dijagramu.
2. `rc` predstavlja broj dolaznih grana u čvor i korisiti se kod primene algoritma za sakupljanje otpadnih čvorova.
3. `f` predstavlja marker za označavanje posećenosti čvora koji se koristi kod algoritama za obilazak SBDD-a.
4. `next` predstavlja pokazivač za ulančavanje svih čvorova na istom nivou u dijagramu odlučivanja.
5. `then` predstavlja pokazivač na čvor sledbenik koji se nalazi na 1-grani (postoji samo za neterminalne čvorove).
6. `else` predstavlja pokazivač na čvor sledbenik koji se nalazi na 0-grani (postoji samo za neterminalne čvorove).
7. `value` - predstavlja vrednost terminalnog čvora, u slučaju binarnih dijagrama '0' ili '1' (postoji samo za terminalne čvorove).

Implementacija memorijske funkcije za pamćenje rezultata sumiranja svih vrednosti prekidačke funkcije predstavljene preko poddijagrama u SBDD-u zahteva dodatnu strukturu podataka. Jezgro BDD paketa kod ovog algoritma korisiti dodatnu strukturu hešsume koja je prikazana na slici 5.11.

Hešsuma kod implementacije SBDD algoritama sa permutovanim labelama na granama se sastoji od sledećih elemenata:



SLIKA 5.11: Struktura hešsume kod implementacije SBDD algoritma sa permutovanim labelama na granama.

1. `node` predstavlja pokazivač na čvor koji je koren poddijagrama u SBDD-u čiji rezultat sumiranja svih vrednosti funkcije predstavljene preko tog poddijagrma treba memorisati.

2. `sum` predstavlja vrednost sume za zadati poddijagrm u SBDD-u.

Sledeći algoritam ilustruje proceduru za permutovanje labela na granama SBDD-a kod SBDD algoritama sa permutovanim labelama na granama opisanu u pseudo-kodu.

Algoritam 5.5 Pseudo-kod procedure za permutovanje labela

```
1: PermuteLabel(p, shift[])
2: if p.f AND nonterminal(p) then
3:    PermuteLabel(p.then,shift[])
4:    PermuteLabel(p.else,shift[])
5:    p.f ← 1
6:    if shift[p.index] = 1 then
7:       tmp ← p.then
8:       p.then ← p.else
9:       p.else ← tmp
10:   end if
11: end if
```

Čvorovi BDD-a predstavljeni su promenljivom p, niz binarnih vrednosti autokorelacionog pomeraja je smešten u nizu $shift[]$. Funkcija $nonterminal(p)$ je tačna ukoliko je p neterminalni čvor, u suprotnom je netačna. U implementacionom smislu, rekurzivni obilazak SBDD za permutovanje svih labela na odgovarajućim garanama se izvršava na sličan način kao i ITE operacija koja se korisiti u jezgru BDD paketa.

Kod implementacije BDD paketa praksa je da se korisiti heš tabela kako bi se sprečilo kreiranje više instanci memorijske funkcije za pamćenje suma poddijagrama u SBDD-u. Jezgro BDD paketa kod ovog algoritma zahteva dodatnu strukturu `HashSum` koja je opisana na slici 5.11. Korisiti se jednostavan pristup za heširanje standardnim procedurama "Findža pretraživanje heš tabele i procedura "Addža smeštanje sume u heš tabelu.

Sledeći algoritam ilustruje proceduru za sumiranje terminala SBDD-a kod SBDD algoritama sa permutovanim labelama na granama opisanu u pseudo-kodu.

Algoritam 5.6 Pseudo-kod procedure za sumiranje terminala

```
1: SumTerminals(p)
2: hashsum.node ← p
3: if hashsum ∈ Find(SumTable) then
4:    return hashsum.sum
5: end if
6: if terminal(p) then
7:    r ← p.value
8: else
9:    i ← p.index − p.then.index
10:   r ← 2^i * SumTerminals(p.then)
11:   i ← p.index − p.else.index
12:   r ← r + 2^i * SumTerminals(p.else)
13: end if
14: hashsum.sum ← r
15: hashsum → Add(SumTable)
16: return r
```

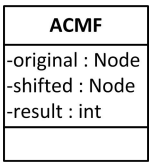
Čvorovi BDD-a predstavljeni su promenljivom p, funkcija $terminal(p)$ je tačna ukoliko je p terminalni čvor, u suprotnom je netačna. U implementacionom smislu, rekurzivni obilazak SBDD za sumiranje svih terminala se izvršava na sličan način kao i ITE operacija koja se korisiti u jezgru BDD paketa.

U ovom poglavlju, segmenti izvornog koda ilustruju jednu moguću implementaciju predloženog SBDD algoritma sa bočnim kretanjem za izračunavanje autokorelacionih koeficijenata. Opis SBDD algoritma sa bočnim kretanjem može se naći u poglavlju 4. Predložena implementacija može biti implementirana samo kao nadgradnja jezgra BDD paketa. Prethodno u ovom poglavlju su opisani osnovni principi i preporuke na kojima se bazira implementacija BDD paketa (jedinstvena tablica, tablica operacija, sakupljač otpada, itd.). Zbog toga su opisi ovih stavki u ovom poglavlju izostavljeni i celokupan fokus je usmeren ka implementaciji SBDD algoritma sa bočnim kretanjem kao nadgradnje jezgra BDD paketa. U te svrhe u ovom poglavlju su predstavljene tri ilustracije koje opisuju detalje implementacije predloženog algoritma: definicija strukture podataka u jezgru BDD paketa, opis procedure za izračunavanje bočnim obilaskom SBDD-a i finalna procedura za izračunavanje totalnih autokorelacionih koeficijenata.

Jezgro BDD paketa kod ovog algoritma korisiti strukturu čvora koja je prikazana na slici 5.10.

Implementacija memorijske funkcije za pamćenje rezultata autokorelacije između dva poddijagrama u SBDD-u zahteva dodatnu strukturu podataka. Jezgro BDD paketa kod ovog algoritma korisiti dodatnu strukturu $ACMF$ (eng. ACMF - Autocorrelation Memory Function) koja je prikazana na slici 5.12.

Autokorelaciona memorijska funkcija kod implementacije SBDD algoritama sa bočnim kretanjem se sastoji od sledećih elemenata:



SLIKA 5.12: Struktura autokorelacione memorijske funkcije kod implementacije SBDD algoritma sa bočnim kretanjem.

1. `original` predstavlja pokazivač na poddijagram funkcije $f(x)$ kod rekurzivnog izračunavanja autokorelacije preko SBDD (opisano u prethodnom poglavlju).

2. `shifted` predstavlja pokazivač na poddijagram funkcije $f(x \oplus \tau)$ kod rekurzivnog izračunavanja autokorelacije preko SBDD (opisano u prethodnom poglavlju).

3. `result` predstavlja vrednost rezultata autokorelacije između poddijagrama na koje ukazuju pokazivači `original` i `shifted`.

Sledeći algoritam ilustruje proceduru za rekurzivni obilazak SBDD-a kod SBDD algoritama sa bočnim kretanjem opisanu u pseudo-kodu.

Kod implementacije BDD paketa praksa je da se koristiti heš tabela *AutocorrelationTable* kako bi se sprečilo kreiranje više instanci autokorelacione memorijske funkcije $acmf$ za pamćenje vrednosti autokorelacije između dva poddijagrama u SBDD-u. Koristiti se jednostavan pristup za heširanje standardnim procedurama *Find* za pretraživanje heš tabele i procedura *Add* za smeštanje vrednosti autokorelacije u heš tabelu.

Čvorovi BDD-a predstavljeni su promenljivama p i q. U implementacionom smislu, rekurzivni obilazak SBDD i istog SBDD-a sa pomerajem se izvršava na sličan način kao i ITE operacija koja se koristiti u jezgru BDD paketa. U cilju izvršavanja autokorelacije nad poddijagramom i poddijagramom sa pomerajem važno je da se odredi veći indeks promenljive ova dva poddijagrama, označenog sa `max`. U slučaju da su poddijagrami terminalni čvorovi izvršava se AND operacija nad vrednostima terminala (`p.value*q.value`). Primenom naizmeničnog obilaska SBDD-a odozdo-nagore, redosled obilaska *then* i *else* poddijagrama ili *then* i *else* poddijagrama sa pomerajem je takođe važno i određeno je promenljivama `maxthen` i `maxelse`. U slučaju da se vrši obilazak poddijagrama sa pomerajem, izvršavanje obilaska SBDD-a je određeno vrednošću elementa čvora `q.f` koji služi za markiranje čvorova. Osim toga, distanca indeksa promenljivih između čvora i njegovih *then* i *else* sledbenika je određena vrednošću `r`. Distanca između indeksa promenljivih se u punoj meri iskorišćava za izračunavanje autokorelacije preko SBDD-a.

Primenom obilaska SBDD-a odozdo-nagore, u proceduri postoje dva rekurzivna poziva za *then* i *else* poddijagrame originalnog i sa pomerajem. Rezultat autokorelacije između dva

Algoritam 5.7 Pseudo-kod procedure za rekurzivni obilazak sa bočnim kretanjem

```
1: Traverse(p,q)
2: acmf.original ← p
3: acmf.shifted ← q
4: if acmf ∈ Find(AutocorrelationTable) then
5:    return acmf.result
6: end if
7: if p.index > q.index then
8:    max ← p.index
9: else
10:   max ← q.index
11: end if
12: if max = 0 then
13:   r ← p.value * q.value
14:   return r
15: else
16:   maxthen ← p.then.index
17:   maxelse ← p.else.index
18:   if q.f then
19:      qt ← q.else
20:      qe ← q.then
21:   else
22:      qt ← q.then
23:      qe ← q.else
24:   end if
25:   if qt.index > maxthen then
26:      maxthen ← qt.index
27:   end if
28:   if qe.index > maxelse then
29:      maxelse ← qe.index
30:   end if
31:   i ← max − maxthen − 1
32:   r ← 2^i * Traverse(p.then, qt)
33:   i ← max − maxelse − 1
34:   r ← r + 2^i * Traverse(p.else, qe)
35: end if
36: acmf.result ← r
37: acmf → Add(AutocorrelationTable)
38: return r
```

poddijagrama se smešta u promenljivu `r` a potom se preko autokorelacione memorijske funkcije `acmf` smešta u autokoralcionu heš tabelu.

Implementacija finalne procedure za izračunavanje totalnih autokorelacionih koeficijenata obilaskom SBDD-a je ilustrovana u sledećem pseudo-kodu.

