Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Nové metódy redukcie rozhodovacích diagramov

Dizertačná práca

Študijný program: Aplikovaná informatika

Číslo študijného odboru: 2511

Názov študijného odboru: 9.2.9 Aplikovaná informatika

Školiace pracovisko: Ústav počítačových systémov a sietí

Vedúci práce: *prof. Ing. Ivan Kotuliak, PhD.*

Bratislava 2022 **Ing. Ján Lúčanský**

Nové metódy redukcie rozhodovacích diagramov

Autor:	Ing. Ján Lúčanský
Vedúci práce:	prof. Ing. Ivan Kotuliak, PhD.
Oponenti:	----------

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný odbor: 9.2.9. Aplikovaná informatika
Študijný program: Aplikovaná informatika

Autor:	Ing. Ján Lúčanský
Názov témy:	Nové metódy redukcie rozhodovacích diagramov
Vedúci dizertačnej práce:	prof. Ing. Ivan Kotuliak, PhD.
Dátum:	Apríl 2022

Práca sa zaoberá problematikou metód redukcie a optimalizácie rozhodovacích diagramov ako jednej zo základných dátových štruktúr pre Booleovu algebru. Analyzuje a opisuje súčasný stav v tejto oblasti, ako aj možnosti využitia v oblasti informatiky.

Práca poskytuje analýzu súčasného stavu, najvýznamnejšie problémy, motiváciu a témy pre ďalší výskum v tejto oblasti. Na základe analýzy je navrhnutá nová metóda redukcie so zachovaním hlavných vlastností rozhodovacích diagramov ako aj optimalizácia z pohľadu viacerých parametrov.

Navrhnutá metóda bola experimentálne overená na referenčných obvodoch s poukázaním na zvýšenú mieru redukcie v počte prvkov, syntézou nového, kompaktnejšieho rozhodovacieho diagramu. Navrhnutá metóda umožňuje uchovávať v jednom uzle informácie o viacerých premenných naraz a vďaka zachovaniu potrebných vlastností rozhodovacieho diagramu umožňuje jeho využitie v už existujúcich optimalizačných metódach.

Ďalším prínosom práce je optimalizácia viacerých parametrov nového, ako aj už existujúcich typov rozhodovacích diagramov. Navrhnuté sú spôsoby na výpočet spotreby uzla a je overený aj ich prínos v novej metóde. Práca tiež obsahuje optimalizáciu priemernej dĺžky výstupnej cesty rozhodovacích diagramov využívajúcich Davio dekompozíciu a ich rôzne formy.

Annotation

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Field of study: 9.2.9 Applied informatics
Doctoral program: Applied informatics

Author: Ing. Ján Lúčanský
Title: Novel reduction methods for decision diagrams
Advisor: prof. Ing. Ivan Kotuliak, PhD.
Date: April 2022

This thesis focuses on methods of reduction and optimization of decision diagrams as one of the fundamental data structures for Boolean algebra. It analyzes and describes the state of art in this area as well as their various uses in the field of informatics.

The work provides analysis of current state, problem domains, motivation and a proposal of future advancement in this field. Based on the analysis, the novel method of reduction with preservation of key attributes of decision diagrams is proposed as well as optimization of multiple parameters.

The proposed model was experimentally validated on the benchmark circuits by synthetizing the newer and more compact decision diagram with focus on reduction of the number of nodes. The proposed method allows use of mutliple variables in one node and thanks to preserving the necessary properties of decision diagrams, it allows for further use in other optimization methods.

Another benefit of the work proposals is optimization using multiple parameters in existing and newly proposed types of decision diagrams. Thesis introduces method for power consumption estimate of nodes and provides proof of its added value in proposed method. In addition, an average path length optimization is used for the previously mentioned types of diagrams and their various forms.

Poďakovanie

Zapracujem po DizP6.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam použitých pojmov a skratiek

ADD	Algebraický rozhodovací diagram (Algebraic Decision Diagram).
APL	Priemerná dĺžka cesty (Average Path Length).
***BMD**	Násobný binárny momentový diagram (Multiplicative Binary Moment Diagram).
B-funkcia	Booleova funkcia.
BDD	Binárny rozhodovací diagram (Binary Decision Diagram).
CNF	Konjuktívna normálova forma (Conjuctive Normal Form)
DD	Rozhodovací diagram (Decision Diagram).
DTL	Vektor dekompozičných pravidiel (Decomposition Type List).
EA	Evolučné algoritmy (Evolutionary Algorithms).
FDD	Funkcionálny rozhodovací diagram (Functional Decision Diagram).
FrDD	Voľný rozhodovací diagram (Free Decision Diagram).
K*BMD	Kroneckerov násobný momentový diagram (Kronecker Binary Moment Diagram).
KFDD	Kroneckerov funkcionálny rozhodovací diagram (Kronecker Functional Decision Diagram).
MMA	Modifikovaný memetický algoritmus (Modified Memetic Algorithm).
MBDD	Modifikovaný binárny rozhodovací diagram (Modified Binary Decision Diagram).
MDD	Rozhodovací diagram s viacerými vstupmi (Multi-valued Decision Diagram).
MTDD	Rozhodovací diagram s viacerými výstupmi (Multi-Terminal Decision Diagram).
MVDD	Rozhodovací diagram s viacnásobnými premennými (Multivariable Decision Diagram).
NP úplný problém	Typ problému so zložitosťou v nedeterministickom polynomiálnom čase.
NPN	Negácia-permutácia-negácia (Negation-Permutation-Negation).
O	Usporiadaný (Ordered). Slúži ako prívlastok k iným diagramom.
ODD	Usporiadaný rozhodovací diagram (Ordered Decision Diagram).
OPDD	Čiastočne usporiadaný rozhodovací diagram (Ordered Partial Decision Diagram).
pBMD	Pozitívny binárny momentový diagram (positive Binary Moment Diagram).
PSO	Optimalizácia rojom častíc (Particle Swarm Optimization).
PTL	Pass Transistor Logic

R	Redukovaný (Reduced). Slúži ako prívlastok k iným diagramom.
RODD	Redukovaný usporiadaný rozhodovací diagram (Reduced Ordered Decision Diagram). Podľa použitej dekompozície môže pribudnúť prívlastok binárny (B), funkcionálny (F) alebo Kroneckerov funkcionálny (KF).
RV	Reziduálna premenná (Residual Variable).
RViDD	Rozhodovací diagram s reziduálnou premennou (Residual Variable in Decision Diagram). Podľa použitej redukcie (Reduced - R), usporiadania (Ordered – O) alebo použitej dekompozície môže pribudnúť písmeno, možné variácie zahŕňajú: RViBDD, RViFDD, RViKFDD, RORViDD, RORViBDD, atď.
SAT	Z anglického SATisfiability – splniteľnosť.
TDD	Ternárny rozhodovací diagram (Ternary Decision Diagram).
TTL	Transistor-Transistor Logic.
ZBDD	Binárny rozhodovací diagram s potlačenou nulou (Zero-supressed Binary Decision Diagram).

1 Úvod

V oblasti informatických vied bol vždy kladený dôraz na rýchlosť a veľkosť navrhovaných obvodov, dátových štruktúr alebo kódu. Jednotlivé parametre je možné upravovať na rôznych úrovniach návrhu, či už sa jedná o logický obvod alebo iný abstrahovaný proces, čím vyššie sa dosiahne zlepšenie (napríklad redukcia veľkosti) v hierarchii návrhu, tým väčšmi sa prejaví a tým jednoduchšie prebieha syntéza na nižších úrovniach. Nápad zahrnúť rozhodovacie diagramy (Decision Diagrams - DDs) do oblasti informatiky a informatických vied sa objavil už dávno. Dnes široko rozšírené využitie DD začalo v [1] kde bola po prvýkrát predstavená sada algoritmov na syntézu a manipulovanie DD ako dátovej štruktúry. Od toho momentu prešli DD rozsiahlym vývojom, pričom bolo nájdených viacero spôsobov ich využitia spolu s viacerými rôznymi druhmi DD [2]. Niektoré zo spôsobov využitia zahŕňajú, ale nie su obmedzené len na, formálnu verifikáciu, syntézu logických obvodov, generovanie testov, klasifikačné metódy alebo sieťovú bezpečnosť. Jednou z oblastí, ktorá významne profitovala z využitia DD je návrh obvodov. S bežnými technológiami, ktoré sa pomaly približovali k ich fyzikálnym limitom a spolu s neustávajúcim dôrazom na vysokú rýchlosť a nízku spotrebu, potreba optimalizovania obvodov na úrovni návrhu sa stávala viac a viac dôležitou a DDs sú prirodzenou cestou ako to dosiahnuť. Dobrý príklad je možné nájsť v [3] kde boli použité Binárne rozhodovacie diagramy (Binary Decision Diagram - BDD) na syntézu multiplexorových stromov alebo v [4] kde boli BDDs použité na syntézu optických obvodov. Ďalší príklad ich neustávajúcej dôležitosti je možné nájsť v [16] kde sú DDs použité na návrh reverzibilných a kvantových obvodov. Problém spotreby obvodov stúpal na význame posledné roky a zvyčajne sa rieši na systémovej úrovni. V dnešných dňoch, kedy IoT zariadenia využívajú viac a viac senzorov, je nevyhnutné znížiť spotrebu obvodu čo najviac na každej úrovni návrhu [9]. Je nutné spomenúť, že DD nie sú jedinou alternatívou a existujú viaceré typy grafov, ktoré môžu byť použité na skúmanie problémov vhodnosti riešenia [17].

Využitím rôznych typov DD sa k dnešnému dňu zaoberal už rozsiahly výskum. Medzi najrozšírenejší typ patria už spomínané BDD, ktoré su založené na Shannonovej dekompozícii Booleovej funkcie. Ako bolo spomenuté v [5], existujú len 2 typy dekompozície, ktoré majú vplyv na redukciu počtu uzlov výsledného diagramu – Shannon a Reed-Mullerova dekompozícia, niekedy označovaná ako Davio dekompozícia. DD, ktoré využívajú Davio dekompozíciu, či už v pozitívnom alebo negatívnom tvare, sa nazývajú Funkčné DD (Functional DD - FDD) a kombinácia oboch typov dekompozícií vyúsťuje v Kroneckerove Funkčné DD (Kronecker Functional DD - KFDD). Aplikovaním redukčných techník a dodržiavaním určitého poriadia premenných pre vstupnú funkciu je možné dosiahnuť Redukovaný Usporiadaný DD (Reduced Ordered DD - RODD), ktorý sa podľa typu dekompozície označuje ako ROBDD, ROFDD alebo ROKFDDD. Všetky doteraz vymenované typy DD dodržiavajú jedno spoločné pravidlo – každá výstupná cesta od koreňa diagramu po terminálne uzly obsahuje rovnaké poradie premenných. Uvoľnením tohto pravidla je možné dostať nový typ diagramu nazývaný

voľný DD (Free DD - FrDD). Zapamätať si všetky skratky tohto druhu môže byť náročné a kontraintuitívne, veľmi trefne pomenované ako „*písmenková polievka*“ v [2]. Napriek tomu, potreba existencie rôznych typov DD stále pretrváva keďže každý typ, a najmä použitá dekompozícia, má rôzne výhody z pohľadu vstupnej funkcie. Experimentálne výsledky naznačujú, že BDDs sú vhodnejšie na redukciu kontrolných funkcií a KFDDs dosahujú lepšie výsledky pri symetrických dátových funkciách. Okrem toho existuje viacero typov logických brán a komponentov, ktoré môžu byť priamo reprezentované pomocou DD, napríklad uzly BDD môžu byť priamo nahradené 2:1 multiplexormi a použité na syntézu multiplexorových stromov. Niektoré pokročilé techniky s rôznymi bránami sa nachádzajú v [20] a [21]. Pre potreby tejto práce sa notifikácia redukovaného a usporiadaného diagramu považuje za vopred danú.

V súčasnosti sa objavovanie nových typov DD mierne spomalilo oproti minulosti, expanziu zažíva skôr oblasť využitia existujúcich typov, najmä BDD. S vývojom technológie našli BDD svoje uplatnenie napríklad pri návrhu pamäťových obvodov tvorených špecializovanými tranzistormi [23]. V čisto teoretickej oblasti boli voľné BDD použité na výpočet aritmetického spektra Booleových funkcií.

Jedným z hlavných problémov pre všetky typy DD je škálovanie. Zložitosť usporiadania pre vstupnú funkciu s n premennými je faktoriálna, vstupná Booleova funkcia obsahuje 2^n hodnôt a bolo preukázané, že sa s rastúcim počtom vstupov jedná o NP-úplný problém [11]. Pri voľných DD táto zložitosť ešte viac narastá na $n!\,(n-1)!\,\ldots 3!\,2!\,1!$. Aby bolo možné jednoznačne povedať, že zvolené poradie premenných je najvhodnejšie pre redukciu, je nutné otestovať celú množinu poradí. Na adresovanie škálovania DD bolo vyskúšaných a využitých viacero metód a heuristík, od statického usporiadania premenných po komplexné usporiadacie algoritmy. Zaujímavé výsledky boli dosiahnuté evolučými algoritmami (Evolutionary Algorithms - EA), ktoré môžu byť použité nielen na redukciu počtu uzlov v DD, ale zároveň aj na optimalizáciu iných parametrov, napríklad dĺžku priemernej cesty (Average Path Length - APL) alebo spotrebu navrhovaného obvodu. APL je kritické pre oneskorenia v obvode kde väčšina výstupných ciest má rovnakú pravdepodobnosť priechodu. Tomuto oneskoreniu môže byť napríklad priradená najvyšia priorita z určených parametrov na všetkých úrovniach abstrakcie [18].

Oblasť s veľkým potenciálom priniesť ďalšie zlepšenie oproti existujúcim metódam sú práve voľné DD. Skúmaním všetkých možných poradí premenných pre každú časť diagramu samostatne sa docieli maximálna možná redukcia diagramu bez straty informačnej hodnoty. Problémom však zostáva exponenciálne rastúca zložitosť takéhoto výpočtu pre každú premennú definovanú na vstupe. Oblasťou výskumu s vysokým potenciálom môže byť zmenšenie veľkosti prehľadávanej množiny na skupinu s určitými vlastnosťami. Existujúce redukčné pravidlá sú schopné redukovať rovnaké, t.j. úplne identické časti diagramu, žiadna z nich však nie je schopná redukovať ekvivalentné časti, teda tie, ktoré po malej modifikácii disponujú identickou štruktúrou alebo inými identickými vlastnosťami. Prehľadávanie ekvivalentných funkcií pri

rôznom usporiadaní premenných môže priniesť zaujímave výsledky z pohľadu viacerých parametrov.

Práca predstavuje optimalizáciu KFDD z pohľadu počtu uzlov, APL a spotreby, pričom sú odvodené rovnice pre spotrebu Davio uzla na základe Shannonovho uzla. Z pohľadu prínosu k teoretickému aparátu v informatických vedách práca predstavuje aj nový typ DD - MVDD (Multi-Variable DD), ktorý umožňuje v jednom uzle uchovávať viac premenných naraz so zachovaním hlavných charakteristík uzlov z BDD alebo FDD. Jedná sa o typ voľného DD, kde sa navrhovaná heuristika zaoberá iba podmnožinou všetkých usporiadaní premenných. Využíva prínos reziduálnej premennej (Residual Variable - RV) použitej na najnižšiu úroveň DD. Hlavným prínosom MVDD je možnosť redukovania nielen identických, ale aj ekvivalentných častí diagramu. Na navrhovanom DD bolo overených viacero typov optimalizácie, ako je napríklad použitie EA na usporiadanie vstupných premenných, poradie dekompozícií pre jednotlivé premenné, na vyhodnotenie APL a teoretickej spotreby diagramu (obvodu). Navrhovaná metóda bola overená na voľne dostupných testovacích obvodoch, ktoré sú v danej oblasti štandardom a porovnaná s existujúcimi riešeniami pre najrozšírenejší typ BDD. Práca sa zaoberá iba úplne definovanými vstupnými Booleovskými funkciami pokiaľ nie je špecifikované inak.