Pomeraj ili distanca autokorelacionog koeficijenta (označenog sa u) je predstavljena ulaznom promenljivom `t`. Takođe se promenljiva `t` korisiti kao ulaz u proceduru `MarkNodes` koja služi za obeležavanje čvorova sa permutovanim labelama na granama u SBDD-u. Implementacija procedure `MarkNodes` je jednostavna i zbog toga je u ovom radu izostavljena. Promenljiva `outputs` označava broj izlaza višeizlazne prekidačke funkcije za koju

Algoritam 5.8 Pseudo-kod procedure za izračunavanje totalnih autokorelacionih koeficijenata

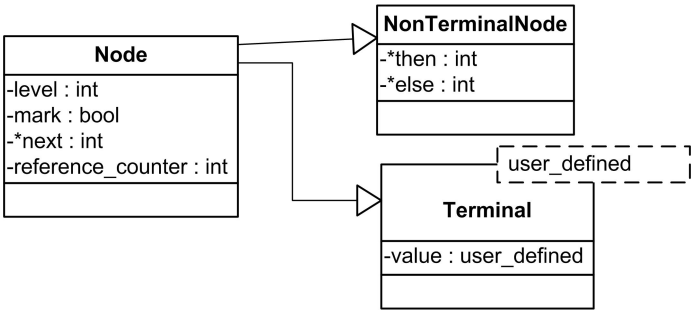
```
1: TotalAutocorrelation(t)
2: MarkNodes(t)
3: total ← 0
4: for i = 0 to i < outputs do
5:     total ← total+Traverse(root[i],root[i])
6:     i ← i + 1
7: end for
8: return total
```

se vrši izračunavanje totalnih autokorelacionih koeficijenata. Pokazivači na čvorove korene izlaza SBDD-a su smešteni u niz pokazivača (`root[i]` koji su tipa `Node`). Promenljiva `total` predstavlja vrednost totalnog autokorelacionog koeficijenta.

Realizacija potrebnih struktura podataka u jezgru BDD paketa i procedura za izračunavanje autokorelacije bočnim obilaskom SBDD-a na prograskom jeziku C++ se može naći u dodatku A.

5.3 Implementacija SMTBDD algoritama preko Wiener-Khinchinove teoreme i brze Walsh-ove transformacije i SMTBDD algoritma preko BDD paketa sa dinamičkim terminalnim čvorovima

Dijagram statičke strukture čvorova za implementaciju BDD paketa sa dinamičkim terminalnim čvorovima je prikazan na slici 5.13.



Slika 5.13: Struktura čvora kod implementacije SBDD algoritma preko BDD paketa sa dinamičkim terminalnim čvorovima.

Ova struktura čvora je veoma slična strukturi koja se korisiti u većini BDD paketa. Razlika u odnosu na većinu paketa je mehanizam za kreiranje dinamičkih terminalnih čvorova. Dinamički terminalni čvorova se korisiti kod implementacije SMTBDD algoritama

za izračunavanje kompletnih autokorelacionih koeficijenta preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije za prekidačke funkcije sa velikim brojem ulaznih promenljivih. Opis strukture čvora je realizovan korišćenjem UML model dijagrama.

Klasa *Node* se korisit za predstavljanje neterminalnih ili terminalnih čvorova. To je objektno-orjentisana klasa koja sadrži tri atributa člana: *level*, *next*, *mark* i $referencecounter$. Član *level* označava indeks promenljive koji određuje na kom nivou u dijagramu se nalazi čvor i ona je celobrojnog tipa. Pokazivač *next* povezuje čvorove koji pripadaju istom nivou u dijagramu, odnosno koji imaju istu vrednost indeksa promenljive. Označavanje posećenosti čvorova kod raznih algoritama za obilazak dijagrama se postiže preko člana *mark*. Član $referencecounter$ se implementira za jednostavno recikliranje čvorova i on korisiti celobrojni tip podataka. Neterminalni čvor je takođe objektno-orijentisana klasa izvedena iz klase *Node* koja sadrži atribute članove: *then* i *else* pokazivače čvorova potomaka. Terminalni čvor je takođe klasa izvedena iz klase čvor i sadrži atribut član *value*. Atribut član *value* smešta celobrojnu vrednost terminalnog čvora.

Objektno-orijentisani programski jezici dozvoljavaju definiciju šablonskih (generičkih) klasa za predstavljanje članica klase koji mogu da budu bilo kog tippodatka (uključujući i korisničko-definisane tipove podatka). cilju implementacije dinamičkih terminalnih čvorova, klasa $TerminalNode$ je implementirana kao šablonska klasa. Članica ove klase *value* korisiti šablonski tip podataka i to može biti bilo koji tip podataka. Glavni program će kasnije deklarisati i koristiti članicu klase *value*. Ova implementacija za tip podataka članice klase *value* postavlja korisničko-definisanu implementaciju klase $userdefined$ koja podržava rad sa velikim celobrojnim vrednostima, gde se dužina celobrojnih vrednosti može postavljati preko parametra. Ova implementacija korisiti BDD operacije koje zahtevaju operacije definisane i za korisničko-definisanu implementaciju tipa $userdefined$. Zbog toga, BDD operacije se moraju implemetiratirati kao šablonski predefinisane operatorske funkcije.

Za korisničko-definisanu implementaciju klase koja podržava rad sa velikim celobrojnim vrednostima takođe je razvijena šablonska klasa $BitVector < size >$, gde parametar *size* predstavlja veličinu binarnog vektora (velike celobrojne vrednosti) i on predstavlja parametar šablonske klase $BitVector$. Ova implementacija korisiti BDD operacije koje zahtevaju operacije nad binarnim vektorom, klasa $BitVector$ mora da podržava predefinisane operatorske funkcije za dodelu vrednosti, relacije, ekvivalentnost i aritmetičke operacije. Implemetacija klase $BitVector$ korisiti standardne programske tehnike za strukture podataka.

Primer implementacije gde se preko parametra postavlja veličina SMTBDD terminalnih čvorova čija maksimalna vrednost terminalnog čvora može da bude 2^{128} (opisanan u C++ programskom jeziku) prikazan je na slici:

```
bddmanager<BitVector<128>> manager;
smtbdd<BitVector<128>> bdd(manager);
```

Slika 5.14: Primer implementacije postavlja veličina SMTBDD terminalnih čvorova u BDD paketu sa dinamičkim terminalnim čvorovima.

U smislu implementacije BDD paketa, uobičajno je da se korisiti klasa *bddmanager* za inicijalizaciju BDD paketa. Nakon inicijalizacije potrebna je defiicija *bdd* objekta koji upravlja radom SMTBDD-a.

Poglavlje 6

Eksperimentalni rezultati

U ovom poglavlju biće opisano eksperimentalno testiranje efikasnosti implementacija postojećih i u ovom radu predloženih algoritama za izračunavanje autokorelacionih koeficijenata za slučaj izračunavanja pojedinačnih i kompletnih koeficijenata. Svi testirani algoritmi su u prethodnim poglavljima detaljno opisani. Za predložene algoritme u ovom radu u prethodnom poglavlju su opisane i njihove moguće implementacije. Svi algoritmi su testirani na skupu standardnih benčmark funkcija. Osim toga, u ovom poglavlju su diskutovane prostorne i vremenske statistike dobijenih eksperimentalnih rezulztata testiranja algoritama i izvršena je njihova uporedna analiza.

U zadnjih 20 godina pokrenuto je bilo nekoliko inicijativa za grupisanje benčmark funkcija u oblasti logičke sinteze u benčmark pakete za evaluaciju CAD algoritama. Istraživanja u pravcu kreiranja benčmark funkcija izvode se u industriji i na univerzitetima. U industriji se kao benčmark funkcije najčešće koriste realna logička kola. Međutim, benčmark funkcije u industriji, u većem broju slučajeva, nisu javno dostupne ili se čuvaju kao poslovna tajna [**?**]. Najveći broj benčmark funkcija na univerzitetima potiće sa naučnih konferencija. Na primer, MCNC.91 [**?**] i IWLS.05 [**?**] benčmark paketi potiču sa naučnih konferencija na temu logičke sinteze. MCNC.91 benčmark paket sadrži standardizovane biblioteke koje sadrže ceo spektar benčmark funkcija, odnosno od najprostijih do najsloženijih logičkih kola koja se najčešće upotrebljavaju u industriji. MCNC paket je veoma popularan u istraživanjima na univerzitetima. IWLS.05 benčmark paket sadrži raznolike benčmark funkcije prikupljenih iz više izvora: sa prethodo održanih naučnih konferencija, iz open-source benčmark zajednice i iz industrije. Namena ovog paketa je širok spektar primene. Benčmark paketi koji potiču sa univerziteta se široko upotrebljavaju, pošto su svi javno dostupni, a njihova distribucija je podržana i od strane brojnih open-source zajednica. Na primer, OpenCores [**?**] je najpoznatija open-source zajdenica koja distribuira benčmark pakete iz oblasti logičke sinteze.

Sve benčmark datoteke korišćene u ovom radu koriste Espresso-MV ili PLA format zapisa prekidačkih funkcija [**?**]. Svaka linija u PLA zapisu prekidačke funkcije predstavlja jednu vrstu u dvonivovsku reprezentaciju binarnih (ili višeznačnih) Bool-ovih funkcija i kao rezultat daje minimalnu ekvivalentnu reprezentaciju. U datoteci se mogu pojaviti i ključne reči kojima se daju dodatne informacije vezane za sam tip formata. Najvažnije ključne reči bez kojih prekidačka funkcija ne može da se opiše su: ".i"(specificira broj ulaznih promenljivih), ".o"(specificira broj izlaznih funkcija), ".p"(specificira broj proizvoda koji po jedan u liniji slede iza ove ključne reči) i ".e"(označava kraj PLA opisa). Primeri nekih karakterističnih prekidačkih funkcija zapisanih u PLA formatu su date u B.

6.1 Eksperimentalno testiranje postojećih algoritama za izračunavanje autokorelacionih koeficijenata

U prethodnim poglavljima detaljno su opisani i diskutovani postojeći algoritmi za izračunavanje pojedinačnih i kompletnih autokorelacionih koeficijenata. Ovo potpoglavlje prikazuje eksperimente koji se baziraju na ovim algoritmima. Ovi eksperimenti su prethodno objavljeni u radu [**?**]. U prvom delu je izvršena eksperimentalna analiza četiri postojeća algoritma za izračunavanje pojedinačnih autokorelacionih koeficijenata: algoritam iscrpljivanja, algoritam preko disjunktnih kubova, BDD algoritam sa permutovanim labelama i BDD rekurzivni algortiam. U drugom delu su analizirani algoritmi za izračunavanje kompletnih koeficijenata: algoritam iscrpljivanja i algoritam preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije.

U cilju eksperimentalnog testiranja u radu [**?**] je izvršena implementacija svih postojećih algoritama u C++ programskom jeziku. Pošto neki algoritmi za izračunavanje autokorelacionih koeficijenata zahtevaju izračunavanja gde se kao struktura podataka koriste dijagrami odlučivanja, za implementaciju ovih metoda neophodna je bila sinteza softverskog alata koji pokriva osnovni skup funkcionalnosti za rad sa dijagramima odlučivanja.

Za eksperimentalno testiranje performansi postojećih algoritama korišćene su standardni benčmark paketi MCNC.91 i IWLS.05. U tabeli 6.1 je prikazana lista benčmark višeizlaznih funkcija sa njihovim osnovnim karakteristikama (broj ulaza, broj izlaza, broj kubova) koja je izabrana za eksperimentalno testiranje postojećih algoritama za izračunavanje pojedinačnih autokorelacionih koeficijenata.

Benčmark funkcije su sortirane u rastućem redosledu u odnosu na broj promenljivih (ulaza) preklidačke funkcije. U ovom skupu funkcija ima 38 funkcije. Broj ulaza ovih

Tabela 6.1: Lista benčmark funkcija sa njihovim osnovnim karakteristikama

Ime benčmark funkcije	Broj ulaza	Broj izlaza	Broj kubova
xor5	5	1	16
rd53	5	3	32
squar5	5	8	32
bw	5	28	87
con1	7	2	9
rd73	7	3	141
inc	7	9	34
5xp1	7	10	75
sqrt8	8	4	40
rd84	8	4	256
misex1	8	7	32
9sym	9	1	87
clip	9	5	167
apex4	9	19	438
sao2	10	4	58
ex1010	10	10	1024
alu4	14	8	1028
table3	14	14	175
misex3c	14	14	305
misex3	14	14	1848
b12	15	9	431
t481	16	1	481
pdc	16	40	2810
spla	16	46	2307
table5	17	15	158
duke2	22	29	87
cordic	23	2	1206
cps	24	109	654
vg2	25	8	110
misex2	25	18	29
apex2	39	3	1035
seq	41	35	1459
apex1	45	45	206
apex3	54	50	280
e64	65	65	65
apex5	117	88	1227
ex4p	128	28	620
o64	130	1	65

funkcija kreće se u opsegu od 7 do 128, broj izlaza u opsegu 1 do 109, broj kubova od 9 do 2810 što pokazuje da su u ovom eksperimentu korišćene prekidačke funkcije sa različitim osnovnim karakteristikama. Eksperimentalno testiranje je izvršeno na platformi sa procesorom PC Pentium IV na 1.4 GHz i veličinom operativne memorije od 256 MB. Veličina jedinstvene tablice i tablice operacija, prilkiom korišćenja BDD paketa, su ograničene na 65535 ulaza. Upotreba sakupljača otpada kod BDD paketa je ograničena maksimalnim brojem čvorova od 200000. Rezaltati testiranja algoritama koji se baziraju na BDD-u uključuju i vreme konstruisanja odgovarajućih BDD-a. Rezulati testiranja su prikazani u tabelama 6.2 i 6.3. Sva vremena su data u sekundama. Crtice u odgovarajućim kolonama u tabelama označavaju neuspeh kod izvršavanja algoritma.

U tabeli 6.2 je prikazana uporedna statistika vremena izvršavanja postojećih algoritma za izračunavanje pojedinačnih autokorelacionih koeficijenata na skupu standardnih benčmark funkcija (algoritam iscrpljivanja, algoritam preko disjunktnih kubova, BDD algoritam sa permutovanim labelama i BDD rekurzivni algoritam).

U drugoj koloni tabele 6.2 je prikazana statistika vremena za izračunavanje pojedinačnog autokorelacionog koeficijenta preko definicije autokorelacije (algoritmom iscrpljivanja). Ovaj algoritam je uspešno izračunao autokorelacioni spektar za 30 od 38 funkcija. Ovde se može primetiti da ovaj algoritam daje relativno dobre rezultate za prekidačke funkcije koje nemaju više od 30 ulaznih promenljivih, dok za prekidačke funkcije sa više od 30 ulaznih promenljivih nije u mogućnosti da izračuna autokorelacioni koeficijent. Razlozi neuspešnih izračunavanja je veliki broj memorijskih lokacija potreban za izračunavanje autokorelacionog koeficijenata.

U trećoj koloni tabele 6.2 je prikazana statistika vremena za izračunavanje pojedinačnog autokorelacionog koeficijenta korišćenjem algoritma preko disjunktnih kubova. Ovaj algoritam je izvršio uspešna izračunavanja autokorelacionih koeficijenata za 33 od 38 funkcija. Ovde se može primetiti da ovaj algoritma daje relativno dobre rezultate za benčmark prekidačke funkcije koje nemaju više od 2400 disjunktnih kubova, dok za prekidačke funkcije sa više od 2400 disjunktnih kubova nije u mogućnosti da izvrši izračunavanje. Razlozi neuspešnih izračunavanja su nemogućnost generisanja disjunktnih kubova (veliki broj disjunktnih kubova) ili veliki broj memorijskih lokacija potreban za izvršenje izračunavanja.

U četvrtoj koloni tabele 6.2 je prikazana statistika vremena za izračunavanje autokorelacionog koeficijenta preko BDD algoritma sa permutovanim labelama. Ovaj algoritam je izvršio uspešna izračunavanja autokorelacionih koeficijenata za 36 od 38 funkcija. Ovde se može primetiti da ovaj algoritm daje relativno dobre rezultate za sve prekidačke funkcije za koje je bilo moguće generisanje dijagrama odlučivanja. Osim toga se primećuje da u nekim slučajevima benčmark funkcija sa malim brojem promenljivih ovaj algoritam

pokazuje lošije rezultate u odnosu na algoritm iscrpljivanja i algoritm preko disjunktnih kubova (apex4, sao2, ex1010, alu4, seq, e64).

U petoj koloni tabele 6.2 je prikazana statistika vremena za izračunavanje autokorelacionog koeficijenta preko BDD rekurzivnog algortima. Ovaj algoritam je izvršio uspešna izračunavanja autokorelacionih koeficijenata za 36 od 38 funkcija. Slično kao kod prethodnog algoritma, ovde se može primetiti da ovaj algoritm daje relativno dobre rezultate za sve prekidačke funkcije za koje je bilo moguće generisanje dijagrama odlučivanja. Osim toga se primećuje da u odnosu na BDD algoritm sa permutovanim labelama u dva slučaja ovaj algoritam pokazuje lošije rezultate (apex2, seq), a u četiri slučaja bolje (sao2, pdc, apex1, apex5).

U tabeli 6.3 je prikazana uporedna statistika vremena izvršavanja postojećih algoritma za izračunavanje kompletnih autokorelacionih koeficijenata na skupu standardnih benčmark funkcija (algoritam iscrpljivanja, algoritam preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije).

U drugoj koloni tabele 6.3 je prikazana statistika vremena za izračunavanje kompletnih autokorelacionih koeficijenta preko definicije autokorelacije (algoritmom iscrpljivanja). Ovaj algoritam je izvršio uspešna izračunavanja autokorelacionih koeficijenata za 22 od 38 funkcija. Slično kao kod izračunavanja pojedinačnog koeficijenta, za prekidačke funkcije sa više od 17 ulaznih promenljivih algoritma nije u mogućnosti da izračuna autokorelacione koeficijente. Razlozi neuspešnih izračunavanja je veliki broj memorijskih lokacija potreban za izračunavanje i smeštanje autokorelacionih koeficijenata.

U trećoj koloni tabele 6.3 je prikazana statistika vremena za izračunavanje kompletnih autokorelacionih koeficijenta korišćenjem algoritma preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije. Ovaj algoritam je izvršio uspešna izračunavanja autokorelacionih koeficijenata za 28 od 38 funkcija. Slično kao kod izračunavanja preko algoritma iscrpljivanja, za prekidačke funkcije sa više od 24 ulaznih promenljivih algoritma nije u mogućnosti da izračuna autokorelacione koeficijente. Razlozi neuspešnih izračunavanja je veliki broj memorijskih lokacija potreban za izračunavanje i smeštanje autokorelacionih koeficijenata.

6.2 Eksperimentalno testiranje SBDD algoritma sa permutovanim labelama

Za evaluaciju predloženog SBDD algoritma sa permutovanim labelama za izračunavanje pojedinačnih autokorelacionih koeficijenata izvršena je implementacija ovog algoritma

TABELA 6.2: Statistika vremena izvršavanja postojećih algoritma za izračunavanje pojedinačnih autokorelacionih koeficijenata

Benčmark	alg. iscrpljivanja [s]	alg. preko disj. kubova [s]	BDD sa perm. labelama [s]	rekurzivni BDD [s]
xor5	0.000	0.000	0.000	0.000
rd53	0.000	0.000	0.000	0.000
squar5	0.000	0.000	0.000	0.000
bw	0.000	0.000	0.000	0.000
con1	0.000	0.000	0.000	0.000
rd73	0.000	0.000	0.000	0.000
inc	0.000	0.000	0.000	0.000
5xp1	0.000	0.000	0.000	0.000
sqrt8	0.000	0.000	0.000	0.000
rd84	0.000	0.020	0.000	0.000
misex1	0.000	0.000	0.000	0.000
9sym	0.000	0.010	0.000	0.000
clip	0.000	0.000	0.000	0.000
apex4	0.000	0.090	0.010	0.010
sao2	0.000	0.000	0.010	0.000
ex1010	0.000	0.080	0.010	0.010
alu4	0.000	1.543	0.010	0.010
table3	0.020	0.030	0.010	0.010
misex3c	0.020	-	0.010	0.010
misex3	0.020	0.330	0.010	0.010
b12	0.020	0.020	0.000	0.000
t481	0.000	1.181	0.000	0.000
pdc	0.170	4.176	0.010	0.000
spla	0.200	5.988	0.010	0.010
table5	0.150	0.081	0.010	0.010
duke2	1.450	0.020	0.010	0.010
cordic	0.180	-	0.000	0.000
cps	19.620	0.030	0.020	0.020
vg2	3.040	0.320	0.010	0.010
misex2	6.498	0.000	0.000	0.000
apex2	-	-	0.040	0.080
seq	-	0.431	1.092	4.560
apex1	-	-	0.471	0.300
apex3	-	5.077	-	-
e64	-	0.000	0.010	0.010
apex5	-	4.737	0.040	0.020
ex4p	-	0.340	0.010	0.010
o64	-	-	-	-