Práca je členená nasledovne:

Na začiatku dokumentu sa nachádza úvod do problematiky, prehľad dokumentu a použité skratky. Druhá kapitola opisuje základné definície a pojmy, ktoré súvisia s rozhodovacími diagramami a ich syntézou pomocou reziduálnej premennej. V tejto časti sú tiež analyzované redukčné a optimalizačné metódy a heuristiky, vrátane rôzneho prístupu k redukcii podľa viacerých parametrov. Nasledujúca, tretia časť dokumentu, uvádza tézy dizertačnej práce. Od tohto bodu začína samotný originálny prínos tejto práce. Vo štvrtej časti je opísaná optimalizácia KFDD pomocou zvolených metód a sledovaných parametrov. V piatej kapitole je uvedený návrh heuristiky pre voľné rozhodovacie diagramy, ktorá je integrovaná do KFDD a z ktorej vyplýva hlavný prínos práce – rozhodovací diagram s viacnásobnými premennými (MVDD). Táto kapitola sa venuje definovaniu nového druhu rozhodovacích diagramov a obsahuje aj ukážku použitia nového typu diagramu pri optimalizácii, odhade dĺžky výstupnej cesty a spotreby. V nasledujúcej kapitole sú realizované experimentálne výsledky na referenčných testovacích obvodoch. Na záver sú zhrnuté prínosy práce a stanovenie ďalšieho smerovania v danej oblasti, nasledované zoznamom použitej literatúry. Prílohy práce obsahujú publikácie vyvodené z výsledkov dosiahnutých v tejto práci.

2 Rozhodovacie diagramy a ich typy

Hoci existuje mnoho spôsobov využitia rozhodovacích diagramov, hlavnou inšpiráciou pre túto prácu je oblasť, ktorá z ich používania vyťažila pravdepodobne najviac – návrh logických obvodov. Obvody sa skladajú z viacerých logických členov, medzi ktoré patria napríklad multiplexory, hradlá AND, OR, XOR alebo iné, zložitejšie prvky. Syntézou logických obvodov sa rozumie proces, pri ktorom sa zo známeho predpisu, napr. Booleovej funkcie, získava logický obvod. Syntéza nie je priamočiara a môže byť optimalizovaná z pohľadu rôznych parametrov, medzi ktoré patria:

- Veľkosť obvodu – je možné ju optimalizovať odstránením redundantných častí obvodu so zachovaním funkcionality kompletného, neredukovaného obvodu.
- Spotreba obvodu – je možné ju optimalizovať výberom vhodnej technológie. Spotreba sa často rozdeľuje na 2 druhy – statická a dynamická. Dynamická spotreba je ovplyvnená počtom členov obvodu, takže je možné dosiahnuť jej zlepšenie redukovaním veľkosti obvodu alebo preusporiadaním vstupných premenných [24].
- Rýchlosť obvodu – na jej zlepšenie majú vplyv rovnaké kritériá ako pri spotrebe obvodu. Pri rovnakej alebo podobnej pravdepodobnosti viacerých ciest v obvode môže na celkovú priemernú rýchlosť vplývať správne usporiadanie vstupných premenných pre optimalizáciu dĺžky jednotlivých trás v obvode.
- Rýchlosť vykonávania. Vplyv na tento parameter má najmä počet stupňov logického obvodu, ale aj typ zvolenej technológie. V prípade synchrónnych sekvenčných logických obvodov aj frekvencia prepínania hodinovej synchronizačnej premennej.

Jedným z najdôležitejších parametrov je veľkosť obvodu. Jeho optimalizáciou, teda redukovaním obvodu, je možné dosiahnuť zlepšenie (v rôznych mierach intenzity) takmer vo všetkých ostatných spomenutých parametroch. Jednotlivé prvky obvodu reprezentujú základné logické funkcie a ich pospájaním je možné dosiahnuť výslednú Booleovu funkciu reprezentujúcu celý obvod. Vhodnou abstrakciou, napríklad výberom dátovej štruktúry, ktorá poskytuje dostatočné množstvo prostriedkov na optimalizáciu, je možné s logickými členmi obvodu manipulovať už pri fáze návrhu. Optimalizácia tejto štruktúry sa potom priamo prejaví na optimalizovaní samotného obvodu.

2.1 Booleova funkcia

Vstupom pre rozhodovacie diagramy je Booleova funkcia (B-funkcia) $f: B_n \rightarrow B$ nad množinou premenných X_n označených ako $f\{x_1, \dots, x_n\} = y$. Premenné sú usporiadané podľa významu zľava doprava, označené poradovým indexom, kde poradie vstupných premenných prislúcha klesajúcim váham priradeným jednotlivým premenným zľava doprava, začínajúc váhou 2^{n-1} pre premennú x_0 až po váhu 2^0 pre x_{n-1}. Akákoľvek B-funkcia špecifikovaná binárnym vektorom y a poradím premenných môže byť jednoducho zobrazená ako DD [3]. Pre jednoduchšiu

transformáciu na reziduálnu funkciu môže byť B-funkcia reprezentovaná pomocou modifikovanej pravdivostnej tabuľky zobrazenej v tabuľke 2.1.

Tabuľka 2.1 - Modifikovaná pravdivostná tabuľka pre B-funkciu

x_1	x_2	...	x_{n-2}	x_{n-1}	y	
					x_0	$\overline{x_0}$
0	0	...	0	0	$f(2^{n-2})$	$f(0)$
0	0	...	0	1	$f(2^{n-2}+1)$	$f(1)$
0	0	...	1	0	$f(2^{n-2}+2)$	$f(2)$
⋮	⋮	...	⋮	⋮	⋮	⋮
1	1	...	1	0	$f(2^{n-1}-2)$	$f(2^{n-2}-2)$
1	1	...	1	1	$f(2^{n-1}-1)$	$f(2^{n-2}-1)$

Modifikáca zvolenej funkcie na funkciu reziduálnych premenných sa môže dosiahnuť reprezentáciou vektora hodnôt funkcie ako kanonickej matice (2.1). Takáto matica [7] obsahuje 2^{n-1} stĺpcov po 2 riadkoch. Každý stĺpec predstavuje dvojicu hodnôt x_0 (spodný riadok) a $\overline{x_0}$ (horný riadok) danej funkcie.

$$\begin{matrix}\big(B(x_0, x_1, \dots, x_{n+1})\big) = \\ \begin{Vmatrix} f(0) & f(1) & \cdots & f(2^{n-2}-2) & f(2^{n-2}-1) \\ f(2^{n-2}) & f(2^{n-2}+1) & \cdots & f(2^{n-1}-2) & f(2^{n-1}-1) \end{Vmatrix}\end{matrix} \tag{2.1}$$

Definíciu B-funkcie je možné zapísať aj pomocou niekoľkých na seba nadväzujúcich výrokov, kde pre každú B-funkciu platí nasledovné:

Definícia 2.1: Booleova funkcia nad množinou *n* premenných $X_n = \{x_0, \dots, x_{n-1}\}$ je zobrazenie $f(X): B^n \to B, B = \{0,1\}$, zapísané tiež ako $f(x_0, \dots, x_{n-1}) = \{0,1\}^n$.

Definícia 2.2: Kofaktorom Booleovej funkcie je $f(x_0, \dots, x_{n-1})$ s ohľadom na premennú x_i je funkcia získaná dosadením konkrétnej hodnoty a z množiny $\{0,1\}$ za premennú, zapísané ako $f_{(x_i=a)} = f(x_0, \dots, a, \dots, x_{n-1})$.

Definícia 2.3: Každá Booleova funkcia je jednoznačne definovaná svojim vektorom hodnôt dĺžky 2^n z množiny $\{0,1\}$, v predchádzajúcich definíciach zapísané ako $\{0,1\}^n$.

Definícia 2.4: Hodnota Booleovej funkcie v bode predstavuje konkrétnu hodnotu plne definovaného kofaktora funkcie. Napríklad pre $f_{n=3} = 00001111$ je prvou hodnotou funkcie $f(000) = 0$ a poslednou hodnotou $f(111) = 1$.

Definícia 2.5: Hammingovou váhou funkcie, známou aj ako Hammingova vzdialenosť, sa rozumie počet hodnôt funkcie rovných 1 [34]. Napríklad pre $f = 00001111$ je Hammingova váha rovná $H(f) = 4$.

2.1.1 Ekvivalencia Booleových funkcií

Za ekvivalentné sa považujú také B-funkcie, ktore patria do rovnakej triedy ekvivalencie, v informatických vedách známe ako NPN triedy (Negation–Permutation–Negation - NPN). NPN klasifikácia priradí dve B-funkcie do rovnakej triedy pokiaľ je možné dosiahnuť jednu funkciu negovaním (prevodom na doplnkový tvar), permutáciou vstupných premenných a opätovným negovaním výstupu. Počet vzniknutých NPN tried je spravidla výrazne menší než počet B-funkcií [46] [47]. V praktických aplikáciách sa pre B-funkcie zisťuje príslušnosť k určitej triede, pre ktorú už sú známe alebo predrátané vlastnosti, napríklad pri syntéze alebo mapovaní FPGA obvodov [46]. Zjednodušene povedané, za *ekvivalentné* sa považujú také B-funkcie, ktoré preusporiadaním premenných jednej z nich poskytujú rovnaký výstupný vektor hodnôt. Za *rovnaké*, alebo *identické*, sa považujú funkcie, ktoré poskytujú rovnaký vektor hodnôt bez nutnosti preusporiadania.

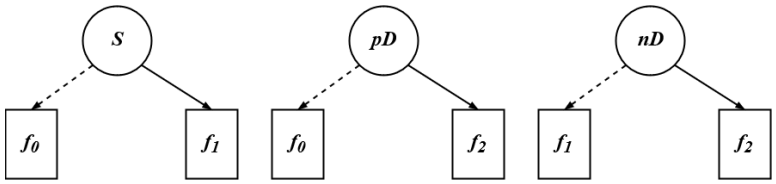
NPN klasifikovanie predstavuje samostatnú oblasť vedeckého výskumu. Existuje viacero metód zaoberajúcich sa návrhom metód s cieľom čo najrýchlejšie určiť príslušnosť k istej triede. Medzi práce so zaujímavými výsledkami patrí [47], kde autori predstavujú algoritmus pozostávajúci zo 4 krokov s postupným ohodnocovaním vstupov, zisťovaním polarity a priraďovaním do skupín symetrie. Najväčší uvažovaný vstup pre autorov je 16 premenných. Práca [46] vychádzajúca z rovnakého základu prezentuje hierarchické rozdelenie NPN tried pre ešte rýchlejšie zaradenie B-funkcie. Príbuznou oblasťou je tiež zisťovanie splniteľnosti B-funkcie, na ktoré sa používajú tzv. SAT riešiče (z ang. SATisfiability - SAT). Prehľad existujúcich riešení je uvedený v [48].

2.1.2 Dekompozícia Booleovej funkcie

Dekompozícia, alebo aj rozklad, B-funkcie berie za vstup binárny vektor y dĺžky 2^n kde n predstavuje počet vstupných premenných a na výstupe poskytuje 2 vektory dĺžky $2^n/2$. Ako už bolo spomenuté v úvodnej kapitole, existujú len 2 typy dekompozície, ktoré maju vplyv na výslednú veľkosť diagramu pri redukcii – Shannonova a Davio, ktorá sa ďalej delí na 2 typy – pozitívne Davio a negatívne Davio. Shannonova dekompozícia pre premennú x_i v princípe rozdeľuje vektor na 2 polovice na základe hodnoty danej premennej – ľavú polovicu $f_{x_i}(x)$ pre logickú hodnotu 1 (kladná) a pravú polovicu $f_{\bar{x_i}}(x)$ pre logickú hodnotu 0 (záporná). Podobne Davio dekompozícia berie rovnaký vstup a poskytuje 2 výstupy rovnakej dĺžky ako Shannon. Pre pozitívne Davio je ľavým výstupom záporná polovica vstupného vektora a pravým výstupom logický XOR oboch polovíc, pričom pre negatívne Davio je ľavým výstupom kladná polovica a pravým rovnako XOR oboch polovíc. Dekompozície B-funckie sú znázornené v (2.2), (2.3) a v obrázku 2.1.

$$f_0 = f_{\bar{x}_i}(x); \qquad f_1 = f_{x_i}(x); \qquad f_2 = (f_0 \oplus f_1) \tag{2.2}$$

$$S = f_0 . f_1; \qquad pD = f_0 \oplus x_i f_2; \qquad nD = f_1 \oplus \bar{x}_i f_2 \tag{2.3}$$

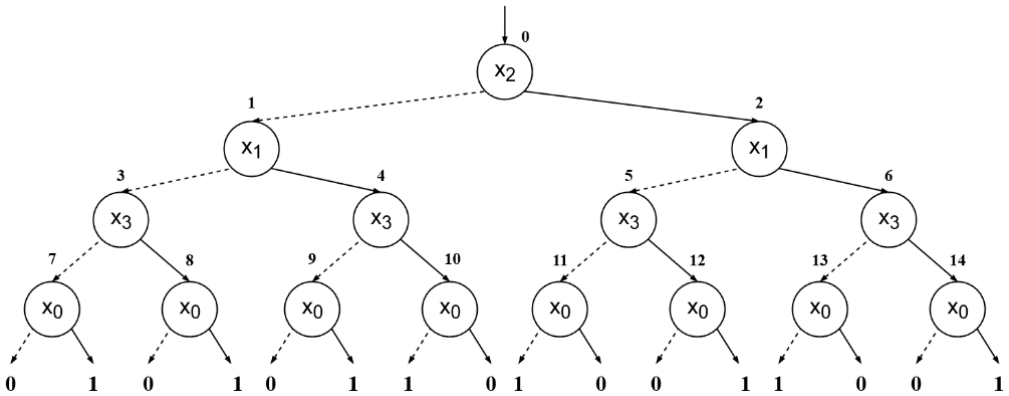


Obrázok 2.1- Tri typy dekompozície – Shannon (S), pozitívne Davio (pD) a negatívne Davio (nD) zo vzťahu (2.3)

2.2 Rozhodovacie diagramy

Rozhodovacie diagramy predstavujú orientovaný acyklický graf skladajúci sa z jedného koreňového uzla, viacerých prechodných uzlov a najviac dvoch terminálnych uzlov reprezentujúcich kladnú (*True*) a zápornú (*False*) logickú hodnotu, najčastejšie zobrazenú ako 1 a 0 v uvedenom poradí. Každý neterminálny uzol je označený Booleovou premennou x_i podľa poradia jej výskytu vo vstupnom poradí. Neterminálne uzly v neoptimalizovanom DD majú jednu vstupnú hranu (s výnimkou koreňa diagramu) a dve výstupné hrany označené ako záporná alebo nulová hrana (ľavá) a kladná alebo jednotková hrana (pravá) znázorňujúc hodnotu funkcie f_i podľa dekompozície rodičovského uzla. Hrana vychádzajúca z uzla i vchádza do uzla j, pričom musí platiť $j > i$ a zároveň $úroveň(j) > úroveň(i)$. Každý nekoreňový neterminálny uzol je koreňom pre samostatný diagram, pre potreby tejto práce nazvaný subdiagram.

Príklad 2.1: Vstupom pre syntézu DD je B-funkcia f_1 so 4 premennými v zadanom poradí $f_1(x_2, x_1, x_3, x_0) = 010101101001100$. Pre jednoduchosť sa predpokladá použitie iba Shannonovej dekompozície. Neredukovaný BDD pre f_1 je zobrazený na obrázku 2.2 . Je nutné poznamenať, že terminálne uzly 0 a 1 v tomto prípade predstavujú výstup funkcie.