Tabela 6.3: Statistike vremena izvršavanja postojećih algoritma za izračunavanje kompletnih autokorelacionih koeficijenata

Benčmark	alg. iscrpljivanja [s]	alg. preko W-K. teoreme i FWT [s]
xor5	0.000	0.000
rd53	0.000	0.000
squar5	0.000	0.000
bw	0.000	0.000
con1	0.000	0.000
rd73	0.000	0.000
inc	0.000	0.000
5xp1	0.000	0.000
sqrt8	0.000	0.000
rd84	0.000	0.000
misex1	0.010	0.000
9sym	0.000	0.000
clip	0.030	0.000
apex4	0.090	0.010
sao2	0.070	0.000
ex1010	0.180	0.010
alu4	202.511	0.500
table3	416.629	1.012
misex3c	417.030	1.012
misex3	416.309	1.012
b12	1031,441	1.282
t481	317.651	0.231
pdc	-	12.107
spla	-	13.890
table5	-	10.626
duke2	-	133.893
cordic	-	18.806
cps	-	2069.692
vg2	-	-
misex2	-	-
apex2	-	-
seq	-	-
apex1	-	-
apex3	-	-
e64	-	-
apex5	-	-
ex4p	-	86
o64	-	-

na C++ programskom jeziku u okviru BDD paketa. Za istraživanje efikasnosti ovog algoritma bilo je neophodno implementirati i algoritam iscrpljivanja za izračunavanje pojedinačnih autokorelacionih. Oba algoritma su testirana na primerima izračunavanja totalnih autokorelacionih koeficijenata prvog reda višeizlaznih prekidačkih benčmark funkcija. Za eksperimentalno testiranje performansi ova dva algortima korišćene su standardni benčmark paketi MCNC.91 i IWLS.05. U tabeli 6.4 je prikazana lista benčmark višeizlaznih funkcija sa njihovim osnovnim karakteristikama (broj ulaza, broj izlaza, broj kubova) koja je izabrana za eksperimentalno testiranje algoritama. Oba algoritma su testirana na 33 benčmark funkcija.

Benčmark funkcije su sortirane u rastućem redosledu u odnosu na broj promenljivih (ulaza) preklidačke funkcije. Broj ulaza ovih funkcija kreće se u opsegu od 10 do 128, broj izlaza u opsegu 5 do 109, broj kubova od 45 do 2144 što pokazuje da su u ovom eksperimentu korišćene prekidačke funkcije sa različitim osnovnim karakteristikama. Eksperimentalno testiranje je izvršeno na platformi sa procesorom PC Pentium IV na 2.66 GHz i veličinom operativne memorije od 4 GB. Korišćenje operativne memorije za sve testove je ograničeno na veličinu od 2 GB. Veličina jedinstvene tablice i tablice operacija, prilkiom korišćenja BDD paketa, su ograničene na 65535 ulaza. Upotreba sakupljača otpada kod BDD paketa je ograničena maksimalnim brojem čvorova od 200000. Rezaltati testiranja algoritama koji se baziraju na SBDD-u uključuju i vreme konstruisanja odgovarajućih BDD-a. Rezulati testiranja su prikazani u tabeli 6.5. Sva vremena su data u sekundama. Crtice u odgovarajućim kolonama u tabeli označavaju neuspeh kod izvršavanja algoritma.

U drugoj koloni tabele 6.5 je prikazana statistika vremena za izračunavanje autokorelacionih koeficijenta prvog reda preko definicije autokorelacije (algoritmom iscrpljivanja). Ovaj algoritam je uspešno izračunao autokorelacione koeficijente 11 od 33 funkcija. Ovde se može primetiti da ovaj algoritam daje relativno dobre rezultate za prekidačke funkcije koje nemaju više od 22 ulaznih promenljivih, dok za prekidačke funkcije sa više od 22 ulaznih promenljivih nije u mogućnosti da izračuna koeficijente. Razlozi neuspešnih izračunavanja je veliki broj memorijskih lokacija potreban za izračunavanje koeficijenata.

U trećoj koloni tabele 6.5 je prikazana statistika vremena za izračunavanje autokorelacionih koeficijenta prvog reda preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije. Ovaj algoritam je izvršio uspešna izračunavanja autokorelacionih koeficijenata za sve benčmark funkcije sa vremenom izvršavanja od najviše 3 sekunde.

Tabela 6.4: Lista benčmark funkcija sa njihovim osnovnim karakteristikama

Ime benčmark funkcije	Broj ulaza	Broj izlaza	Broj kubova
ex1010	10	10	1081
clpl	11	5	20
alu1	12	8	20
misex3	14	14	1703
b12	15	9	431
al2	16	47	103
table5	17	15	158
in2	19	10	137
mark1	20	31	129
cc	21	20	45
duke2	22	29	87
cps	24	109	654
alupla	25	5	2144
bc0	26	11	479
x9dn	27	7	120
c8	28	18	79
chkn	29	7	154
exep	30	63	175
b3	32	20	234
b4	33	23	54
in3	35	29	75
jbp	36	57	166
signet	39	8	124
seq	41	35	336
ti	47	72	271
ibm	48	17	173
misg	56	23	75
e64	65	65	65
x7dn	66	15	622
x2dn	82	56	112
soar	83	94	529
mish	94	43	91
ex4	128	28	620

Tabela 6.5: Statistike vremena izvršavanja algoritma iscrpljivanja i SBDD algoritma sa permutovanim labelama za izračunavanje autokorelacionih koeficijenata prvog reda

Benčmark	alg. iscrpljivanja [s]	SBDD alg. sa permutovanim labelama [s]
ex1010	0.4	0.2
clpl	0.8	0.1
alu1	1.2	0.1
misex3	1.2	0.2
b12	1.5	0.2
al2	2.9	0.2
table5	2.2	0.2
in2	3.4	0.3
mark1	30.1	0.2
cc	33.1	0.3
duke2	84.2	0.3
cps	-	0.6
alupla	-	0.6
bc0	-	0.5
x9dn	-	0.2
c8	-	0.2
chkn	-	0.2
exep	-	0.4
b3	-	0.4
b4	-	0.5
in3	-	0.6
jbp	-	0.6
signet	-	0.4
seq	-	0.8
ti	-	0.9
ibm	-	0.7
misg	-	0.9
e64	-	2.0
x7dn	-	1.3
x2dn	-	1.6
soar	-	1.8
mish	-	2.1
ex4	-	2.4

6.3 Eksperimentalno testiranje SBDD algoritma sa bočnim kretanjem

Za evaluaciju predloženog SBDD algoritma sa bočnim kretanjem izvršeno je poređenje izračunavanja totalnih autokorelacionih koeficijenata prvog reda sa rekurzivnim BDD algoritmom za izračunavanje pojedinačnih autokorelacionih koeficijenata. Izvršene su implementacije rekurzivnog BDD algoritma i predloženoh SBDD algoritma sa bočnim kretanjem. Za implementaciju jezgra BDD paketa korišćene su sve osnovne preporuke za programiranje BDD paketa (jedinstvena tablica, tablica operacija, sakupljač otpada, itd.) Takođe su implementirane i određene procedure za izračunavanje totalnih autokorelacionih koeficijenta prvog reda. Sve implementacije koriste C++ programski jezik. Za eksperimentalno testiranje performansi ova dva algortima korišćene su standardni benčmark paketi MCNC.91 i IWLS.05. U tabeli 6.6 je prikazana lista benčmark višeizlaznih funkcija sa njihovim osnovnim karakteristikama (broj ulaza, broj izlaza, broj kubova) koja je izabrana za eksperimentalno testiranje algoritama. Oba algoritma su testirana na 33 benčmark funkcija. Testiranje algoritma je podeljeno u dve grupe u skladu sa osobinama benčmark funkcija (veličina SBDD-a i broj izlaza funkcije).

Eksperimentalno testiranje je izvršeno na platformi sa procesorom PC Pentium IV na 2.66 GHz i veličinom operativne memorije od 4 GB. Korišćenje operativne memorije za sve testove je ograničeno na veličinu od 2 GB. Veličina jedinstvene tablice i tablice operacija, prilkiom korišćenja BDD paketa, su ograničene na 8191 ulaza. Veličina autokorelacione heš tablice takođe je ograničena na 8191 ulaza. Upotreba sakupljača otpada kod BDD paketa je ograničena maksimalnim brojem čvorova od 200000. Rezaltati testiranja algoritama koji se baziraju na SBDD-u uključuju i vreme konstruisanja odgovarajućih BDD-ova ili SBDD-a. Sva vremena su data u sekundama. Rezulati testiranja su prikazani u tabelama 6.7 i 6.8 gde su prikazani rezultati kreiranja SBDD-a, vremena izračunavanja algoritma preko rekurzivnog BDD algortima i SBDD algoritma sa bočnim kretanjem.

Vremena izvršavanja algoritama u tabeli 6.7 su uređena u rastući redosled na osnovu broja SBDD čvorova. Poređenje vremena izvršavanja algoritama (treća kolona - rekurzivni BDD algoritam, četvrta kolona - SBDD algoritam sa bočnim kretanjem) ukazuje na značajniju prednost predloženog SBDD algoritma za benčmark funkcije sa velikim brojem čvorova u SBDD-u (gde je broj čvorova aproksimativno veći od 1000). Ova prednost je ilustrovana grafikom na slici 6.1. Ubrzanje u odnosu na veličinu SBDD dijagrama varira između 3 do 10 % za benčmark funkcije sa manjim brojem ulaza i virtuelno duplo za benčmark funkcije sa velikim brojem ulaza.

Tabela 6.6: Lista benčmark funkcija sa njihovim osnovnim karakteristikama

Ime benčmark funkcije	Broj ulaza	Broj izlaza	Broj kubova
b7	8	31	74
exps	8	38	196
ex5	8	63	256
apex4	9	19	438
ex1010	10	10	1024
alu4	14	8	1028
table3	14	14	175
misex3	14	14	1848
misex3c	14	14	305
alcom	15	38	47
b2	16	17	110
pdc	16	40	2810
spla	16	46	2307
al2	16	47	103
table5	17	15	158
opa	17	69	342
ts10	22	16	128
cordic	23	2	1206
cps	24	109	654
misex2	25	18	29
bcd	26	38	243
bcc	26	45	245
bca	26	46	301
mainpla	27	54	181
exep	30	63	175
b4	33	23	54
misj	35	14	48
in3	35	29	75
jbp	36	57	166
signet	39	8	124
xparc	41	73	551
ibm	48	17	173
e64	65	65	65
x2dn	82	56	224
soar	83	94	529
mish	94	43	91
apex5	117	88	1227
ex4	128	28	620

TABELA 6.7: Statistike vremena izvršavanja BDD rekurzivnog algoritma i SBDD algoritma sa bočnim kretanjem za izračunavanje autokorelacionih koeficijenata prvog reda posmatrano na osnovu veličine SBDD-a

Benčmark	veličina SBDD-a [broj čvorova]	rekurzivni BDD [s]	SBDD alg. sa bočnim kretanjem [s]
misj	58	0.015	0.021
cordic	80	0.607	0.089
b7	106	0.015	0.004
mish	131	0.031	0.047
misex2	140	0.001	0.016
in3	377	0.046	0.046
b4	512	0.062	0.047
jbp	550	0.046	0.072
ibm	835	0.107	0.109
table5	873	0.041	0.064
table3	941	0.124	0.072
apex4	1021	0.094	0.093
ex1010	1079	0.093	0.078
misex3	1301	0.499	0.281
alu4	1352	0.296	0.234
misex3c	1893	0.140	0.078
cps	2318	0.327	0.265
signet	2956	1.276	0.358
ts10	4291	1.154	0.312
b2	4454	1.123	0.405

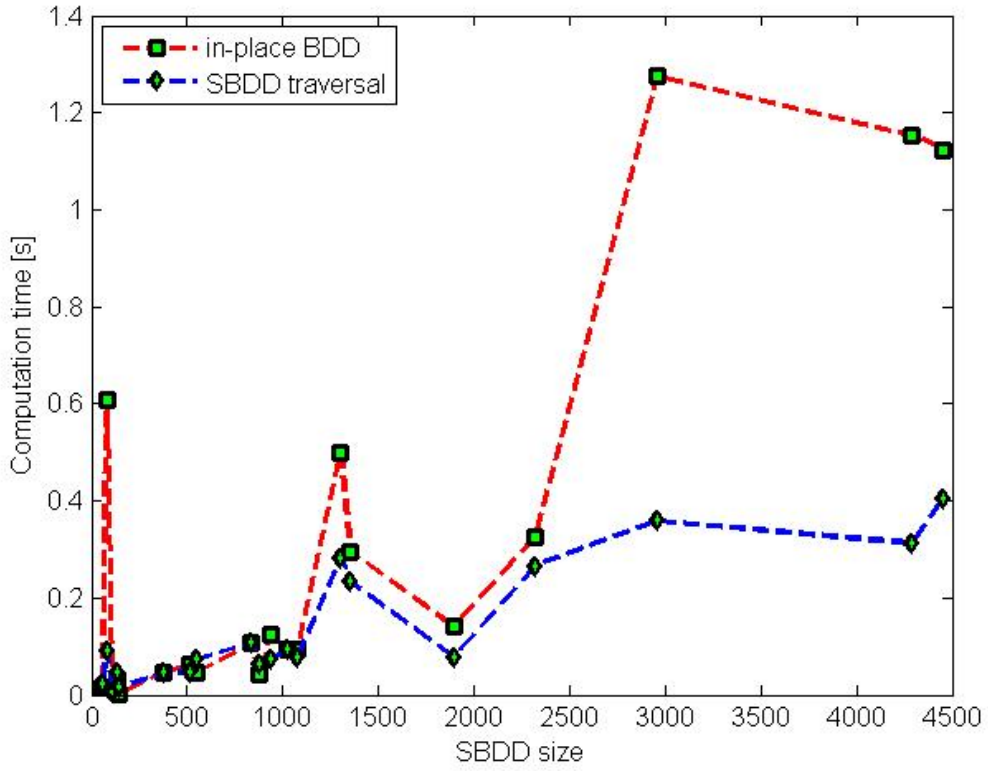
Vremena izvršavanja algoritama u tabeli 6.8 su uređena u rastući redosled na osnovu broja izlaza funkcija. Poređenje vremena izvršavanja algoritama (treća kolona - rekurzivni BDD algoritam, četvrta kolona - SBDD algoritam sa bočnim kretanjem) ukazuje na relativno malu prednost predloženog SBDD algoritma za benčmark funkcije sa velikim brojem izlaza. Ova prednost je ilustrovana grafikom na slici 6.2. Ubrzanje u odnosu na broj izlaza funkcije varira najviše do 10 %.

Tabela 6.8: Statistike vremena izvršavanja BDD rekurzivnog algoritma i SBDD algoritma sa bočnim kretanjem za izračunavanje autokorelacionih koeficijenata prvog reda posmatrano na osnovu broja izlaza funkcije

Benčmark	veličina SBDD-a [broj čvorova]	rekurzivni BDD [s]	SBDD alg. sa bočnim kretanjem [s]
ex4	1301	0.342	0.475
alcom	256	0.011	0.021
bcd	846	0.073	0.082
exps	573	0.040	0.041
pdc	705	0.423	0.513
mish 43	131	0.046	0.069
bcc	1121	0.125	0.111
bca	1433	0.135	0.133
spla	681	0.717	0.592
al2	594	0.031	0.040
mainpla	3308	2.074	0.421
x2dn	223	0.031	0.078
jbp	550	0.058	0.062
ex5	311	0.046	0.060
exep	902	0.046	0.093
e64	1446	0.124	0.202
opa	542	0.052	0,061
xparc	2752	0.764	0.532
apex5	2705	1.123	0.982
soar	995	0.358	0.265

6.4 Eksperimentalno testiranje SMTBDD algoritama za izračunavanje kompletnih autokorelacionih koeficijenata

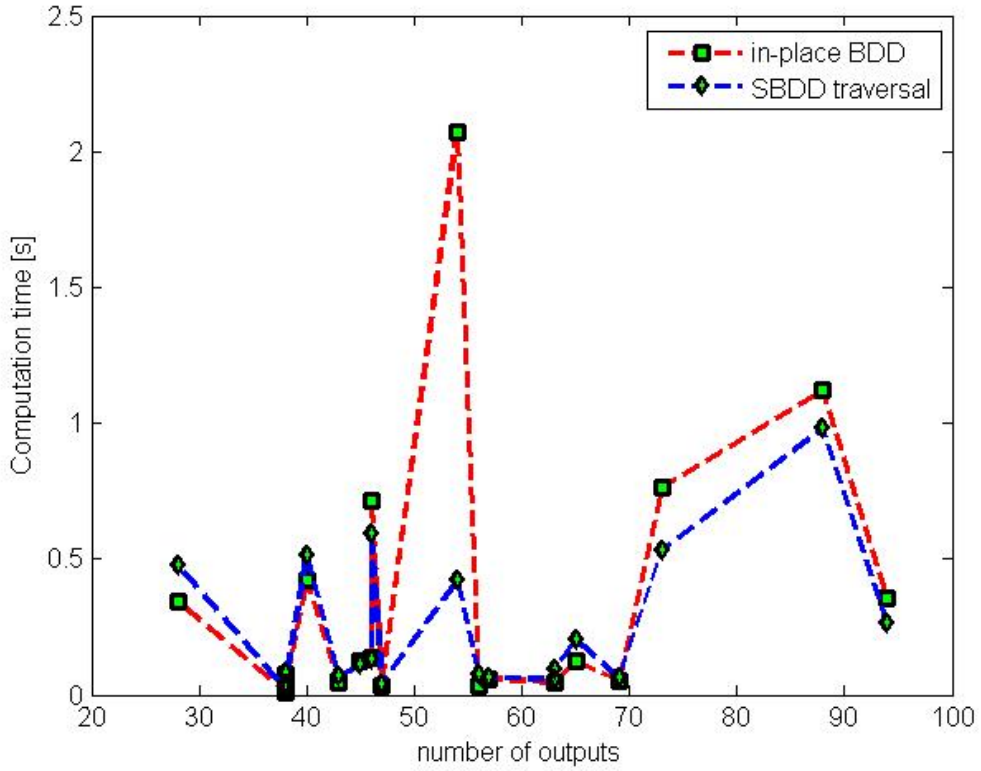
Za evaluaciju predloženog SMTBDD algoritma za izračunavanje kompletnih totalnih autokorelacionih koeficijenata izvršeno je izračunavanja totalnih autokorelacionih koeficijenata. Izvršena je implementacija predloženog SBDD algoritma sa dinamičkim čvorovima. Za implementaciju jezgra BDD paketa korišćene su sve osnovne preporuke za programiranje BDD paketa (jedinstvena tablica, tablica operacija, sakupljač otpada, itd.) Takođe su implementirane i određene procedure za izračunavanje kompletnih totalnih autokorelacionih koeficijenta. Sve implementacije koriste C++ programski jezik. Za eksperimentalno testiranje performansi ovog algortima korišćene su standardni benčmark paketi MCNC.91 i IWLS.05. U tabeli 6.10 je prikazana lista benčmark višeizlaznih funkcija



SLIKA 6.1: Grafičko predstavljanje vremena izvršavanja rekurzivnog BDD algoritama i SBDD alg. sa bočnim kretanjem u odnosu na broj čvorova u SBDD-u.

sa njihovim osnovnim karakteristikama (broj ulaza, broj izlaza, broj kubova) koja je izabrana za eksperimentalno testiranje algoritma. Algoritam je testiran na 19 benčmark funkcija.

Eksperimentalno testiranje je izvršeno na platformi sa procesorom PC Pentium IV na 2.66 GHz i veličinom operativne memorije od 4 GB. Korišćenje operativne memorije za sve testove je ograničeno na veličinu od 2 GB. Veličina jedinstvene tablice i tablice operacija, prilkiom korišćenja BDD paketa, su ograničene na 262139 ulaza. Veličina autokorelacione heš tablice takođe je ograničena na 262139 ulaza. Upotreba sakupljača otpada kod BDD paketa je ograničena maksimalnim brojem čvorova od 400000. Rezaltati testiranja algoritama koji se baziraju na SMTBDD-u uključuju i vreme konstruisanja odgovarajućih dijagrama. Rezulati testiranja su prikazani u tabelama 6.10 i 6.11. Tabela 6.10 prikazuje listu vremena izvršavanja algoritma i veličinu terminalnih čvorova. Sva vremena su data u sekundama. Veličina terminalnih čvorova je izražena u broju bitova potrebnih da se predstavi vrednost dinamičkog terminalnog čvora. Tabela 6.11 prikazuje listu memorijskih zahteva kod izvršavanja algoritma. Svi rezultati su dati u broju čvorova u dijagramu. Crtice u odgovarajućim kolonama u tabelama označavaju neuspeh kod izvršavanja algoritma i odnosi se na nedostatak potrebnog memorijskog



Slika 6.2: Grafičko predstavljanje vremena izvršavanja rekurzivnog BDD algoritama i SBDD alg. sa bočnim kretanjem u odnosu na broj izlaza funkcije.

prostora za smeštanje međurezultata koji se generišu prilikom izračunavanja.