Obrázok 2.2 - BDD pre f_1 z Príkladu 2.1

Základné typy DD (napr. BDD, FDD, KFDD) dodržiavajú určité implicitne dané pravidlá:

1. Rozhodovací diagram je v každej forme (redukovaný / neredukovaný) úplný (nesie plnú informáciu vstupnej B-funkcie).

2. Binárne diagramy majú na každej ceste od koreňa k terminálnemu uzlu práve jednu výstupnú hodnotu z dvoch možných logických hodnôt 0 a 1.

3. Každý uzol v diagrame reprezentuje práve jednu premennú.

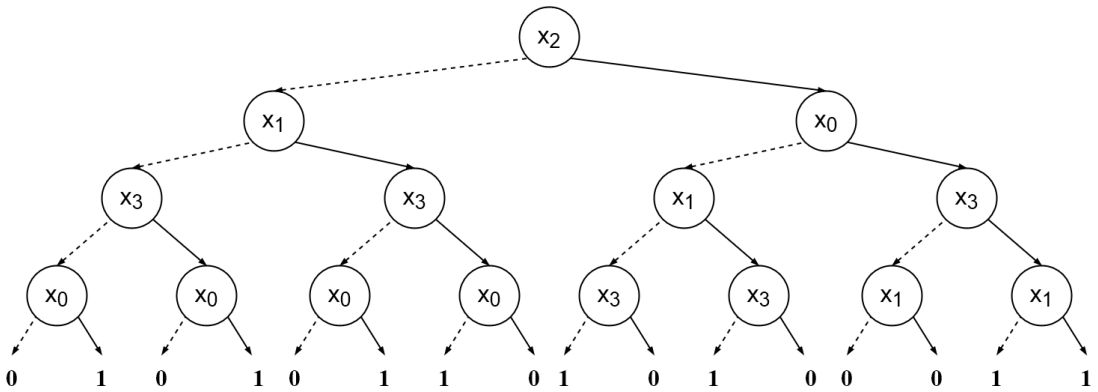
4. Všetky cesty od koreňa k terminálnemu uzlu produkujú rovnaký vektor premenných.

5. Každý uzol v diagrame reprezentuje práve jednu dekompozíciu premennej.

6. Všetky cesty od koreňa k terminálnemu uzlu produkujú rovnaký vektor dekompozícií.

2.2.1 Voľné rozhodovacie diagramy

Uvoľnením pravidiel spomenutých v kapitole 2.2 vznikajú nové typy diagramov. Povolením viacerých vektorov premenných (pravidlo 4.) vznikajú štruktúry nazývané voľné rozhodovacie diagramy (FrDD). Tieto diagramy umožňujú v každom subdiagrame rôzne poradie premenných so zachovaním ostatných pravidiel. Prínosom takýchto diagramov je rozšírenie poľa možných redukcií v diagrame avšak za cenu vyššej výpočtovej zložitosti. Pre každý uzol v diagrame je možné zvoliť rozdielny vektor premenných pre jeho ľavého a pravého potomka použitím rovnakého algoritmu ako pri určovaní vstupného poradia. Ku každému subdiagramu je možné pristupovať ako k samostatnému diagramu a všetky optimalizačné metódy môžu byť použité nanovo so zložitosťou upravenou na $n-1$ vstupných premenných. Príklad voľného rozhodovacieho diagramu pre funkciu f_1 z príkladu 2.1 je znázornený na obrázku 2.3, logika FrDD bola aplikovaná na dvoch po sebe idúcich jednotkových potomkov. Za povšimnutie stojí preusporiadanie terminálnych (výstupných) uzlov práveho subdiagramu.



Obrázok 2.3 - FrDD pre f_1 z Príkladu 2.1

2.2.2 Optimalizácia rozhodovacích diagramov

Pri optimalizácii DD je nutné uvažovať okrem výsledného DD aj o samotnom vstupe, napríklad poradí premmenných vstupnej funkcie. V zásade môžu byť optimalizačné metódy rozhodovacích diagramov rozdelené do dvoch kategórií [8]:

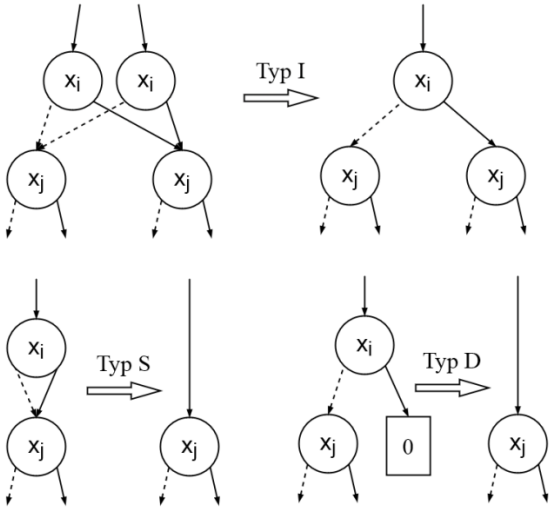
1. Usporiadanie – aplikovaním metódy usporiadania je možné dosiahnuť Usporiadaný DD (Ordered DD - ODD), ktorý dodržiava určité poradie premenných. Usporiadanie premenných má zásadný vplyv na mieru dosiahnutej redukcie DD.

2. Redukcia - aplikovaním metódy redukcie na ODD je možné dosiahnuť Redukovaný ODD (Reduced ODD - RODD), ktorý má nižší počet uzlov ako neredukovaný diagram. Kompletne redukovaný DD predstavuje kanonickú formu DD. RODD dodržiava v základnej forme 3 pravidlá, znázornené na obrázku 2.4:

    a. Jedinečnosť uzla (Typ I) – každé dva neidentické uzly *u* a *v* predstavujúce rovnakú premennú s rovnakou dekompozíciou majú rozdielne nasledujúce uzly (potomkov). Inými slovami, ak *prem(u) = prem(v)*, *dek(u) = dek(v), ľavý(u) = ľavý(v)*, a *pravý(u) = pravý(v)* tak potom *u = v*.

    b. Neredukovateľnosť pre Shannonov uzol (Typ S) – každý uzol so Shannonovou dekompozíciou má rôznych potomkov. Inými slovami, ak *dek(u) =S, ľavý(u) != pravý(u).*

    c. Neredukovateľnosť pre Davio uzol (Typ D) – každý uzol s Davio dekompozíciou má pravého potomka iného ako terminálny uzol *0*. Inými slovami, ak *dek(u) =pD* alebo *dek(u) =nD,* tak potom *pravý(u) != 0.*



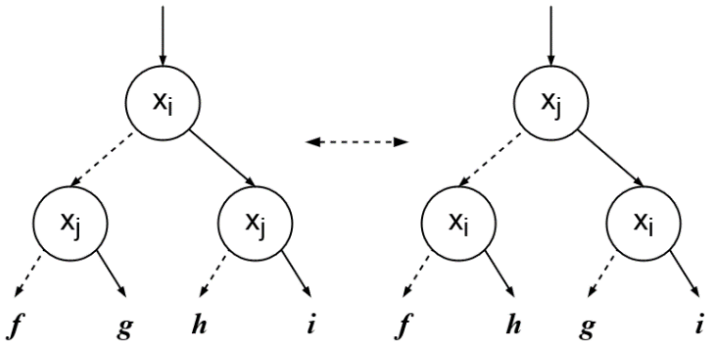
Obrázok 2.4 - Redukčné pravidlá pre redukovaný a usporiadaný DD

2.2.3 Usporiadanie vstupných premenných

Hľadanie optimálneho poradia premenných má exponenciálnu zložitosť. Pre funkcie s väčším počtom vstupných premenných je prehľadávanie celej množiny poradí nemožné v akceptovateľnom čase. Výpočtová zložitosť záleží od vstupných parametrov. Pri usporiadaní je zložitosť pre celú množinu poradí n premenných rovná $n!$ pre BDD, FDD alebo KFDD. Pre FrDD stúpa táto zložitosť na $n!\,(n-1)!\,\ldots 3!\,2!\,1!$, keďže môže každý subdiagram disponovať vlastným poradím. Uvedené zložitosti však rátajú iba s jedným typom dekompozície. Pri použití 3 typov dekompozície sa táto zložitosť násobí hodnotou 3^n pre základné aj voľné diagramy.

Existuje viacero metód snažiacich sa zmierniť tento problém, napríklad zmenšovaním prehľadávanej množiny s rôznou úrovňou vplyvu na výslednú redukciu diagramu. Tieto metódy môžu byť rozdelené do kategórií na základe zložitosti vnútorného algoritmu.

Základné metódy usporiadania predstavujú jednoduchú modifikáciu vstupného poradia. Jednou z takých metód je napríklad výmena susedných premenných na rovnakej úrovni. Cieľom výmeny premenných na úrovni k a $k+1$ pre DD G funkcie f je transformácia G na G' funkcie f. Rozdiel v poradí X a X' je iba na úrovniach k a $k+1$ a môže byť vyjadrený ako $X(k) = X'(k+1)$ a $X(k+1) = X'(k)$. Výmena týchto premenných nemá vplyv na zvyšok diagramu a je znázornená na obrázku 2.5 [10].



Obrázok 2.5 – Výmena susedných premenných v DD

V heuristický metódach je poradie premenných určené vzhľadom na dostupné informácie o cieľenej optimalizácii pred samotnou syntézou (konštrukciou) DD. *Force* algoritmus predstavený v [10] patrí medzi najznámejšie príklady v tejto kategórii. Základnou myšlienkou tohto algoritmu je výpočet vplyvov na každú premennú a následné rozloženie premenných v smere týchto vplyvov. V algoritme *Force* sa na CNF rovnicu pozerá ako na hypergraf, kde premenné rovnice prislúchajú k uzlom a operátory rovnice k hyperhranám grafu. Samotný algoritmus určí 2 hodnoty počas vykonávania a iteratívne ich používa na preusporiadanie premenných. Ďalšou heuristickou métodou hodnou spomenutia je metóda opísaná v [11] založená na dokázanom predpoklade kde počet uzlov na jednej úrovni DD záleží len na usporiadaní premenných na nižších úrovniach. Algoritmus tejto metódy postupne umiestňuje

všetky premenné na prvú úroveň a určuje, ktorá z nich dosiahne po redukcii najmenší počet uzlov. Premenná alebo skupina premenných s najlepšími výsledkami sú uložené pre danú úroveň DD a zvyšné premenné sa testujú na vyšších úrovniach. Tento proces sa opakuje až kým sa nedosiahne výsledný DD. Zložitosť algoritmu je síce exponenciálna, ale dosahuje lepšie výsledky ako iteratívny prechod celou množinou poradí premenných.

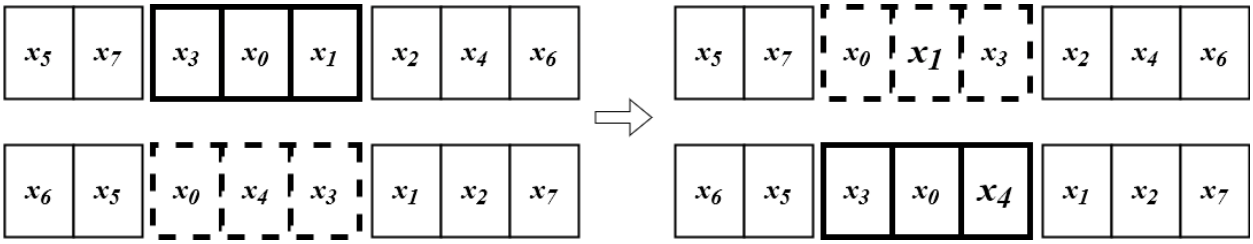
Treťou a poslednou skupinou sú alternatívne metódy založené na evolučných algoritmoch (EA). EA sa v súčasnosti zaraďujú k najvýznamnejším algoritmom v danej oblasti. Základným pojmom EA je populácia, ktorá reprezentuje sadu chromozómov. Chromozóm môže predstavovať poradie premenných alebo poradie dekompozícií pre jednotlivé premenné ako vektor génov, napríklad $\{x_0, x_5, x_6, x_3, x_2, x_1, x_4, x_7\}$. V takomto prípade je génom premenná v konkrétnom poradí (chromozóme) danej populácie poradí. Populácia jednej fázy EA sa nazýva generáciou. Vstupná populácia, alebo aj prvá generácia, je vytvorená náhodným generovaním populácie chromozómov, ktoré sú následne usporiadané na základe hodnoty vhodnosti. Vhodnosť sa určuje pre každý chromozóm novovytvorenej populácie. Z usporiadanej generácie sa vyberie zvolený počet najvhodnejších jedincov (chromozómov) pre vytvorenie nasledujúcej generácie. Algoritmus výpočtu vhodnosti záleží na riešenom probléme.

Najvhodnejšie chromozómy predchádzajúcej (rodičovskej) generácie prechádzajú 2 genetickými operáciami, každá s určitou pravdepodobnosťou pre zvýšenie rôznorodosti aktuálne vytváranej generácie a pre zníženie pravdepodobnosti uviaznutia v lokálnom optime. Niektoré formy EA tiež pridávajú techniku zvanú elitizmus, ktorá zachováva najvhodnejších jedincov naprieč všetkými generáciami pre zabezpečenie stáleho zlepšovania nových generácií. Chromozómy aktuálnej generácie sú usporiadané a vyberané podľa kritérií vhodnosti. V tomto kroku EA začne vypĺňať novú generáciu postupným genetickým mutovaním a krížením zoradených chromozómov. Mutáciou sa vo vektore poradí (chromozóme) rozumie náhodné prehodenie 2 alebo viacerých premenných (génov) medzi sebou. Operácia býva štandardne implementovaná ako prehodenie premennej na náhodnom poradí s premennou na jej doplnkovej pozícii, napr. mutácia chromozómu dĺžky 8 na pozícii 2 je zobrazená na obrázku 2.6.

Obrázok 2.6 – Mutácia chromozómu poradia premenných

Druhou genetickou operáciou je kríženie, pri ktorom sa 2 vybrané chromozómy rozdelia na rovnakom, náhodne zvolenom úseku a vymenia si segmenty medzi sebou. V niektorých variáciách sú náhodné zvolené 2 body na jednom chromozóme a segment medzi týmito 2 bodmi je nahradený segmentom z napárovaného chromozómu. Keďže jednoduchá výmena segmentov poradí premenných by mohla spôsobiť znásobený výskyt jednej alebo viacerých premenných, je potrebné túto operáciu ošetriť. Namiesto výmeny častí chromozómu sa budú premenné zo

segmentu jedného chromozómu postupne vkladať do druhého chromozómu za bodom kríženia v poradí, v akom sa vyskytujú v prvom chromozóme. Ak by mala byť pridaná premenná, ktorá sa už v danom chromozóme nachádza, je namiesto toho nahradená inou, chýbajúcou premennou. Kríženie je znázornené na obrázku 2.7. Poradie operácií mutácie a kríženia nie je striktne dané, rôzne algoritmy môžu požívať rôzne poradia a dokonca pridať aj iné operácie. Pre potreby tejto práce sa uvažuje iba o týchto dvoch operáciach v poradí kríženie - mutácia.



Obrázok 2.7 – Kríženie chromozómov

EA začína vytvorením iniciálnej populácie aplikáciou mutácie a prípadného kríženia na vstupný vektor poradia premenných. Veľkosť poradia ako aj pravdepodobnosť výskytu jednotlivých operácií je témou na samostatný výskum a môže byť rôzna v závislosti od oblasti použitia EA. Obe hodnoty majú vplyv na výpočtový čas a presnosť (mieru optimalizácie) dosiahnutých výsledkov.