Tabela 6.9: Lista benčmark funkcija sa njihovim osnovnim karakteristikama

Ime benčmark funkcije	Broj ulaza	Broj izlaza	Broj kubova
b4	33	23	54
in3	35	29	75
jbp	36	57	166
signet	39	8	124
apex2	39	3	1035
seq	41	35	336
apex1	45	45	206
ti	47	72	271
ibm	48	17	173
apex3	54	50	280
misg	56	23	75
e64	65	65	65
x7dn	66	15	622
x2dn	82	56	112
soar	83	94	529
mish	94	43	91
apex5	117	88	1227
ex4p	128	28	620
o64	130	1	65

TABELA 6.10: Statistika vremena izvršavanja algoritma i veličina terminalnih čvorova kod izračunavanja kompletnih totalnih autokorelacionih koeficijenata preko BDD paketa sa dinamičkim terminalnim čvorovima.

Benčmark	**Veličina terminala u SMTBDD-u [bits]**	**Vreme izračunavanja algoritma [s]**
b4	96	1.28
in3	96	1718.38
jbp	96	0.88
signet	96	-
apex2	96	-
seq	96	-
apex1	96	-
ti	96	33.18
ibm	128	113.27
apex3	128	-
misg	128	0.28
e64	160	0.49
x7dn	160	16035.81
x2dn	192	0.59
soar	192	7.52
mish	192	0.65
apex5	256	32.96
ex4p	288	360.26
o64	288	-

TABELA 6.11: Statistika potrebnog memorijskog prosotra kod izračunavanja kompletnih totalnih autokorelacionih koeficijenata preko BDD paketa sa dinamičkim terminalnim čvorovima.

Benč.	**veličina SMTBDD(f)**	**veličina SMTBDD(S_f)**	**veličina SMTBDD($S_f{}^2$)**	**veličina SMTBDD(B_f)**
	[broj neterminalnih čvorova / broj terminalnih čvorova]			
b4	512 / 2	5923 / 277	3831 / 156	1842 / 158
in3	377 / 2	13874 / 538	9563 / 335	7496 / 1618
jbp	550 / 2	6813 / 260	4290 / 157	1501 / 211
signet	2956 / 2	-	-	-
apex2	7102 / 2	-	-	-
seq	142321 / 2	44743 / 3993	24093 / 2013	-
apex1	28414 / 2	77535 / 3191	55024 / 2193	-
ti	6187 / 2	16437 / 950	8782 / 493	49947 / 2091
ibm	835 / 2	40264 / 517	23680 / 311	5085 / 1078
apex3	-	-	-	-
misg	107 / 2	2994 / 120	1748 / 72	377 / 73
e64	1446 / 2	3039 / 99	1610 / 50	1686 / 39
x7dn	863 / 2	73602 / 1046	53813 / 761	32217 / 6280
x2dn	223 / 2	5541 / 116	3283 / 68	657 / 107
soar	995 / 2	35346 / 465	16116 / 254	2584 / 453
mish	131 / 2	5595 / 99	3832 / 64	142 / 65
apex5	2705 / 2	73230 / 238	29499 / 122	4645 / 158
ex4p	1301 / 2	133953 / 1095	56278 / 621	4621 / 1149
o64	-	-	-	-

Poglavlje 7

Zaključci

U ovom poglavlju biće rezimirani doprinosi istraživanja u ovoj disertaciji. Ovde su posebno rezimirane performanse svih razvijenih algoritama za izračunavanje autokorelacionih koeficijenata prekidačkih funkcija preko dijagrama odlučivanja i preporuke za njihovu primenu. Takođe je izvršena diskusija nekoliko potencijalnih pravaca za budućih istraživanja vezanih za ovaj doktorski rad.

7.1 Poređenje algoritama po performansama i preporuke za njihovu primenu

U ovoj doktorskoj disertaciji su izvršani pregled, diskusija i dopuna neophodne teorijske osnove u skladu sa razmatranim problemom. Potom je izvršen pregled i analiza postojećih algoritama za izračunavanje autokorelacionih koeficijenata. U disertaciji su razvijeni novi generalizovani algoritami za izračunavanje pojedinačnih i kompletnih autokorelacionih koeficijenata višeizlaznih prekidačkih funkcija korišćenjem raznih vrsta dijagrama odlučivanja. Izvršena je implementacija postojećih i razvijenih algoritama i njihova komparativna analiza performansi na osnovu standardnog skupa benčmark funkcija.

U ovom radu su eksperimentalno testirani postojeći algoritmi za izračunavanje autokorelacionih koeficijenata prekidačkih funkcija preko vektora, kubova i dijagrama odlučivanja. Izvršeno je projektovanje i implementacija odgovarajućeg softverskog alata za izvršavanje postojećih algoritama. Namena ovog programskog paketa, pored izračunavnaja autokorelacionih funkcija, jeste da pruži pogodno okruženje za rad sa prekidačkim funkcijama relativno velikog broja promenljivih na skromnim hardverskim resursima. U tom cilju posebno su razmatrani problemi efikasnog korišćenja memorije sa očuvanjem brzine izračunavanja.

Iz rezultata testiranja postojećih algoritama, uočava se nemogućnost izračunavanja autokorelacionih koeficijenata preko vektora istinitosti zbog velikih memorijskih zahteva. Ovi nedostaci mogu biti prevaziđeni primenom algoritma za izračunavaje preko disjunktnih kubova ili dijagrama odlučivanja. Algoritam preko disjunktnih kubova pokazuje dobre rezultate u slučajevima da je broj disjunktnih kubova za datu prekidačku funkciju relativno mali. U suprotnom ovaj algoritam zahteva mnogo memorijskih resursa za pamćenje pomoćnih rezultata poređenja kubova. Algoritmi za izračunavanje autokorelacionih koeficijenata preko dijagrama odlučivanja pokazuju dobre rezultate u slučajevima da je broj čvorova u generisanim dijagramima odlučivanja za datu prekidačku funkciju relativno mali. U suprotnom ovi algoritmi, zbog velikih memorijskih zahteva za pamćenje čvorova i dodatnih memorijskih struktura (jedinstvena tablica, tablica računanja, tablica nivoa), nisu u mogućnosti da izvrše zahtevana izračunavanja. Algoritam za izračunavanje autokorelacije preko Wiener-Khinchin-ove teoreme i dijagrama odlučivanja je najefikasniji u slučaju izračunavanja kompletnih koeficijenata. Ovaj algoritam može da izračuna sve autokorelacione koeficijente odjednom. Međutim, memorijski zahtevi ovog algoritma su najveći jer se u procesu izračunavanja zahteva generisanje više pomoćnih dijagrama odlučivanja. Algoritmi preko dijagrama odlučivanja sa permutovaim labelama na granama i rekurzivnim obilaskom dijagrama su znatno sporiji, ali je njihova memorijska složenost proporcionalna memorijskoj složenosti predstavljanja zadate prekidačke funkcije. Algoritam sa permutovanim labelama na granama u procesu izračunavanja zahteva generisanje jednog pomoćnog dijagrama odlučivanja, dok se kod algoritma sa rekurzivnim obilaskom dijagrama ne generišu pomoćni dijagrami. U tom smislu, algoritmi za izračunavanje autokorelacionih koeficijenata preko dijagrama odlučivanja prevazilaze probleme eksponencijalne memorijske i vremenske zavisnosti algoritama za izračunanje preko vektora ili kubova.

Prostorna i vremenska kompleksnost algoritama za izračunavanje autokorelacionih koeficijenata je eksponencijalna u odnosu na broj promenljivih prekidačke funkcije. U slučajevima prekidačkih funkcija sa manjim brojem promenljivih ova eksponencijalna kompleksnost ne utiče na proces izračunavanja i tu se mogu koristiti postojeći algoritmi koji u svojoj osnovi izračunavanja izvršavaju preko nizova (algoritam iscrpljivanja, algoritam preko disjunktnih kubova, algoritam preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije).

Kod algoritama koji u svojoj osnovi imaju dijagram odlučivanja kao centralnu strukturu podataka razlikujemo dve grupe postojećih algoritama: algoritmi za izračunavanje pojedinačnih koeficijenata (BDD sa permutovanim labelama i BDD rekurzivni) i algoritam za izračunavanje kompletnih koeficijenata (MTBDD algoritam preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije). U slučaju algoritama preko dijagrama

odlučivanja, kompleksnost izračunavanja je proporcionalna veličini dijagrama odlučivanja. Stoga primena alternativnih pristupa za izračunavanje koeficijenata kao i primena različitih vrsta dijagrama odlučivanja poboljšava efikasnost izračunavanja.

U slučaju višeizlaznih prekidačkih funkcija izračunavanja autokorelacionih koeficijenata moraju da se ponavljaju za svaki izlaz posebno i onda se vrši sumiranje rezultata kako bi se dobili totalni autokorelacioni koeficijenti. Ova disertacija u detalje istražuje korišćenje raznih tipova razdeljenih dijagrama odlučivanja za efikasnije izračunavanje totalnih autokorelacionih koeficijenta. Svrha istraživanja je da učini efikasnijim i dostupnijom primenu autokorelacionih koeficijenata u praksi, što je posebno izraženo u slučaju sistema sa ograničenim resursima.

U ovom radu razvijen je efikasan algoritam (SBDD algoritam sa permutovanim labelama) za izračunavanje pojedinačnih totalnih autokorelacionih koeficijenata. Razvoj ovog algoritma se bazira na SBDD reprezentaciji višeizlaznih prekidačkih funkcija. Osim toga, ovaj algoritam predstavlja generalizaciju postojećeg algoritma za izračunavanje preko BDD-a sa permutovanim labelama na granama. Prednosti izračunavanja preko SBDD-a se postižu deljenjem izomorfnih podstabala u BDD-ima preko kojih su predstavljene višeizlazne prekidačke funkcijeje. Za istraživanje efikasnosti ovog algoritma bilo je neophodno implementirati predloženi algoritam kao nadogradnju BDD paketa. Izvršeno je poređenje efikasnosti predloženog algortima sa algoritmom iscrpljivanja za izračunavanje pojedinačnih autokorelacionih koeficijenata na primerima izračunavanja totalnih autokorelacionih koeficijenata prvog reda višeizlaznih prekidačkih benčmark funkcija. Iz eksperimentalnih rezultata se može zaključiti da algoritam iscrpljivanja nije primenljiv za benčmark funkcije sa više od 22 promenljive, s obzirom da dolazi do eksponencijalnog povećavanja zahteva za vremenom i prostornim resursima u zavisnosti od broja promenljivih prekidačke funkcije.

U ovom radu je takođe razvijen efikasan algoritam (SBDD algoritam sa bočnim kretanjem) za izračunavanje pojedinačnih totalnih autokorelacionih koeficijenata. Razvoj ovog algoritma se takođe bazira na SBDD reprezentaciji višeizlaznih prekidačkih funkcija. Osim toga, ovaj algoritam predstavlja generalizaciju postojećeg rekurzivnog ili "in-place"algoritma za izračunavanje pojedinačnih autokorelacionih koeficijenata preko BDD-a. Prednosti izračunavanja se baziraju na SBDD reprezentaciji višeizlazne prekidačke funkcije i omogućavanju izračunavanja totalne autokorelacije u jednom obilasku dijagrama. Algoritam koristi rekurzivnu strukturu autokorelacione matrice i SBDD-a. Zbog toga se sva izračunavanja izvršavaju samo na jednom SBDD-u koji zahteva minimalne memorijske resurse. Za istraživanje efikasnosti ovog algoritma bilo je neophodno implementirati algoritam kao nadogradnju BDD paketa. Izvršeno je poređenje efikasnosti predloženog algortima sa rekurzivnim BDD algoritmom za izračunavanje pojedinačnih

autokorelacionih koeficijenata na primerima izračunavanja totalnih autokorelacionih koeficijenata prvog reda višeizlaznih prekidačkih benčmark funkcija. Iz eksperimentalnih rezultata se može zaključiti da je algoritam izuzetno efikasan za benčmark funkcije sa velikim brojem čvorova u SBDD-u. Pokazalo se da je efikasnost ovog algoritma uglavnom uslovljena veličinom SBDD-a. Eksperimentalni rezultati su takođe pokazali da broj izlaza nema toliko uticaja na efikasnost algoritma kao što je veličina SBDD-a. Dodatni memorijski prostor neophodan za smeštanje rezultata izračunavanja autokorelacije između dva poddijagrama u SBDD-u je neizbežan. Ako višeizlazna prekidačka funkcija ima manji broj izomorfnih poddijagrama u SBDD-u, algoritam ne može da postigne pun potencijal. Ako u SBDD reprezentaciji ne postoje izomorfni poddijagrami, izračunavanje treba izvršavati postojećim rekurzivnim BDD algoritmom.

Kompleksnost algoritama za izračunavanje kompletnih totalnih autokorelacionih koeficijenata je i prostorno i vremenski eksponencijalna u odnosu na broj promenljivih višeizlazne prekidačke funkcije. U ovom radu je razvijen efikasni algoritam za izračunavanje ovih koeficijenata. Algoritam je baziran na SMTBDD reprezentaciji višeizlaznih prekidačkih funkcija. Osim omogućavanja procesiranja višeizlaznih funkcija sa velikim brojem promenljivih u uslovima sa ograničenim memorijskim resursima, SMTBDD pruža značajnu flesibilnost u procesu izračunavanja autokorelacionih koeficijenata. Izračunavanje se izvršava u spektralnom domenu korišćenjem Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije preko SMTBDD-a.

Međutim, izračunavanje pomoću SMTBDD algoritma preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije za prekidačke funkcije sa velikim brojem promenljivih zahteva memorisanje i izračunavanje velikih celobrojnih vrednosti u terminalnim čvorovima SMTBDD-a. Iz tog razloga korišćenje klasičnih BDD paketa koji imaju 32-bitne ili 64-bitne celobrojne terminalne čvorove je ograničeno na prekidačke funkcije sa manjim brojem promenljivih (do 16 ili 32 promenljive respektivno). Sa ovom motivacijom u ovom radu je razvijena dopuna SMTBDD algoritma preko Wiener-Khinchin-ove teoreme i brze Walsh-ove transformacije uvođenjem nove strukture podataka: SMTBDD-a sa dinamičkim terminalnim čvorovima. SMTBDD algoritam sa dinamičkim terminalnim čvorovima zahteva implementaciju specijalizovanog BDD paketa sa dinamičkim terminalnim čvorovima koji omogućava izračunavanje kompletnih autokorelacionih koeficijenata za funkcije sa velikim brojem promenljivih. Eksperimentalna evaluacija SBDD algoritma sa dinamičkim terminalnim čvorovima potvrđuje da implementacija ovog algoritma kao nadgradnje BDD paketa sa dinamičkim terminalnim čvorovima, u većini slučajeva dozvoljava izračunavanje kompletnih autokorelacionih koeficijenta za funkcije sa velikim brojem promenljivih. Za evaluaciju ovog algoritma korišćene su najzahtevnije benčmark funkcije kao što su: "o64" sa 130 ulaza, "ex4p" sa 128 ulaza i "apex5" sa 117 ulaza. Razlozi neuspeha izračunavanja koeficijenata su posledica ograničenog memorijskog prostora za

smeštanje SMTBDD reprezentacija funkcija, Wlash-ovog spektra ili autokorelacionog spektra. Porast kapaciteta memorijskih resursa trebalo bi u budućnosti da otkloni sve nedostatke kod primene SBDD algoritma sa dinamičkim terminalnim čvorovima.

7.2 Budući rad

Postoji mnogo oblasti u kojima rezultati predstavljeni u ovoj tezi mogu da se primene i prošire.

Kod SBDD algoritma sa permutovanim labelama na granama pokazano je da izračunavanje preko manipulacija sa labelama na granama ima velike potencijale. Budući rad bi bio usmeren u pravcu kreiranja novih algoritama koji u procesu izračunavanja koriste manipulacija sa labelama na granama, kao i korišćenje labela kod različitih tipova dijagrama odlučivanja.

Kod SBDD algoritma sa bočnim kretanjem pokazano je da uvođenjem novih pomoćnih struktura podataka u algoritam izračunavanja može da znatno poboljša efikasnost izračunavanja. Budući rad bi bio usmeren u pravcu primene sličnih struktura i kod drugih tipova algoritama baziranih na dijagramima odlučivanja kao i korišćenje ovih strukura kod različitih tipova dijagrama odlučivanja.

Koncepti predstavljeni u ovom radu su efikasni, međutim, moguća je dalja optimizacija implementacija u smislu vremenskih i prostornih zahteva. Predložene implementacije mogu biti modifikovane i primenjene kod izračunavanja konvolucije, korelacije, kroskorelacije i sličnih matematičkih operacija. Takođe, koncepti predstavljeni u ovom radu mogu biti uspešno primenjeni i kod izračunavanja drugih spektralnih transformacija za prekidačke funkcije sa velikim brojem ulaznih promenljivih.

Prilog A

Implementacija SBDD algoritma sa bočnim kretanjem na C++ jeziku

---

```
class Node{
    bool Mark;
    int Level;
    int Reference_counter;
    Node *Link;
}
class NonTerminalNode: public Node{
    Node *Then;
    Node *Else;
}
class TerminalNode: public Node{
    int Value;
}
```

```
class ACMF{
    Node *Original;
    Node *Shifted;
    int Result;
}
```

```
int Traverse(Node *node_original, Node *node_shifted)
{
    ACMF *new_acmf = new ACMF(node_original, node_shifted, 0);
    ACMF *exist_acmf = hash->Find(new_acmf);
    if (exist_acmf != NULL){
        delete new_acmf;
        return exist_acmf->Result;
    }
    if (node_original->Level > node_shifted->Level)
        maxlevel = node_original->Level;
    else
        maxlevel = node_shifted->Level;
    if (maxlevel == 0)
        return node_original->Value * node_shifted->Value;
    else{
        maxlevel_then = node_original->Then->Level;
        maxlevel_else = node_original->Else->Level;
        if (node2->Mark){
            node_shifted_then = node_shifted->Then;
            node_shifted_else = node_shifted->Else;
        }
        else{
            node_shifted_then = node_shifted->Else;
            node_shifted_else = node_shifted->Then;
        }
        if (node_shifted_then->Level > maxlevel_then)
            maxlevel_then = node_shifted_then->Level;
        if (node_shifted_else->Level > maxlevel_else)
            maxlevel_else = node_shifted_else->Level;
        r = maxlevel - maxlevel_then - 1;
        res = (int)(pow(2,r)) * Traverse(node_original->Then, node_shifted_then);
        r = maxlevel - maxlevel_else - 1;
        res += (int)(pow(2,r)) * Traverse(node_original->Else, node_shifted_else);
    }
    new_acmf->Result = res;
    hash->Add(new_acmf);
    return res;
}
```

```
int TotalAutocorrel(int t)
{
    MarkNodes(t);
    total = 0;
    for (int i=0; i<outputs; i++ )
        total += Traverse(root[i], root[i]);
    return total;
}
```

Prilog B

Primeri prekidačkih funkcija u PLA formatu

Benčmark funkcija: xor5.pla

```
.i 5
.o 1
.p 16
10000 1
01000 1
00100 1
11100 1
00010 1
11010 1
10110 1
01110 1
00001 1
11001 1
10101 1
01101 1
10011 1
01011 1
00111 1
11111 1
.e
```

Benčmark funkcija: sqrt8.pla

```
.i 8
.o 4
.type fd
.ilb v[7] v[6] v[5] v[4] v[3] v[2] v[1] v[0]
.ob sqrt[0] sqrt[1] sqrt[2] sqrt[3]
.p 40
0--11--1 1000
-00011-- 1000
0-1000-- 1000
1-0000-- 1000
1-11---- 1000
-011---1 1000
1-1-1--1 1000
-000-0-1 1000
0--11-1- 1000
111--1-- 1000
1000---- 1000
0101---1 1000
-11000-1 1000
-011--1- 1000
1-1-1-1- 1000
-000-01- 1000
0--111-- 1000
0101--1- 1000
-110001- 1000
1-1-11-- 1000
-011-1-- 1000
-0111--- 1000
0101-1-- 1000
01011--- 1000
111-1--- 1000
0-1--1-- 0100
11-1---- 0100
-000-1-- 0100
1-00-1-- 0100
0-1-1--- 0100
-0001--- 0100
1-001--- 0100
0-11---- 0100
111----- 0100
1000---- 0100
```

```
-0-1---- 0010
-01----- 0010
11------ 0010
-1------ 0001
1------- 0001
.e
```