Existuje viacero variácií EA, napríklad algoritmus rojom častíc (Particle Swarm Optimization - PSO) [12] predstavuje populačno-orientovanú stochastickú techniku inšpirovanú spoločenským správaním kŕdľa vtákov alebo húfom rýb. V PSO prelietávajú potenciálne riešenia, nazývané častice, oblasťou riešeného problému nasledovaním aktuálnych častíc optima. Častice sa učia z vlastných predchádzajúcich skúseností, zo skúseností okolitých častíc a nakoniec spolu konvergujú blízko riešenia, ktoré môže byť optimálne alebo približne optimálne ak bola splnená ukončovacia podmienka. Častice vycítia svoju blízkosť optimálnemu riešeniu pomocou funkcie vhodnosti. Inou formou EA so zaujímavými výsledkami je modifikovaný memetický algoritmus (Modified Memetic Algorithm - MMA) [13]. Kľúčovou vlastnosťou MMA je použitie viacerých techník na lokálne vyhľadávanie. Zatiaľ čo gén, ktorý prechádza z rodiča na potomka nemôže byť zmenený v klasickom EA (okrem mutácie), mémy MMA prenášajú informácie medzi sebou aby čo najviac zapadli do vyhodnocovacej funkcie (napríklad cez lokálne vyhľadávanie), ktoré je umožnené vďaka znalosti oblasti riešenia.

Všetky vyššie spomenuté algoritmy majú jednu spoločnú vlastnosť, ktorou je primárna orientácia na redukciu veľkosti. Počet uzlov v diagrame je priamo úmerný veľkosti reprezentovaného obvodu, sady, funkcie alebo iného cieleného objektu a má priamy vplyv na viacero parametrov naraz.

Treba podotknúť, že pri KFDD je EA použiteľný, v istej obmenenej podobe, aj na vektor dekompozícií. Pri diagramoch, ktoré používajú iba jeden typ dekompozície, napríklad BDD

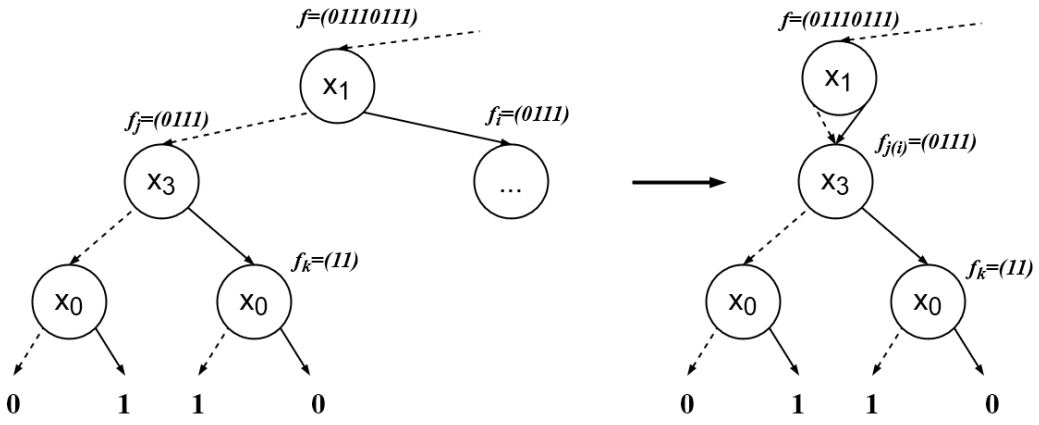
alebo FDD, je použitá dekompozícia na premennú rovnaká bez ohľadu na ich poradie. Pri KFDD je nutné ku každej premennej uviesť aj dekompozíciu a tak vzniká k vektoru poriadia premenných aj vektor poradia dekompozícii, ktorý môže taktiež podliehať operáciam EA. Pri vektore dekompozícií je možné zvoliť odlišnú funkciu vhodnosti a nie je potrebné ošetrovať operáciu kríženia, pretože vektor dekompozícií bude mať opakujúce sa prvky (iba 3 dekompozície S, pD a nD) v každom prípade pre $n > 3$.

2.2.4 Konštrukcia a redukcia rozhodovacích diagramov

Konštrukcia (syntéza) diagramu prebieha postupným uplaťnovaním rozkladu (dekompozície) na vstupnú funkciu a jej výstupy až po dosiahnutie konkrétnych logický hodnôt 0 alebo 1.

Pri redukcii diagramov sa v prvom kroku aplikujú optimalizačné metódy na vektor premenných, prípadne na vektor dekompozícií. Pod aplikáciou metód sa rozumie vygenerovanie celej populácie (množiny) poradí, pre ktorú budú diagramy skonštruované a následne redukované. Na generovanie množiny vektorov poradí a vektorov dekompozícií môže byť použitá ľubovoľná metóda, napr. statické metódy usporiadania alebo EA. Ak je nevyhnutné otestovať všetky možné poradia premenných a dekompozícii, je celková velkosť množiny kombinácií pre n premenných rovná $n!.\,3^n$ (pre základné typy DD).

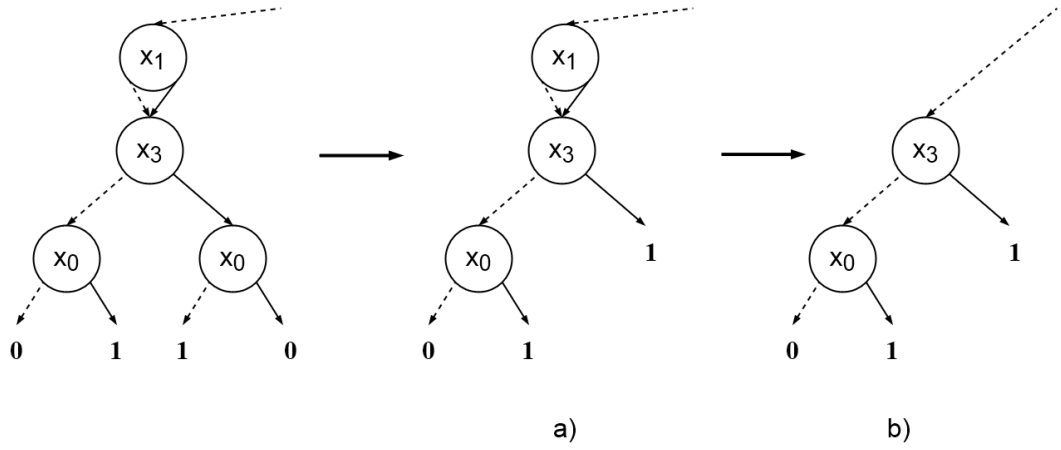
Samotná redukcia prebieha postupným uplatňovaním redukčných pravidiel zdola nahor. Je preto potrebné v prvom kroku skonštruovať kompletný diagram od koreňa po terminálne uzly. Počas tohto kroku môže byť uplatnené iba redukčné pravidlo typu I. Ak sa vytvára subdiagram pre funkciu f_i z rozkladu vstupnej funkcie f a na tej istej úrovni existuje subdiagram f_j s rovnakými parametrami (vektor premenných, vektor dekompozícií), je možné presmerovať výstupnú hranu rodičovského uzla na už existujúci diagram (obrázok 2.8).



Obrázok 2.8 – Redukcia typu I použitá pri syntéze diagramu

Po syntéze kompletného diagramu s redukciou typu I sa začnú zdola nahor aplikovať redukčné pravidlá S a D v závislosti od dekompozície použitej na daný uzol. Tento proces je znázornený na obrázku 2.9, kde sa predpokladá použitie Shannonovej dekompozície na premenné x_1, x_3

a pozitívnej Davio dekompozície na premennú x_0. Prvá je aplikovaná redukcia typu D na uzol najnižšej úrovne, teda na uzol premennej x_0 (obrázok 2.9-a). Na vyššej úrovni je následne aplikovaná redukcia typu S na uzol x_1 a potomok rodičovského uzla je presmerovaný na uzol premennej x_3 (obrázok 2.9-b).



Obrázok 2.9 – Redukcia typu D (a) a typu S (b) použitá pri redukovaní diagramu

2.2.4 Typy rozhodovacích diagramov

Typy DD spomínané v predchádzajúcich kapitolách patria k znakovým DD. Ich spoločnou vlastnosťou je dodržanie pravidla jednej premennej a jednej dekompozície (typu rozkladu) na uzol. Typ zvolenej dekompozície má vplyv na dosiahnutú mieru redukcie, napríklad FDD v porovnaní s BDD dosahuje lepšie výsledky pri funkciách používajúcich logické členy AND/XOR (najmä artitmeticko-logické, telekomunikačné obvody a obvody opravujúce chyby) [25]. BDD naproti tomu zvyšuje mieru redukcie pri obvodoch založených na multiplexoroch [26]. V niektorých prípadoch môže kombinácia oboch typov rozkladov použitých na vstupnú funkciu viesť až k lineárnej veľkosti výsledného KFDD [27]. Výstupom takýchto DD je zvyčajne jedna z dvoch binárnych logických hodnôt. Pri rozšírení možných výstupov o nedefinovanú hodnotu je možné dosiahnuť modifikovaný binárny rozhodovací diagram (Modified Binary Decision Diagram - MBDD), vhodný pre reprezentáciu neúplne definovaných B-funkcií [28].

V kapitole 2.2.2 sa uvázda redukčné pravidlo typu D (2.c) pre uzol s Davio dekompozíciou. Použitím tohto pravidla na uzly v BDD (iba Shannonova dekompozícia) sa získava nový typ BDD nazývaný binárny rozhodovací diagram s potlačenou nulou (Zero-Suppressed Binary Desicion Diagram - ZBDD). Použitie pravidla D má však za následok stratu možnosti použitia pravidla S kvôli nekonzistentnosti, ktorá môže vyplynúť po predchádzajúcom aplikovaní pravidla D (jeden uzol môže mať rovnakého ľavého aj pravého potomka). Cesty od koreňa k listu (terminálnemu uzlu) v ZBDD tvoria termy predpisu B-funkcie kde neprítomnosť premennej na danej ceste znamená jej zápis v negovanom tvare v terme funkcie. ZBDD sú vhodné na reprezentáciu množín [29] [54].

Rozšírením výstupnej množiny hodnôt je možné získať rozhodovací diagram s viacerými konečnými hodnotami (Multi-Terminal Decision Diagram - MTDD) alebo algebraický rozhodovací diagram (Algebraic Decision Diagram - ADD). Premenné v týchto diagramoch sú binárneho charakteru a je dovolené použitie len Shannonvho rozkladu. Pri použití pozitívneho Davio rozkladu a ohodnotení hrán celočíselnými hodnotami je možné dosiahnuť momentový rozhodovací diagram (positive Binary Moment Diagram - pBMD) [30]. Umožnením viacnásobného ohodnotenia hrán sa získava násobný BMD (*BMD), prípadne Kroneckerov násobný momentový diagram (K*BMD). Hodné zmienky sú napríklad aj ternárne rozhodovacie diagramy (Ternary DD - TDD) kde má každý neterminálny uzol práve 3 potomkov [31] alebo čiastočné usporiadané DD (Ordered Partial DD - OPDD).

2.3 Reziduálna premenná

Ak má byť implementovaná vstupná B-funkcia s $n+1$ premennými na úroveň zložitoti DD s n premennými, je potrebné aby bola aspoň jedna premenná dostupná v priamom aj doplnkovom tvare. Vektor premenných musí byť preusporiadaný tak, aby táto premenná mala najvyššiu váhu.

Definícia 2.6 - Ak sú definované výskyty vo funkcii s $n+1$ premennými (napríklad podľa ich poradia v pravdivostnej tabuľke), je možné im priradiť váhy. Premenná s najvyššou váhou môže byť z vektora premenných odstránená a nahradená logickou hodnotou v priamom a doplnkovom tvare. Takáto premenná sa nazýva reziduálnou premennou (Residual Variable - RV). Ľubovoľná premenná môže byť RV pokiaľ je dostupná v priamom aj doplnkovom tvare [32].

Definícia 2.7 – Ak je určená RV vo funkcii s $n+1$ premennými, obvodová reprezentácia tejto funkcie môže byť transformovaná na obvodovú reprezentáciu funkcie s n premennými kde RV je pripojená na dátový vstup obvodu [32].

Definícia 2.8 – DD pre funkciu s $n+1$ premennými sa nazýva rozhodovací diagram s reziduálnou premennou (Residual Variable in DD - RViDD) pokiaľ je jedna premenná reziduálnou premennou a je zároveň terminálnym uzlom v RViDD. Podľa typu alebo typov použitej dekompozície potom vzniká RViBDD, RViFDD alebo RViKFDD [32].

Pre vytvorenie DD s n premennými sa opakuje procedúra vytvárania DD postupnou dekompozíciou až kým sa nedosiahne úroveň definovaná vzťahom 2.4.

$$f_{x_n=c}(x_1, \dots, x_n) \coloneqq f((vector\ n-1), c) \tag{2.4}$$

Vo vzťahu 2.4 platí $c \in \{0,1\}$ a $(vector\ n-1)$ obsahuje prislúchajúce nahradenie hodnôt 0 a 1 podľa špecifikovaného poradia premenných $(x_1, \dots, x_{n-1})$ vo vyšších vrstvách DD. Týmto sa dosiahne DD, ktorý určuje hodnotu funkcie pre všetky kombinácie vstupov pre premenné $x_1, \dots, x_n$ [32]. V prípade, kedy je dekompozícia vstupnej funkcie ukončená o jednu iteráciu skôr, je vzťah 2.4 upravený na vzťah 2.5.

$$f_{x_{n-1}=c}(x_1, \dots, x_{n-1}, x_n) \coloneqq f((vector\ n-2), c, x_n) \tag{2.5}$$

Vo vzťahu 2.5 platí $c \in \{0, 1\}$ a $(vector\ n-1)$ obsahuje prislúchajúce nahradenie hodnôt 0 a 1 podľa špecifikovaného poradia premenných $(x_1, \dots, x_{n-1})$ vo vyšších vrstvách DD. Týmto postupom je možné dosiahnuť 4 konečné stavy alebo skôr substitučné pravidlá, pri ktorých je výsledná hodnota závislá na c a x_n. Substitučné pravidlá majú 2 vstupy a poskytujú jednu výstupnú hodnotu, ktorá reprezentuje RV, znázornené v tabuľke 2.2.

Tabuľka 2.2 – Substitučné pravidlá pre reziduálne funkcie

Pravidlo	v_1	v_2	u_i
0	0	0	0
1	0	1	x_i
2	1	0	$\overline{x_i}$
3	1	1	1

Použitím vzťahov 2.2, 2.3 a pravidiel v tabuľke 2.2 je možné získať konečné stavy RV pre všetky typy dekompozícií tak, ako je to znázornené v tabuľke 2.3.

Tabuľka 2.3 – Konečné stavy reziduálne premennej pre všetky typy dekompozícií

f_0	f_1	f_2	Hodnota funkcie			Konečný stav		
			S	pD	nD	S	pD	nD
0	0	0	0.0	$0 \oplus x_i.0$	$0 \oplus \overline{x_i}.0$	0	0	0
0	1	1	0.1	$0 \oplus x_i.1$	$1 \oplus \overline{x_i}.1$	x_i	x_i	x_i
1	0	1	1.0	$1 \oplus x_i.1$	$0 \oplus \overline{x_i}.1$	$\overline{x_i}$	$\overline{x_i}$	$\overline{x_i}$
1	1	0	1.1	$1 \oplus x_i.0$	$1 \oplus \overline{x_i}.0$	1	1	1

Príklad 2.2: Je daná vstupná B-funkcia f_2 so 4 premennými v tvare $f_2(x_2, x_0, x_1, x_3) = 100001001011011$ kde premenná x_2 je zvolená za RV. Pre jednoduchosť je použitá iba Shannonova dekompozícia. Pravdivostná tabuľka a jej modifikovaná verzia sú zobrazené v tabuľke 2.4. Stĺpec y môže byť vyjadrený kanonickou maticou B_2 výsledných hodnôt vo vzťahu 2.6.

$$\left(B_2(x_2, x_0, x_1, x_3)\right) = \left\|\begin{matrix}1 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1\end{matrix}\right\| \tag{2.6}$$

Tabuľka 2.4 – Modifikovaná pravdivostná tabuľke pre f_2 z príkladu 2.2

x_2	x_0	x_1	x_3	y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

➡

x_0	x_1	x_3	y	
			x_2	$\overline{x_2}$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

Použitím substitučných pravidiel v tabuľke 2.2 môže byť kanonická matica B_2 prepísaná ako vektor reziduálnych funkcií Z_2 vo vzťahu 2.7. Za povšimnutie stojí fakt, že dĺžka vektora Z_2 je polovičná oproti dĺžke vstupnej funkcie f_2. Rovnaký postup sa uplatňuje pri ostatných typoch dekompozície podľa substitučných pravidiel uvedených tabuľke 2.3.

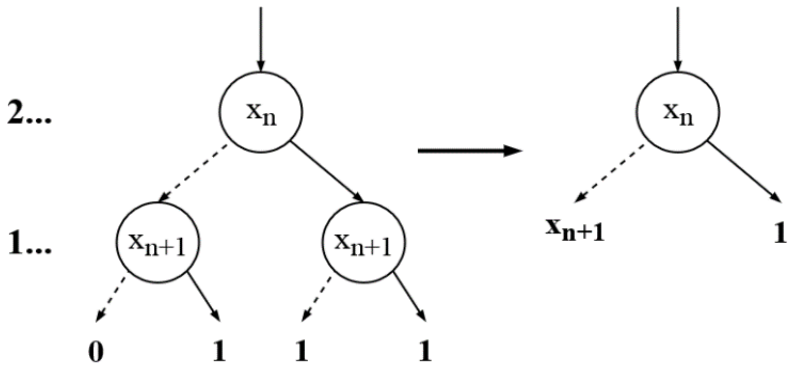
$$\left(Z_2(x_2, x_0, x_1, x_3)\right) = (\overline{x_2}, x_2, 0, x_2, x_2, \overline{x_2}, x_2, x_2) \tag{2.7}$$

2.3.1 Náhrada reziduálnej premennej

Keďže RViDD predstavuje nový typ DD, je dôležité zachovať jeho predispozíciu pre základné redukčné pravidlá používané na BDD alebo FDD pre ich ďalšie využitie v optimalizácii. Vďaka tomu je možné ľubovoľný BDD alebo FDD transformovať na RViDD so zachovaním vlastností pôvodného typu DD a priamo porovnať prínos RV z pohľadu niekoľkých parametrov, hlavne veľkosti diagramu (počet uzlov). Kapitola je spravovaná na základe [32].

Transformácia DD uzla na prvej úrovni na RViDD uzol je vykonaná podľa pravidiel v tabuľke 2.3 kde sú reziduálne funkcie nahradené zodpovedajúcim konečným stavom RV (obrázok 2.10). Keďže hodnoty konečných stavov sú rovné, transformácia je použiteľná na všetky typy dekompozície. Vyššie úrovne RViDD ostávajú nezmenené oproti svojim DD náprotivkom.

Dôležitou vlastnosťou pri redukcii je schopnosť výmeny susedných premenných. Aj keď samotná výmena nie je významne efektívna, slúži ako základ pre viacero algoritmov. Ak sú vymenené susedné premenné na úrovniach i a j v DD, poradie premenných je modifikované



Obrázok 2.10 – Transformácia DD uzla na RViDD uzol

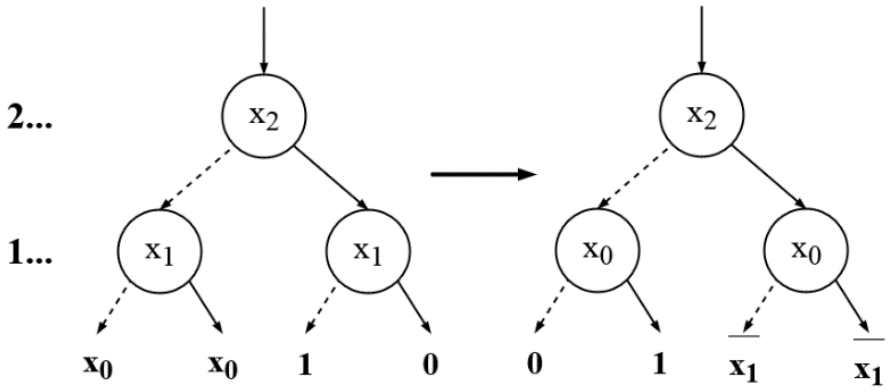
z $f(x_0, \dots, x_i, x_j, \dots, x_{n-1})$ na $f(x_0, \dots, x_j, x_i, \dots, x_{n-1})$ bez prejavenia zmien vo vyšších alebo nižších úrovniach.

Proces výmeny RV s inou premennou je mierne zložitejší. Zatiaľ čo terminálne uzly v DD môžu nadobúdať iba dve hodnoty z množiny {0,1}, ktoré vedú k 4 možným stavom {00,01,10,11}, RViDD nadobúda hodnoty $\{0,1,x_i,\overline{x_i}\}$, ktoré vedú k 16 možným stavom. Výmena reziduálnej premennej s inou teda musí dodržiavať pravidlá zobrazené v tabuľke 2.2 a konečné stavy po výmene sa dosiahnu jednoduchým rozložením RV na B-funkciu, výmenou premenných ako v DD a následným skonštruovaním RViDD pomocou nahradenia funckií poslednej úrovne hodnotami reziduálnych funkcií. Pravidlá pre výmenu RV sú zobrazené v tabuľke 2.5, kde sa nachádzajú len zmenené výstupné hodnoty oproti vstupom. Pravidlá s nezmenenými hodnotami ako aj s hodnotami, kde je RV jednoducho nahradená novou premennou pri zachovaní pozície a negácie, sú vynechané.

Tabuľka 2.5 – Pravidlá pre výmenu reziduálne premennej

Pred výmenou	Po výmene	Pred výmenou	Po výmene
Binárny vektor (v_1, v_2, v_3, v_4)	Binárny vektor (v_1, v_3, v_2, v_4)	Hodnoty (u_1, u_2)	Hodnoty (u_1', u_2')
00,10	01,00	$0, \overline{x_i}$	$x_j, 0$
00,11	01,01	$0, 1$	x_j, x_j
01,00	00,10	$x_i, 0$	$0, \overline{x_j}$
01,01	00,11	x_i, x_i	$0, 1$
10,10	11,00	$\overline{x_i}, \overline{x_i}$	1,0
10,11	11,01	$\overline{x_i}, 1$	$1, x_j$
11,00	10,10	1,0	$\overline{x_j}, \overline{x_j}$
11,01	10,11	$1, x_i$	$\overline{x_j}, 1$

Priama výmena RV umožňuje modifikáciu existujúceho RViDD bez potreby nového konštruovania diagramu s novou RV. Keďže jej výmena prechádza viacerými krokmi oproti výmene uzla na iných úrovniach, je vhodné použiť základné metódy usporiadania na určenie najvhodnejšej premennej na miesto RV. Napriek zvýšenému počtu krokov pri výmene susedných premenných sa zdá, že je to jedinou nevýhodou RV. Príklad výmeny RV je zobrazený na obrázku 2.11.



Obrázok 2.11 – Výmena susednej premennej s reziduálnou premennou

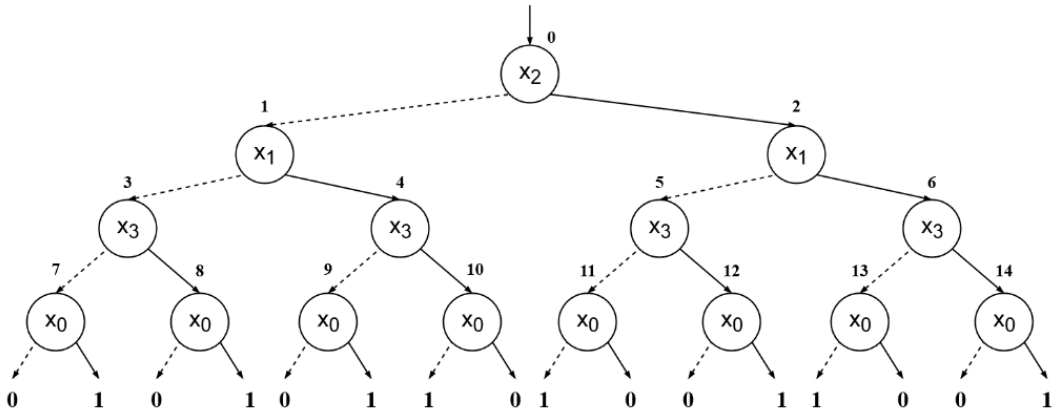
2.3.2 Konštrukcia a redukcia rozhodovaciaho diagramu s reziduálnou premennou

Podstatnou výhodou RV je rovnosť konečných stavov pre všetky dekompozície z tabuľky 2.3. Týmto sú automaticky zachované možnosti aplikovania redukčných pravidiel na uzly v RViDD. Redukčné pravidlo I môže byť použité na ľubovoľné dva uzly s oboma rovnakými potomkami. Redukčné pravidlo S môže byť použité na ľubovoľný Shannonov uzol s dvoma identickými potomkami aj v prípade, ak sú obaja potomkovia RV, napríklad pravý potomok uzla x_2 na obrázku 2.11. Redukčné pravidlo D môže byť použité na ľubovoľný Davio uzol, ktorého pravý potomok je rovný terminálnemu uzlu 0.

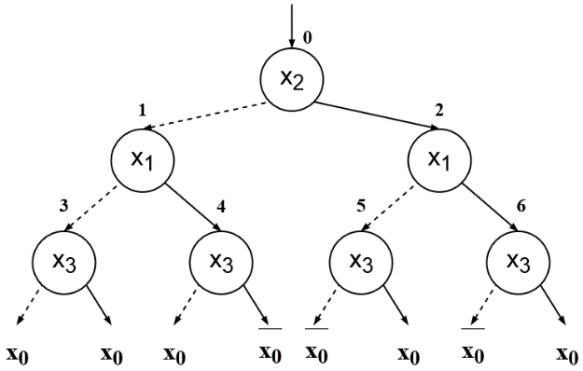
Príklad 2.3: Vstupom pre syntézu DD je B-funkcia f_3 so 4 premennými $f_3(x_2, x_1, x_3, x_0) = (0101011010011001)$. Pre jednoduchosť sa predpokladá použitie iba Shannonovej dekompozície. Neredukovaný BDD pre f_3 je zobrazený na obrázku 2.12. Premenná x_0 je zvolená za RV. Dodržiavaním rovnakých krokov ako v príklade 2.2 je modifikovaná pravdivostná tabuľka s vektorom reziduálnych premenných Z_3 znázornená vo vzťahu 2.8.

$$\left(Z_3(x_2, x_1, x_3, x_0)\right) = (x_0, x_0, x_0, \overline{x_0}, \overline{x_0}, x_0, \overline{x_0}, x_0) \tag{2.8}$$

Pri syntéze diagramu zhora nadol je možné použiť redukčné pravidlo I na uzly, ktoré zdieľajú určité charakteristiky – rovnaká úroveň, premenná, dekompozícia a potomkovia. Týmto sa dosiahne zníženie času potrebného na syntézu celého diagramu.

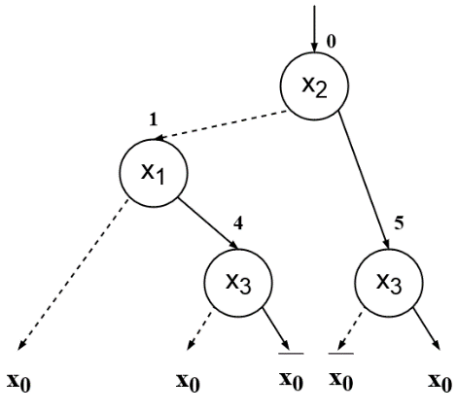


Obrázok 2.12 – BDD pre f_3 z Príkladu 2.3



Obrázok 2.13 – RViBDD pre f_3 z Príkladu 2.3

Neredukovaný RViBDD je zobrazený na obrázku 2.13. Na obrázkoch 2.12 a 2.13 je znázornený rozdiel vo veľkosti oboch diagramov, RViBDD má polovičný počet uzlov oproti BDD vďaka vynechaniu najpočetnejšej vrstvy diagramu a jej nahradením reziduálnou premennou.



Obrázok 2.14 – Redukovaný RViBDD pre f_3 z Príkladu 2.3

Syntéza a redukovanie RViDD prebieha rovnakým postupom ako syntéza DD s ľubovoľnou dekompozíciou. Vďaka vlastnostiam RV ostávajú redukčné pravidlá nezmenené a môžu byť

rovnako aplikované na DD aj RViDD, čo robí z RV významného prispievateľa pri redukcii rozhodovacích diagramov na binárnej báze. RViBDD má nižší počet uzlov oproti RViBDD, redukčný čas by mal byť teda (aspoň v teoretickej rovine) polovičný.

Pri syntéze voľného DD (FrDD) s reziduálnou premennou platia rovnaké pravidlá ako pri DD s jednou výnimkou. Syntéza bežného DD používa rovnaké poradie premenných v každej ceste od koreňa k terminálnym uzlom a RV je preto vždy rovnaká v každom subdiagrame. FrDD naproti tomu umožňuje použiť ľubovoľné, od seba odlišné, poradia v ľavom a pravom subdiagrame, čo v niektorých prípadoch vedie k rôznym premenným na najnižšej úrovni diagramu. Pre zníženie výpočtovej náročnosti je preto vhodné zvoliť jednu RV, ktorá ostane nemenná pre všetky subdiagramy v celom DD. Preusporiadanie premenných bude pri každom kroku syntézy rešpektovať zvolenú RV a jej pozíciu na konci (prípadne na začiatku podľa orientácie) vektora.

2.4 Zhodnotenie

Kapitola popisuje základné termíny a metódy spájané s problematikou rozhodovacích diagramov. V úvode kapitoly je opísaná B-funkcia, jej možný zápis pomocou modifikovanej pravdivostnej tabuľky a odvodený je aj vektor reziduálnych hodnôt v podobe kanonicke matice. Nasledujúca kapitola opisuje možnosti rozkladu B-funkcie, pričom sú spomenuté len tie typy, ktoré majú prínos v oblasti DD. V ďalších podkapitolách sú predstavené rozhodovacie diagramy a ich základné vlastnosti. Pridaný je aj opis voľných rozhodovacích diagramov, ktoré predstavujú záujmovú oblasť práce práve pre ich možnosť maximálnej redukcie so zachovaním úplnej vstupnej informácie.

Opis redukčných pravidiel je nasledovaný opisom metód usporiadania premenných B-funkcie. Existuje niekoľko metód na optimálne usporiadanie vzhľadom na redukciu veľkosti diagramu, práca sa však zaoberá hlavne evolučnými algoritmami, ktorých výhodou je aplikovateľnosť aj na optimalizáciu dekompozícií. Podkapitolu uzatvára opis typov existujúcich diagramov. Posledná časť sa venuje opisu reziduálnej premennej, sú odvodené jej funkcie pre všetky typy rozkladu B-funkcie a predstavené je aj jej integrovanie do DD ako náhrada uzla na najnižšej vrstve diagramu.

Medzi hlavné nedostatky prezentovaných metód patrí ich takmer výhradné používanie na optimalizáciu BDD. Optimalizácia KFDD bola pomerne zanedbávaná vo výskume a preto je zaujímavým smerovaním práve preskúmanie vplyvu metód bežne používaných v BDD práve na KFDD, najmä EA spolu s RV. Druhým hlavným nedostatkom je krátky dosah redukčných pravidiel. Pravidlá I, S a D sa vedia vysporiadať s identickými, ale nie s ekvivalentnými subdiagramami (rozdiel medzi rovnakými a ekvivalentnými subdiagramami je rovnaký ako medzi rovnakými a ekvivalentnými funkciami, opísaný v kapitole 2.1.1). Redukcia ekvivalentných subdiagramov predstavuje rovnaký prínos ako každé z používaných redukčných pravidiel, avšak je možná iba vo voľnej verzii DD (FrDD).

3 Tézy dizertačnej práce

V analytickej časti práce boli identifikované hlavné vlastnosti rozhodovacích diagramov a ich prípadné nedostatky, na základe ktorých vyplývajú nasledovné tézy práce:

- Overenie prínosu reziduálnej premennej (RV) pri Kroneckerových rozhodovacích diagramoch (KFDD), návrh parametrov evolučného algoritmu (EA) na vstupe, syntéza a následná redukcia novovzniknutého KFDD spolu s RV (tzv. RViKFDD).

- Optimalizácia RViKFDD z pohľadu viacerých parametrov.

- Návrh novej metódy na určenie ekvivalencie Booleových funkcií pri preusporiadaní premenných.

- Návrh nového typu voľného rozhodovacieho diagramu (FrDD) uvoľnením pravidla jednej premennej pre každý uzol diagramu – MVDD (Multi-Variable Decision Diagram).

Prvý cieľ práce sa sústreďuje na overenie prínosu RV pri použití všetkých troch dostupných dekompozícii. Pri KFDD je možné použiť RV pred samotnou syntézou ako aj na jej konci jednoduchým nahradením najnižšej úrovne diagramu. Táto práca sa zameriava na prvý prípad, kedy je vstupná B-funkcia transformovaná najskôr na vektor reziduálnych funkcií a potom použitá ako vstup pre syntézu diagramu v spolupráci s EA. Práca sa tiež zaoberá určením vhodných vstupných parametrov pre EA so zameraním na optimalizáciu veľkosti diagramu.

Zámerom nasledujúceho cieľa je optimalizácia RViKFDD z pohľadu viacerých parametrov. Optimalizácia BDD aj RViBDD z pohľadu APL a teoretickej spotreby reprezentovaného obvodu je už dobre známa. Práca sa zaoberá návrhom funkcií približne odhadujúcich teoretickú spotrebu Davio uzlov na základe dobre známych funkcií pre BDD uzly.

Pri niektorých symetrických funkciách dochádza k zaujímavému javu. Dekompozícia aplikovaná na B-funkciu f poskytuje dve výstupné funkcie, ktoré na vstupe majú rovnaké poradie premenných, avšak presusporiadaním poradia v jednej funkcii by sa dosiahla identická funkcia s druhou. Ako príklad je možné uviesť funkciu $f = (10101100)$, na ktorú je aplikovaná Shannonova dekompozícia udávajúca výstupné funkcie $f_L = (1010)$ a $f_P = (1100)$. Pri zvolenom poradí premenných $\{x_i, x_j\}$ je možné preusporiadať funckiu f_P na poradie $\{x_j, x_i\}$ s výsledným vektorom $f_{PP} = (1010)$. Takéto funkcie sa nazývajú ekvivalentné a práca navrhuje novú heuristiku na určovanie ekvivalencie B-funkcií.

Výstupy predchádzajúceho cieľa je možné aplikovať pri syntéze diagramu použitím logiky voľných DD. Práca sa v poslednej téze zaoberá návrhom nového typu DD, jeho vlastnosťami a kompatibilitou s redukčnými metódami, ktorý v jadre využíva navrhovanú heuristiku.

4 Optimalizácia KFDD

K optimalizácii KFDD sa pristupuje so zámerom rozšírenia už dosiahnutých výsledkov, hlavne tých, pre ktoré sú známe výsledky pre BDD ale ešte neboli aplikované na KFDD. Základom kapitoly je hlavne hlbší pohľad na východzí stav, prispôsobenie iniciálneho nastavenia optimalizačných parametrov a aplikovanie niektorých metód v inej fáze redukcie, napríklad aplikovanie RV pred samotnou syntézou. Teoretické podklady a reziduálne funkcie odvodené pre Davio dekompozíciu sú novým prínosom práce, avšak logicky boli umiestnené do kapitoly 2.

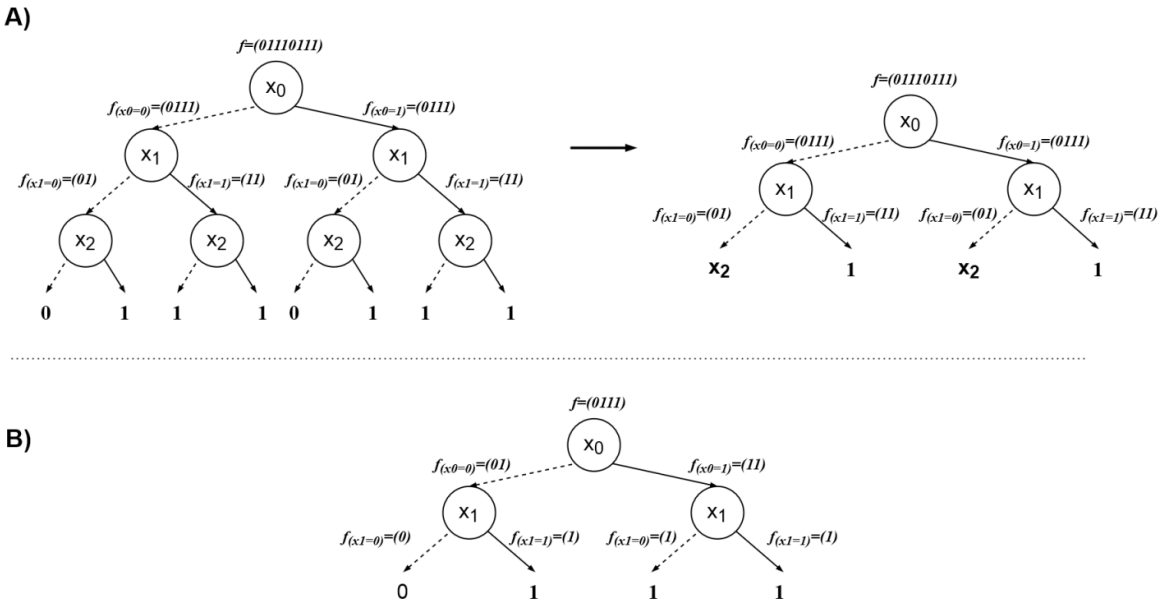
4.1 Reziduálna premenná v KFDD

V kapitole 2.3 sú opísané pravidlá aplikácie RV na vstupnú B-funkciu a na vygenerovanie matice reziduálnych funkcií (vzťah 2.6). Pri syntéze DD existujú 2 spôsoby aplikovania RV. Prvým je syntéza DD z celej vstupnej B-funkcie a aplikácia RV na najnižšiu úroveň, teda nahradenie najnižšej úrovne uzlov hodnotami RV, redukcia diagramu nasleduje až po tomto kroku. Vďaka rovnosti hodnôt RV pri všetkých dekompozíciach je možné aplikovať tento postup na ľubovoľný uzol a teda aj na ľubovoľný typ diagramu. Aplikovanie RV až po syntéze diagramu (pred redukciou) neberie do úvahy možnosť skrátenia dĺžky vstupnej B-funkcie a zníženia zložitosti DD s $n+1$ premennými na DD s n premennými, ktoré sú priamym dôsledkom vektora reziduálnych premenných. Druhý spôsob naproti tomu vstupnú funkciu v prvom kroku transformuje na reziduálny vektor a použije ho ako vstupnú B-funkciu pre syntézu.

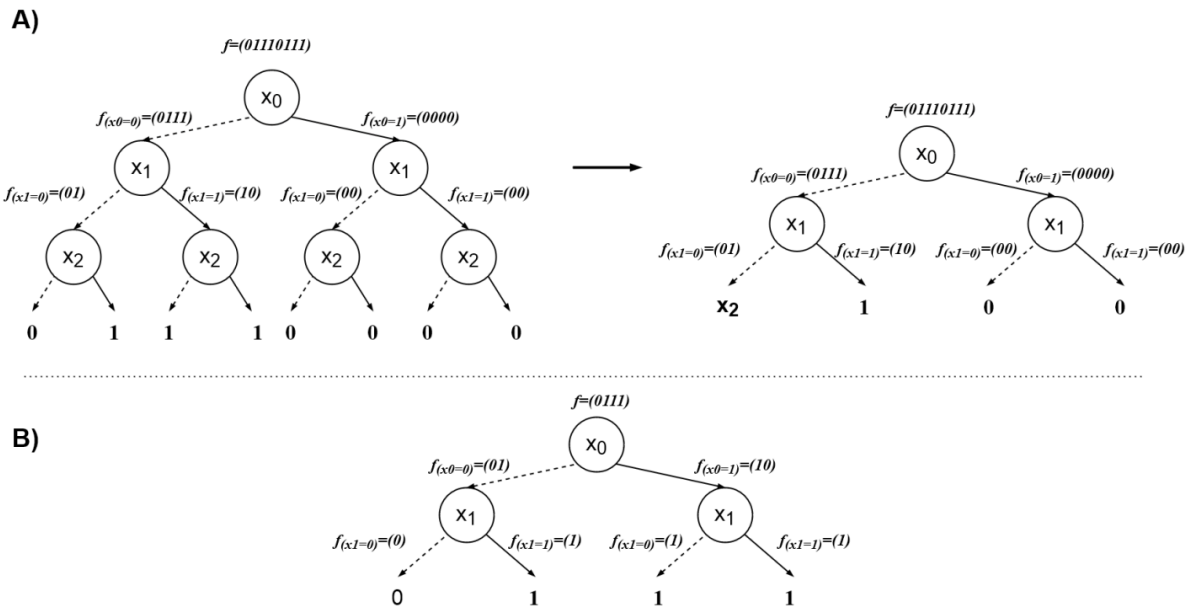
Nastáva otázka, ktorý zo spôsobov je *správny*. Na jednej strane použitie prvého spôsobu zachováva väčšiu podobnosť s pôvodným diagramom a nesie vo všetkých uzloch úplnú informáciu o B-funkcii, avšak za cenu zvýšenej výpočtovej náročnosti (vstupom je vektor hodnôt dĺžky 2^n). Na druhej strane transformácia vstupnej B-funkcie a následná syntéza s redukciou sa javia byť viac v súlade s definíciami a zamýšľaným prínosom RV k optimalizácii DD (vstupom je vektor hodnôt dĺžky 2^{n-1}, teda polovičná dĺžka).

Spôsoby aplikácie RV sú znázornené na obrázkoch 4.1 a 4.2. Na vstupe je zadaná B-funkcia $f = (01110111)$, časť A) znázorňuje syntézu celého diagramu a nahradenie RV na najnižšej vrstve diagramu, časť B) znázorňuje použtie matice reziduálnych funkcií a z nej vyplývajúci vektor reziduálnych funkcií $Z_f = (0111)$. Obrázok 4.1 používa iba Shannonovu dekompozíciu, obrázok 4.2 používa iba pozitívne Davio. Za povšimnutie stoja rôzne terminálne hodnoty.

Druhý spôsob stojí na pevnejších teoretických základoch a využíva výhody RV vo väčšej miere oproti prvému spôsobu. Aj keď je možno vhodné otestovať oba spôsoby a porovnať ich prínos, je pravdepodobnejšie, že druhý spôsob bude prínosnejší v redukcii počtu uzlov. Práca prezentovaná v tomto dokumente sa sústredí na prínos druhého spôsobu, teda variant B z obrázkov 4.1 a 4.2. Všetky potrebné teoretické koncepty sú opísané v analytickej časti práce.



Obrázok 4.1 –RViBDD pri nahradení poslednej vrstvy (A) a pri použití reziduálneho vektora (B)



Obrázok 4.2 –RViFDD pri nahradení poslednej vrstvy (A) a pri použití reziduálneho vektora (B)

4.2 Parametre evolučného algoritmu

Ako už bolo spomenuté, EA je možné použiť v dvoch inštanciách - na poradie premenných a na poradie dekompozícií pre jednotlivé premenné. Pre oba prípady je potrebné určiť niekoľko spoločných parametrov, medzi ktoré patria napríklad:

- Veľkosť celkovej populácie.
- Veľkosť generácie.

- Pravdepodobnosť mutácie.
- Pravdepodobnosť kríženia.
- Funkcia vhodnosti.

Pre vysvetlenie významu jednotlivých parametrov je nutné uviesť poradie vykonávania pri behu algoritmu. Vstupom celého procesu je plne definovaná B-funkcia (existujú metódy na úpravu neúplne definovaných B-funkcií, ale ich optimalizácia nie je cieľom tejto práce) dĺžky 2^n, kde n predstavuje počet premenných funkcie. Navrhovaná verzia algoritmu vykoná v následnosti kroky:

1. Vygenerovanie populácie poradí premenných.
2. Vygenerovanie populácie poradí dekompozícií.
3. Pre každú kombináciu poradí premenných a dekompozícii:
    a. Aplikovanie RV.
    b. Syntéza diagramu.
    c. Redukcia diagramu.

Pri generovaní populácie sa berie do úvahy hodnota počtu premenných n. Pokiaľ je toto číslo dostatočne malé, je jednoduchšie otestovať celú množinu poradí premenných aj množinu poradí dekompozícií (DTL – Decomposition Type List). Hraničnou hodnotou sa pri experimentálnych výsledkoch ukázala hodnota 6 premenných, teda pre vstupy, ktoré majú 6 alebo menej premenných, je otestovaná celá množina o veľkosti 720 poradí premenných ($n!$) a 729 DTL (3^n). Pre väčšie hodnoty n sa aplikuje EA. Samotné použitie EA pri optimalizácii DD nie je novinkou, problémom je určenie vstupných hodnôt. Dodatkovými experimentami boli pre potreby tejto práce a zvolenú implementáciu určené hodnoty opísané nižšie, keďže sa nejedná o nosnú časť práce, výsledky experimentov sú opísané len v krátkosti.

Veľkosť populácie

Pri určovaní vhodnej veľkosti populácie boli vykonané experimenty s viacerými hodnotami stúpajúce po 50 v intervale od 50 až po 500. Menej ako 50 jedincov v populácii by predstavovalo príliš malú vzorku, viac ako 500 jedincov je pre menšie obvody zbytočné a pre väčšie dosahuje hranicu prijateľného výpočtového času. Experimenty prebehli na niekoľkých obvodoch zo sady LGSynth93 [20]. Zvolené hodnoty populácie premenných sú 500 (menej ako 12 vstupných premenných), 200 (medzi 12 až 21 vstupných premenných) a 100 (21 a viac). Od hodnoty 200 vyššie sa dosahovalo iba zanedbateľné zlepšenie a aj to iba vo výnimočných prípadoch, preto bola veľkosť populácie ustálená na 200 pre DTL. Je nutné podotknúť, že sa jedná o 500, resp. 200 unikátnych chromozómov (poradí premenných alebo dekompozícií).

Veľkosť generácie

S veľkosťou generácie prebehli podobné experimenty ako s veľkosťou populácie. Interval testovaných hodnôt bol v rozmedzí 4 až 40 s krokom 4, pri menej ako 4 rodičoch by sa nedosahovala potrebná rozmanitosť nových generácií. Veľkosť generácie však mala väčší vplyv na výpočtový čas ako na samotnú mieru redukcie, hlavne na čas potrebný na dosiahnutie zvolenej populácie. Prijateľné výsledky s dostatočnou rozmanitosťou jedincov boli dosahované pri 20-členných generáciách a so zvyšovaním tohto čísla neboli pozorované významné zmeny, preto bola zvolená výsledná hodnota 20.

Pravdepodobnosť mutácie a kríženia

Určovanie pravdepodobnosti mutácie a kríženia bolo vykonané empiricky. Pre dosiahnutie rozmanitosti je zmena v rodičoch nevyhnutná, či už jedným alebo druhým spôsobom. Za východzí bod bolo zvolené kríženie, ktoré prebieha v každom prípade. Nový chromozóm je získaný krížením dvoch rodičov, ktorí sú náhodné vybraní z predchádzajúcej generácie. Po ošetrení je chromozóm zmutovaný. Opäť boli vykonané experimenty pre rôzne hodnoty od 20% po 100%, žiadna výrazná zmena v rýchlosti ani dosiahnutej redukcii nebola pozorovaná. Pravdepodobnosť kríženia bola teda stanovená na 100% a pravdepodobnosť mutácie na 20%.

Funkcia vhodnosti

Na vygenerovanie novej generácie potomkov je potrebné vybrať vhodných rodičov. Na kvantifikovanie vhodnosti sa používa práve funkcia vhodnosti. V oblasti DD neexistuje ustálená funkcia s preukázateľnou nadradenosťou ostatným metódam a jej výber je témou na samostatný výskum. V [33] je vhodnosť vyhodnocovaná až po redukcii každého rodiča, čím menší diagram, tým vhodnejší rodič, algoritmus je však aplikovaný iba na BDD. Prezentovaná práca naproti tomu oddeľuje kroky generovania populácie a redukcie najmä kvoli zvýšenej zložitosti pri syntéze KFDD oproti BDD (hodnoty terminálnych uzlov nemusia byť vopred známe). Bolo navrhnutých a porovnaných 5 rôznych funkcií vhodnosti:

1. Pozorovanie zmien binárnej hodnoty vo vektore, t.j. počet prechodov z 0 na 1 a naopak.
2. Hammingova vzdialenosť polovíc vektora tak, ako je definovaná v [34].
3. Hammingova vzdialenosť polovíc vektora s jednou polovicou negovanou.
4. Exkluzívny binárny súčet polovíc vektora, pričom ako vhodnosť je použitý počet zmien hodnoty vo výsledku.
5. Exkluzívny binárny súčet polovíc vektora, pričom ako vhodnosť je použitý rozdiel dĺžky štvrtiny vektora a počtu výskytov 0 vo výsledku.

Najprínosnejšou sa ukázala metóda č. 5. Výber funckie vhodnosti je inšpirovaný Davio dekompozíciou, kde je pravý potomok exkluzívnym súčtom oboch polovíc vstupného vektora hodnôt. Pokiaľ by sa na rovnakých pozíciách vo vektore nachádzali rovnaké hodnoty, je zvýšená pravdepodobnosť redukcie typu I. Použitím logickej funkcie XOR na obe polovice by sa teda na miestach s rovnakými hodnotami dosiahol výsledok 0 . Pri určovaní vhodnosti poradia premenných je pre každé poradie preusporiadaný vstupný vektor, jeho polovice su vstupom pre XOR a vo výstupe sa určí počet núl. Pre funkciu $f_1 = (10101010)$ je XOR polovíc rovný $f_{X1} = (0000)$ a pre funkciu $f_2 = (01011010)$ zase $f_{X2} = (1111)$. Obe funkcie však po syntéze a redukcii diagramu dosiahnu rovnako veľký diagram, pričom pre f_{X1} je počet výskytu 0 rovný 4 a pre f_{X2} je rovný 0. Za kritérium vhodnosti sa teda považuje absolútna hodnota vzdialenosti (rozdielu) počtu 0 od dĺžky štvrtiny vektora, teda pomocou vzťahu $dĺžka(f_1)/4 = 2$ je vhodnosť oboch vektorov rovná 2, všeobecne zobrazené vo vzťahu 4.1.

$$vhodnost\ (fitness) = \left| \left(\frac{dlzka(f)}{4}\right) - pocet\_0\left(xor\left(f(x_0 = 0), f(x_0 = 1)\right)\right) \right| \qquad (4.1)$$

Výsledky porovnávacích experimentov nie sú v tejto práci uvedené, pretože sú iba doplnkom k hlavnej téme výskumu. Primárnym cieľom je optimalizácia KFDD a doplnkové experimenty boli vykonané iba na zistenie podkladových hodnôt pre ďalšie experimenty.

Navrhnutý algoritmus vygeneruje pomocou EA populáciu 200 (500) poradí a 200 dekompozícií. Ich kombináciou teda vznikne 40 000 rôznych vstupov pre syntézu a redukciu. Pre každé poradie je preusporiadaný vstupný vektor a vygenerovaný vektor reziduálnych funkcií aplikovaním RV. Reziduálny vektor následne spolu s poradím dekompozícií slúži ako vstup pre syntézu diagramu. Najskôr prebehne syntéza diagramu od koreňa k uzlom aplikáciou iba redukčného pravidla I, ktoré je platné pre všetky uzly rovnako. Po syntéze nasleduje redukcia od najnižšej úrovne, teda od terminálnych uzlov ku koreňu. Algoritmus prechádza od potomka na jednotkových hranách (najviac vpravo pri zvolenom zobrazení v práci) postupne po potomka na nulových hranách (najviac vľavo) na každej úrovni. Keď dosiahne potomka nulových hrán danej úrovne, posunie sa na jednotkového potomka vyššej úrovni.

4.3 Optimalizácia KFDD z pohľadu viacerých parametrov

Veľkosť DD nie je jediný parameter hodný optimalizácie. Najmä pri návrhu obvodov, kde našli DD najväčšie uplatnenie, je vhodné sledovať aj iné parametre, napríklad spotrebu obvodu alebo priemernú dĺžku výstupnej cesty. Pri redukcii veľkosti často existuje viacero riešení s rovnakým (najnižším) dosiahnutým počtom uzlov avšak s rozdielnou spotrebou obvodu. Pri návrhu čipov môže byť, aj za cenu väčšieho počtu uzlov, dôležitejší rozdiel dĺžky jednotlivých ciest v obvode, napr. môže byť prípustný rozdiel najviac 1 úrovne uzlov. Pri *Pass Transistor Logic* (PTL) obvodoch je dĺžka priemernej cesty najdôležitejším parametrom oproti klasickým *Transistor-Transistor Logic* (TTL) obvodom.

Dynamická spotreba obvodu a prepínanie medzi uzlami záleží od rozdelenia uzlov v redukovanom DD, ktoré je priamo závislé od usporiadania premenných. Existuje viacero algoritmov na optimalizáciu spotreby zároveň s veľkosťou obvodu, napr. autori v [36] využívajú dva rôzne algoritmy na usporiadanie premenných s ohľadom na spotrebu.

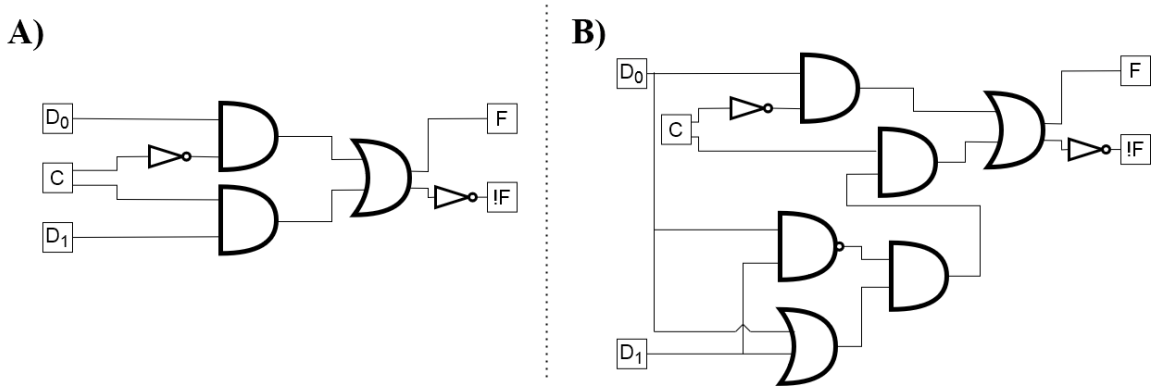
4.3.1 Spotreba obvodu

Práca [33] sa venuje spotrebe multiplexorového stromu pomocou optimalizácie BDD a čerpá informácie z [19], kde sa uvádza súvis dynamickej spotreby s dátovými vstupmi obvodu. Dynamická spotreba P_D pre multiplexor typu 2:1 je vyjadrená vzťahom 4.2-a, kde $OP(F)$ predstavuje pravdepodobnosť hodnoty 1 na výstupe F. Hodnota $OP(F)$ je vyjadrená vo vzťahu 4.2-b, kde D_0 a D_1 predstavujú dátové vstupy multiplexora a C predstavuje kontrolný vstup. Výpočet celkovej spotreby obvodu z multiplexorov je určený ako súčet spotrieb jednotlivých multiplexorov, teda $P = \sum_{i=0}^{u} P_D(u_i)$.

$$P_D = 2 * OP(F) * (1 - OP(F)) \tag{4.2-a}$$

$$OP(F) = OP(D_0) * \big(1 - OP(C)\big) + OP(D_1) * OP(C) \tag{4.2-b}$$

Autor sa v práci [33] zaoberá len BDD, ktorého uzly je možné priamo preklopiť na 2:1 multiplexor. Davio uzly disponujú mierne komplikovanejšou stavbou. Ukážka rozdielu vnútornej štruktúry 2:1 multiplexora zostaveného z logických hradiel AND a OR pre Shannonovu aj pozitívnu Davio dekompozíciu je znázornená na obrázku 4.3. Ako vidieť, použité logické hradlá sá výrazne líšia. Prepínanie vnútri takýchto multiplexorov však záleží na použitej architektúre, napríklad v [37] je uvedené porovnanie 3 rôznych prístupov k návrhu multiplexorov.



Obrázok 4.3 –Multiplexor 2:1 pre Shannonovu (A) a pozitívnu Davio (B) dekompozíciu

Z obrázku 4.3 sa môže zdať jasné, že spotreba Davio multiplexora bude vyššia ako spotreba Shannonovho multiplexora. Vzhľadom na vyšší počet hradiel je pravdepodobná vyššia statická spotreba, ktorá v sebe zahŕňa napríklad úniky prúdov v tranzistoroch. Táto práca sa však zaoberá

spotrebou čisto z teoretického pohľadu, každý uzol reprezentuje časť obvodu s neznámou štruktúrou (*black-box*) a spotreba uzla sa vyjadruje iba ako pravdepodobnosť prepínania v danom uzle. Dynamická aj statická spotreba obvodu sú závislé od zvolenej architektúry použitého prvku. DD môžu slúžiť na reprezentáciu viacerých typov obvodov, jedným z nich sú už spomenuté stromové štruktúry pozostávajúce z multiplexorov, môžu to však byť napríklad optické obvody [4], kvantové obvody [16] alebo pamäťové rezistory [23]. Pre účel vyjadrenia teoretickej spotreby uzla preto postačia funkcie odvodené pre BDD uzly v [33]. Rovnaké funkcie budú použité na Davio uzly, pričom sa bude sledovať rozdiel spotreby pre redukcie s najnižšou veľkosťou diagramu. Týmto spôsobom sú vyjadrené iba teoretické spotreby obvodov pre porovnanie rozdielov medzi BDD a KFDD a preukázanie použiteľnosti optimalizačnej metódy aj na KFDD, spotreba iného typu obvodu je námetom pre ďalší výskum.

4.3.2 Dĺžka priemernej výstupnej cesty

Priemerná dĺžka výstupnej cesty (Average Path Length - APL) určuje čas potrebný na vyhodnotenie funkcie aplikovaním poradia hodnôt premenných a má priamy vplyv na rýchlosť vykonávania obvodovo realizovanej funkcie. APL reprezentuje priemer všetkých výstupných ciest v diagrame a na jej optimalizáciu existuje viacero metód, štandardne rozdelených do dvoch skupín – statické a dynamické metódy.

Statické metódy minimalizácie APL upravujú usporiadanie premenných pred syntézou diagramu vyhodnotením vlastností B-funkcie, napríklad Walshove koeficienty alebo Walshove spektrum [38]. Do tejto skupiny možno zaradiť aj prácu [39], ktorá sa venuje analýze zložitosti častí vstupnej funkcie a statickému preusporiadaniu premenných s ohľadom na APL. Dynamické metódy naproti tomu iteratívnym postupom upravujú DD a porovnávajú ho s odhadovanou funkciou. Vykonaním zmien v DD je možné zamietnuť niektoré poradia vzhľadom na optimalizáciu APL. Príkladom dynamických metód je [40], kde sa APL upravuje lokálnymi zmenami v diagrame.

Na výpočet APL existuje niekoľko metód. V práci [41] autor opisuje 3 rôzne metódy, pričom prvá z nich využíva binárne rozhodovacie stromy a im ekvivalentné diagramy. V tejto metóde má každá hrana hodnotu 1 a dĺžka cesty k uzlu je súčet hrán vyskytujúcich sa na ceste od koreňa k danému uzlu. V druhej metóde autor využíva postupné skladanie binárneho stromu s ohodnotením terminálnych uzlov a v tretej autokorelačné hodnoty podfunkcií, všetky metódy však vedú k rovnakému výsledku. Práca [42] opisuje heuristickú metódu založenú na preosievaní premenných pred syntézou BDD. APL je v tejto práci určovaná ako pravdepodobnosť výskytu uzla na ceste od koreňa k terminálnemu uzlu. Koreň, nachádzajúci sa v každej ceste, má danú pravdepodobnosť 1. Pri binárnych DD má každý uzol práve dvoch potomkov, každý potomok má pravdepodobnosť prechodu rovnú polovici pravdepodobnosti jeho rodiča. Pri redukovaných diagramoch môže viacero uzlov odkazovať na toho istého potomka, ktorého pravdepodobnosť prechodu je rovná súčtu polovíc pravdepodobností všetkých

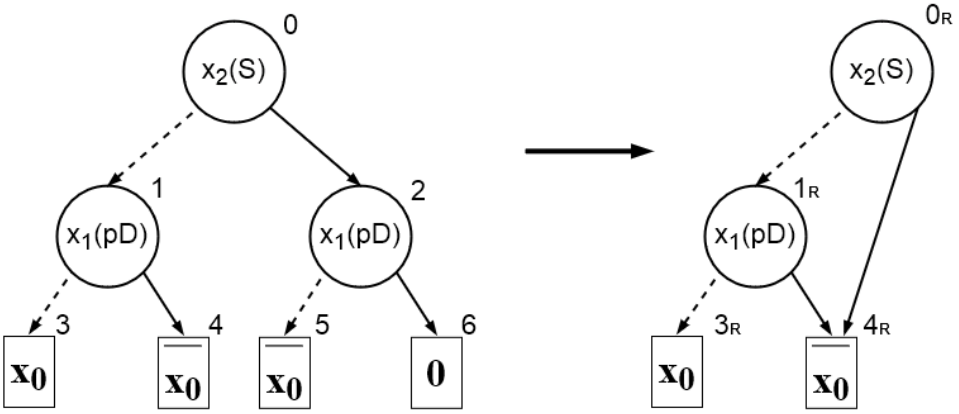
rodičov. APL celého diagramu je potom určená ako súčet všetkých pravdepodobností uzlov $APL = \sum_{i=0}^{n} P(u_i)$.

K dátumu písania tejto práce nie je známy žiaden výskum zaoberajúci sa APL pri KFDD. V práci [43] je uvedený teorém s dôkazom, ktorý poukazuje na BDD pre paritné funkcie. Za paritné funkcie sa považujú také, ktoré využívajú logický XOR alebo jeho doplnok. Tento dôkaz je však uvedený iba pre BDD, využitie dekompozície postavenej práve na funkcii XOR (akou je Davio dekompozícia) môže priniesť zaujímavé výsledky s ohľadom na APL, ako je aj ukázané v neskorších kapitolách.

4.3.3 Príklad výpočtu spotreby a APL pre RViKFDD

Podkapitola obsahuje zjednodušený príklad výpočtu spotreby a APL pre RViKFDD. Použité vzťahy sú identické so vzťahmi opísanými v predchádzajúcich kapitolách.

Príklad 4.1: Na vstupe je zadaná B-funkcia $f_4 = (01101100)$ s poradím premenných $X_n = \{x_0, x_1, x_2\}$ a poradím dekompozícií $DTL = \{S, S, pD\}$. Aplikovaním RV na x_0 so Shannonovou dekompozíciou sa získa vektor reziduálnych funkcií $Z_4 = (x_0, 1, \overline{x_0}, 0)$. RViKFDD a jeho redukcia pre f_4 s indexovaním uzlov je zobrazený na obrázku 4.4.



Obrázok 4.4 –RViKFDD a jeho redukovaná verzia pre f_4 z Príkladu 4.1

Výpočet teoretickej spotreby prebieha zdola nahor. Terminálnym uzlom je priradená pravdepodobnosť hodnoty 1 na výstupe, teda pre terminálny uzol 0 je táto pravdepodobnosť rovná 0, pre terminálny uzol 1 je rovná 1 a pre reziduálne premenné x_i a $\overline{x_i}$ je priradená pravdepodobnosť 0,5. Každý nasledujúci uzol na vyššej úrovni má na základe vzťahu 4.2-a vyjadrenú pravdepodobnosť výstupu 1 ako priemer svojich potomkov. Spotreba uzla je vyjadrená dosadením do vzťahu 4.2-b. Spotreba celého diagramu je vyjadrená ako súčet spotrieb všetkých uzlov. Pre redukovaný diagram z príkladu 4.1 je teda spotreba určená:

1. Terminálnym uzlom 3_R a 4_R je priradená pravdepodobnosť $OP(3_R) = OP(4_R) = 0{,}5$.

2. Uzlu 1_R je priradená pravdepodobnosť priemeru jeho potmokov $OP(1_R) = (0{,}5 + 0{,}5)/2 = 0{,}5$. Spotreba je vyjadrená dosadením hodnôt $P(1_R) = 2 * OP(1_R) * \big(1 - OP(1_R)\big) = 0{,}5$
3. Koreňu diagramu je rovnakým postupom určená pravdepodobnosť $OP(0_R) = (OP(1_R) + OP(4_R))/2 = 0{,}5$ a spotreba $P(0_R) = 2 * OP(0_R) * \big(1 - OP(0_R)\big) = 0{,}5$
4. Spotreba celého diagramu je vypočítaná ako súčet jednotlivých spotrieb, teda $P(RViKFDD) = 0{,}5_{1R} + 0{,}5_{0R} = 1$ v teoretických jednotkách spotreby.

Výpočet APL prebieha zhora nadol. Postupnosť priraďovania pravdepodobností prechodu je nasledovná:

1. Koreňu 0_R je priradená pravdepodobnosť $P(0_R) = 1{,}0$.
2. Potomkom koreňa je priradená polovičná pravdepodobnosť, teda uzly 1_R a 4_R majú pravdepodobnosť prechodu $P(1_R) = P(4_R) = 0{,}5$.
3. Potomkom uzla 1_R je priradená polovičná pravdepodobnosť $0{,}25$. Uzol 4_R už má priradenú pravdepodobnosť z kroku 2, nová pravdepodobnosť je prirátaná už k existujúcej $P(4_R) = 0{,}5 + 0{,}25 = 0{,}75$. Každý uzol okrem koreňa má teda pravdepodobnosť prechodu rovnú súčtu polovičných pravdepodobností svojich rodičov.
4. Výsledné APL diagramu je určené ako súčet hodnôt všetkých uzlov diagramu okrem terminálnych, teda $APL(RViKFDD) = 1{,}0_{0R} + 0{,}5_{1R} + 0{,}25_{3R} + 0{,}75_{4R} = 2{,}5$.

4.4 Prínos k optimalizácii KFDD

Kapitola sa venuje opisu optimalizácie KFDD z pohľadu viacerých parametrov. Na úvod sú popísané parametre evolučného algoritmu, ktorý je kľúčovou optimalizáciou pri diagramoch s väčším počtom vstupných premenných. Uvedené sú zistené hranice hodnôt, pri ktorých prestáva narastať pridaná hodnota parametra, napr. veľkosť populácie alebo veľkosť generácie. Je nutné spomenúť, že vplyv opísaných hodnôt platí iba pri zvolenej implementácii. Mieru vplyvu EA na optimalizáciu DD ovplyvňuje viacero faktorov, medzi nimi najmä výber programovacieho jazyka, vývojového prostredia a asi v najväčšej miere návrh algoritmu, jeho kód a konfigurácia.

Kapitola pokračuje opisom RV v KFDD a dvoch možných prístupov k aplikovaniu RV. Táto práca sa zaoberá hlavne prístupom, ktorý sa javí ako *správny* na základe definície samotnej RV. Aj keď optimalizácia KFDD nie je novinkou v oblasti DD, publikovaných prác na túto tématiku je pomerne málo v porovnaní s BDD, ktoré sú základom drvivej väčšiny výskumu. Jedným z prínosov práce je návrh optimalizácie RViKFDD z pohľadu veľkosti diagramu, spotreby a APL.

Kapitola pokračuje odvodením funkcií pre výpočet spotreby a priemernej dĺžky výstupnej cesty v KFDD na základe prác [19], [42] a ich využitia v [33] pre BDD. Uzly s Davio dekompozíciou majú rozdielnu štruktúru od Shannonových uzlov, ich spotreba však závisí od zvolenej architektúry uzla, preto sa k uzlom KFDD pristupuje ako čiernym skrinkám a je odhadnutá iba teoretická spotreba na základe prepínania v prvku obvodu. Jedným z výstupov práce je aj návrh KFDD s použitím RV, tzv. RViKFDD.

5 Rozhodovací diagram s viacnásobnými premennými

Vplyv na optimalizáciu DD má viacero faktorov, medzi ktoré patria použité dekompozície, funkcie vhodnosti zvolených heuristík alebo modifikácie štruktúry DD akými je napríklad použitie FrDD. Z množstva doteraz známeho výskumu je však jasné, že oblasťou s najväčším záujmom a pravdepodobne aj s najväčším prínosom je preusporiadanie premenných. Priamy vplyv usporiadania premenných B-funkcie na veľkosť výsledného diagramu bol už niekoľkokrát dokázaný experimentálnymi výsledkami aj teoretickým opisom [2], [11], [45]. Hlavným úskalím je exponenciálne rastúca zložitosť (veľkosť) množiny všetkých poradí s narastajúcim počtom premenných. Povolením rôznych poradí v subdiagramoch táto zložitosť opäť neúmerne narastá. Z tohto dôvodu sa problematike voľných DD venuje relatívne málo prác (napríklad v porovnaní s BDD) avšak teoretické podklady pre výskum v tejto oblasti sú dobre známe, už opísané v kapitole 2.2.1.

Hlavným problémom FrDD je nedostatok heuristík na zmenšenie záujmovej oblasti premenných, inými slovami ako identifikovať prípady, kedy má význam sa zaoberať preusporiadaním v subdiagramoch a kedy nie. Dosiahnutie *najlepšieho* výsledku je možné len otestovaním všetkých kombinácií poradí. Pri optimalizácii KFDD bol pozorovaný jeden jav, ktorý sa môže ukázať ako krok správnym smerom. Pri redukcii sa pomerne často vyskytovali funkcie, ktoré boli tvorené dvoma rovnakými vektormi, avšak s rôznym usporiadaním premenných. Napríklad funkcia $f_{(x_0x_1x_2)} = (11001010)$ po použití Shannonovej dekompozície na x_0 poskytuje dva subdiagramy so vstupnými funkciami $f_{L(x_1x_2)} = (1100)$ a $f_{P(x_1x_2)} = (1010)$. Obrátením poradia premenných v f_P na (x_2x_1) je možné dosiahnuť rovnaký binárny vektor ako pri f_L, teda $f_{P(x_2x_1)} = (1100)$. Vstupná funkcia f má po preusporiadaní premenných identických potomkov, čo poskytuje priestor pre syntézu a redukciu iba jedného potomka a aplikovanie výsledku na druhého. Čím vyššie v úrovniach diagramu je takáto zhoda objavená, tým väčšia teoretická úspora času a prostriedkov pri redukcii diagramu.

Práca sa v nasledujúcich podkapitolách zaoberá návrhom metódy na zistenie podobnosti dvoch binárnych vektorov s rovnakou dĺžkou. Keďže motivácia pre tento problém sa nachádza v doméne rozhodovacích diagramov, uvažujú sa iba vektory dĺžky 2^n, pre ktoré je možné zostrojiť plne definovanú pravdivostnú tabuľku. V čase písania tejto práce nie je známy algoritmus ani heuristika riešiaca tento problém v lineárnom čase, ako už bolo spomenuté v [11], jedná sa o NP problém. Práca sa ďalej zaoberá návrhom nového typu diagramu, ktorý je schopný implementovať výhody vyplývajúce z predchádzajúcej metódy a zároveň zachovať plnú informáciu o vstupnej funkcii, ako aj kompatibilitu so všetkými predstavenými optimalizačnými metódami.

5.1 Ekvivalencia Booleových funkcií

V kapitole 2.1.1 je v krátkosti opísaný význam ekvivalencie B-funkcií spolu s príkladmi exstujúcich postupov na jej určovanie. Cieľom tejto práce je navrhnúť novú metódu na riešenie ekvivalencie dvoch B-funkcií a s ohľadom na problematiku DD. K dnešnému dňu nie je známy výskum zaoberajúci sa ekvivalenciou funkcií a subdiagramov DD.

5.1.1 Booleova funkcia ako graf

Základom pre navrhovanú metódu je teória grafov. Izomorfizmus grafov zisťuje podobnosť dvoch grafov na základe definovaných vlastností, akými sú počet uzlov, stupne uzlov, počet alebo orientácia hrán medzi uzlami. Ako bolo dokázané v [49], jedná sa o kombinatorický problém s faktoriálnou zložitosťou. Všetky grafy je možné zaznačiť graficky alebo pomocou tabuľky obsahujúcej predpis grafu. Tabuľka grafu obsahuje zoznam všetkých uzlov, hrán medzi nimi a práve jej nápadná podobnosť s pravdivostnou tabuľkou B-funkcií boli inšpiráciou pre navrhovanú metódu. Pravdivostná tabuľka plne definovanej B-funkcie predstavuje kompaktný zápis grafu *G* s nasledujúcimi vlastnosťami:

- Každú B-funkciu je možné zakresliť ako graf pomocou indexovanej pravdivostnej tabuľky.

- Uzlami grafu *G* je množina *V* obsahujúca všetky premenné vstupnej B-funkcie a indexy z pravdivostnej tabuľky $V = \{x_0, \dots, x_n, 0{,}1, \dots, 2^n - 1\}$. Množina uzlov je rozdelená na dve podmnožiny V_V a V_I, kde $V_V = \{x_0, \dots, x_n\}$ obsahuje iba uzly premenných a $V_I = \{0{,}1, \dots, 2^n - 1\}$ iba indexové uzly.

- Hrany grafu existujú iba medzi uzlami z rôznych skupín uzlov, nikdy nie medzi uzlami v rámci skupiny. Každý uzol z podmnožiny V_V môže mať hrany iba s uzlami z podmnožiny V_I (s viacerými naraz) a naopak.

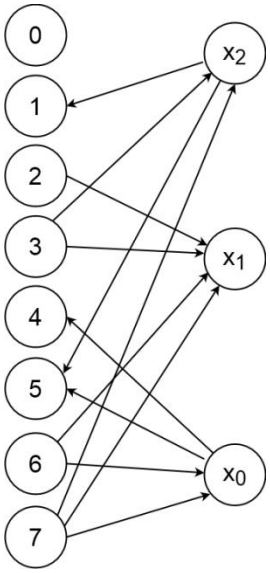
- Hrana medzi uzlom premennej a uzlom indexu existuje iba v prípade, kedy je hodnota premennej na danom indexe tabuľky rovná 1 (premenná *sa podieľa* na výstupe funkcie), zapísané ako $\exists E\left(V_{V_i}, V_{I_j}\right) \underset{i<n}{\Longleftrightarrow} I_j(x_i) = 1$, kde $I_j(x_i)$ predstavuje hodnotu premennej x_i na indexe j.

- Orientácia hrany je určená výstupnou hodnotou funkcie na danom riadku tabuľky, resp. hodnotou výstupného vektora na danej pozícii. Hodnota funckie 0 predstavuje orientáciu hrany z uzla premennej do uzla indexu, značené ako $E\left(V_{x_i}, V_{I_j}\right) = V(x_i) \rightarrow V(I_j)$. Hodnota funkcie 1 predstavuje opačnú orientáciu, z indexu do premennej, značené ako $E\left(V_{x_i}, V_{I_j}\right) = V(x_i) \leftarrow V(I_j)$.

Príklad 5.1: Na vstupe je zadaná B-funkcia s množinou premenných $X_A = \{x_0, x_1, x_2\}$, vektorom $f_A = (10110011)$ dĺžky $2^n = 8$. Pravdivostná tabuľka funkcie je znázornená v tabuľke 5.1, stĺpec I predstavuje indexy výsledných hodnôt funkcie (zoznam indexových uzlov). Graf vyplývajúci z pravdivostnej tabuľky je znázornený na obrázku 5.1.

Tabuľka 5.1 – Pravdivostná tabuľka pre f_A z príkladu 5.1

I	x_0	x_1	x_2	y
0	0	0	0	1
1	0	0	**1**	**0**
2	0	**1**	0	**1**
3	0	1	1	1
4	**1**	0	0	**0**
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1



Obrázok 5.1– Graf pre f_A z príkladu 5.1

Zakreslením pravdivostnej tabuľky ako grafu sú zjavnejšie niektoré vlastnosi B-funkcie, ktoré nie sú očividné z tabuľkového alebo vektorového predpisu. Každý uzol premennej má priradený jeden indexový uzol, s ktorým má práve jednu hranu. Tento fakt nie je vďaka povahe B-funkcie a binárnej sústavy prekvapivý, indexové uzly s práve jednou hranou sú mocniny čísla 2, t.j. 1, 2 a 4, zvýraznené v tabuľke 5.1. Každý z uzlov premennej má konečný počet vstupných a výstupných hrán, na základe ktorých je možné uzly ohodnotiť. Za ohodnotenie uzla sa teda považuje počet vstupných a výstupných hrán, zapísaných ako $\{vstup, vystup\}$. Podobným spôsobom je možné ohodnotiť indexové uzly a zaradiť ich do tried podľa početnosti hrán. Pri triedach indexových uzlov sa ignoruje orientácia hrán a do úvahy sa berie iba súčet všetkých hrán pripojených na uzol. V grafe pre funkciu f_A z príkladu 5.1 existujú 4 triedy indexových uzlov podľa počtu hŕan – uzly s 0, 1, 2 a 3 hranami ľubovoľného smeru. Ohodnotenie všetkých uzlov je možné znázorniť tabuľkovým zápisom, pre príklad 5.1 zobrazeným v tabuľke 5.2. Šípky v bunkách tabuľky reprezentujú orientáciu hrany z indexového uzla do uzla premennej (↑) alebo naopak (↓). Šípky tiež prislúchajú hodnotám vstupnej funkcie, šípka ↑ prislúcha logickej hodnote 1 a naopak. Stĺpce *Premenná* a *Ohodnotenie* zobrazujú tú istú informáciu s rôznym zápisom. Rovnako je možné pridať tretí stĺpec obsahujúci hodnoty z výstupného vektora y, pre prehľadnosť však boli zvolené šípky a množinový zápis ohodnotenia.