Benčmark funkcija: 5xp1.pla

```
.i 7
.o 10
.p 75
---0--- ~~~~~~~1~
----10- 1~~~~~~~~
----010 ~1~~~~~~~
----101 ~1~~~~~~~
1---010 ~~1~~~~~~
01---01 ~~~1~~~~~
-00---1 ~~~~1~~~~
-011--0 ~~~~1~~~~
-100--0 ~~~~1~~~~
0-10--- ~~~~~1~~~
1-11--- ~~~~~1~~~
0101--- ~~~~~1~~~
1-00--- ~~~~~1~~~
100---- ~~~~~1~~~
-001--- ~~~~~~1~~
-1-0--- ~~~~~~1~~
-11---- ~~~~~~1~~
--01--- ~~~~~~~1~
--10--- ~~~~~~~1~
----111 ~~~~~~~~1
1---11- ~~~~~~~~1
00--1-0 1~~~~~~~~
1---011 1~~~~~~~~
-1--011 1~~~~~~~~
11---01 ~1~~~~~~~
1-1--01 ~1~~~~~~~
111-10- ~1~~~~~~~
11-110- ~1~~~~~~~
```

```
00---10 ~1~~~~~~~
0---001 ~~1~~~~~~
-00-001 ~~1~~~~~~
111-0-0 ~~1~~~~~~
11-10-0 ~~1~~~~~~
0---100 ~~1~~~~~~
-0--100 ~~1~~~~~~
--00100 ~~1~~~~~~
-111010 ~~1~~~~~~
00---10 ~~~1~~~~~
100--0- ~~~1~~~~~
0-11-01 ~~~1~~~~~
0111-0- ~~~1~~~~~
1-00-00 ~~~1~~~~~
111--1- ~~~1~~~~~
11-1-1- ~~~1~~~~~
00-0--1 ~~~~1~~~~
-111--1 ~~~~1~~~~
111---1 ~~~~1~~~~
11-1--1 ~~~~1~~~~
101---0 ~~~~1~~~~
-11111- ~~~~~~~~1
0--01-0 1~~~~~~~~
0-0-1-0 1~~~~~~~~
--11011 1~~~~~~~~
00-001- ~1~~~~~~~
000-01- ~1~~~~~~~
0--0-10 ~1~~~~~~~
0-0--10 ~1~~~~~~~
00-00-1 ~~1~~~~~~
000-0-1 ~~1~~~~~~
1---111 ~~1~~~~~~
-1--111 ~~1~~~~~~
001111- ~~1~~~~~~
0--01-0 ~~1~~~~~~
0-0-1-0 ~~1~~~~~~
11--1-1 ~~1~~~~~~
1-1-1-1 ~~1~~~~~~
0--0-10 ~~~1~~~~~
0-0--10 ~~~1~~~~~
10---00 ~~~1~~~~~
00-0-1- ~~~1~~~~~
000--1- ~~~1~~~~~
11---11 ~~~1~~~~~
```

```
1-1--11 ~~~1~~~~~~
01-0--0 ~~~~1~~~~~
010---0 ~~~~1~~~~~
.e
```

Biografija

Mr Miloš M. Radmanović je rođen 16.06.1976. godine u Nišu, sa adresom stalnog boravka u Nišu.

1991. godine završio je osnovnu školu *Sveti Sava* u Nišu. Za postignute uspehe u toku školovanja nagrađen je Vukovom diplomom. 1995. godine završio je gimnaziju *Svetozar Marković* u Nišu u specijalizovanom odeljenju za učenike posebno nadarene za matematiku i fiziku sa odličnim uspehom. U toku školovanja bio je nagrađivan diplomama na opštinskim, regionalnim i republičkim takmičenjima iz matematike, fizike i informatike. Nosilac je zlatne medalje iz oblasti fizike na Saveznoj smotri naučno-tehničkog stvaralaštva 1995. godine. Diplomirao je februara 2001. godine na Elektronskom fakultetu u Nišu na smeru za Računarsku tehniku i informatiku sa prosečnom ocenom 9,29 i ocenom 10 na diplomskom ispitu. Diplomski rad na temu *ASP programski jezik za kreiranje dinamičkih Web stranica* rađen je u okviru predmeta Programski jezici. Tokom studija, nosilac je studentskih nagrada 1996. i 2000. godine. Diplomski rad je nagrađen 2001. godine nagradom za najbolju praktičnu realizaciju. Zbog svog velikog interesovanja za naučnu oblast računarstva i informatike na fakultetu je bio autor mnogobrojnih projekata iz oblasti računarstva. Studiranje je bilo podržano i stipendijom Norveške Kraljevske Ambasade 2000 godine. Magistrirao je jula 2006. godine na Elektronskom fakultetu u Nišu na smeru za Računarsku tehniku i informatiku sa prosečnom ocenom 10.00 i ocenom 10 na magistarskom ispitu. Magistarski rad je bio na temu *Autokorelacija prekidačkih funkcija*. Doktorske studije je upisao 2007. godine na Elektronskom fakultetu u Nišu, smer za Računarsku tehniku i informatiku.

Miloš Radmanović zapošljen je na Elektronskom fakultetu u Nišu od 2001. godine kao asistent pripravnik na Katedri za računarstvo, a 2010. godine izabran je u zvanje asistent. Do 2003. godine bio je angažovan u procesu nastave na predmetima: Programiranje i Programski jezici. Tokom 2002. godine bio je angažovan u izvođenju specijalističkih komercijalnih kurseva na Elektronskom fakultetu u Nišu *Programski jezik C* i *Programski jezik C++*. Do 2007. godine bio je angažovan na predmetima: Uvod u računarstvo, Programiranje, Logičko projektovanje, Napredne tehnike za obradu slike, Napredne tehnike u logičkom projektovanju, Uvod u fazi logiku, Prepoznavanje uzoraka i Napredne tehnike za obradu signala. Trenutno je angažovan na predmetima: Algoritmi i programiranje, Logičko projektovanje, Osnovi analize signala i sistema, Programski jezici, Metodi i sistemi za obradu signala, Prepoznavanje uzoraka, Spektralne metode, Napredne metode za obradu slike, Napredne tehnike digitalne logike, Fazi logika i Sistemi za digitalnu obradu signala. Oblasti njegovog interesovanja tokom dosadašnjih naučnih istraživanja su: digitalna logika, logičko projektovanje, programski jezici i Internet tehnologije. Bio je učesnik na projektima pod pokroviteljstvom Ministarstva za nauku Republike Srbije: *Virtelne Web laboratorije za permanentno inženjersko znanje* i *Spektralne tehnike na konačnim grupama sa primenama u obradi signala i projektovanju digitalnih sistema*, a trenutno je angažovan na projektima: *Infrastruktura za elektronski podržano učenje u Srbiji* i *Razvoj novih informaciono-komunikacionih tehnologija korišćenjem naprednih matematičkih metoda sa primenama u medicini, telekomunikacijama, energetici, zaštiti nacionalne baštine i obrazovanju.* Njegov naučno-istraživački rad u oblasti digitalne logike i logičkog projektovanja bio je podržan

DAAD stipendijom u okviru Pakta za stabilnost jugoistočne Evrope i studijskim boravcima na Univerzitetu u Dortmundu, Nemačka, 2004., 2005. i 2006. godine.

Prilog 1.

IZJAVA O AUTORSTVU

Izjavljujem da je doktorska disertacija, pod naslovom

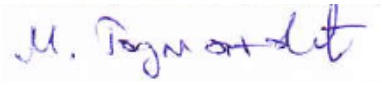
Razvoj algoritama za izračunavanje
autokorelacije prekidačkih funkcija
preko dijagrama odlučivanja

- rezultat sopstvenog istraživačkog rada,
- da predložena disertacija, ni u celini, ni u delovima, nije bila predložena za dobijanje bilo koje diplome, prema studijskim programima drugih visokoškolskih ustanova,
- da su rezultati korektno navedeni i
- da nisam kršio autorska prava, niti zloupotrebio intelektualnu svojinu drugih lica.

U Nišu, 11.12.2014. godine.

Autor disertacije: Miloš M. Radmanović

Potpis doktoranda:



Prilog 2.

IZJAVA O ISTOVETNOSTI ŠTAMPANE I ELEKTRONSKE VERZIJE DOKTORSKE DISERTACIJE

Ime i prezime autora: *Miloš Radmanović*

Studijski program: *Računarstvo i informatika*

Naslov rada:

Razvoj algoritama za izračunavanje
autokorelacije prekidačkih funkcija
preko dijagrama odlučivanja

Mentor: *prof. dr Radomir S. Stanković*

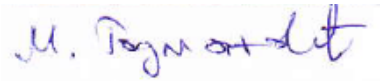
Izjavljujem da je štampana verzija moje doktorske disertacije istovetna elektronskoj verziji, koju sam predao za unošenje u **Digitalni repozitorijum Univerziteta u Nišu**.

Dozvoljavam da se objave moji lični podaci, koji su u vezi sa dobijanjem akademskog zvanja doktora nauka, kao što su ime i prezime, godina i mesto rođenja i datum odbrane rada, i to u katalogu Biblioteke, Digitalnom repozitorijumu Univerziteta u Nišu, kao i u publikacijama Univerziteta u Nišu.

U Nišu, 11.12.2014. godine.

Autor disertacije: Miloš M. Radmanović

Potpis doktoranda:



Prilog 3.

IZJAVA O KORIŠĆENJU

Ovlašćujem Univerzitetsku biblioteku "Nikola Tesla" da, u Digitalni repozitorijum Univerziteta u Nišu, unese moju doktorsku disertaciju, pod naslovom:

Razvoj algoritama za izračunavanje
autokorelacije prekidačkih funkcija
preko dijagrama odlučivanja

koje je moje autorsko delo.

Disertaciju sa svim prilozima predao sam u elektronskom formatu, pogodnom za trajno arhiviranje.

Moju doktorsku disertaciju, unetu u Digitalni repozitorijum Univerziteta u Nišu, mogu koristiti svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice (Creative Commons), za koju sam se odlučio:

1. Autorstvo – nekomercijalno – deliti pod istim uslovima

U Nišu, 11.12.2014. godine.

Autor disertacije: Miloš M. Radmanović

Potpis doktoranda: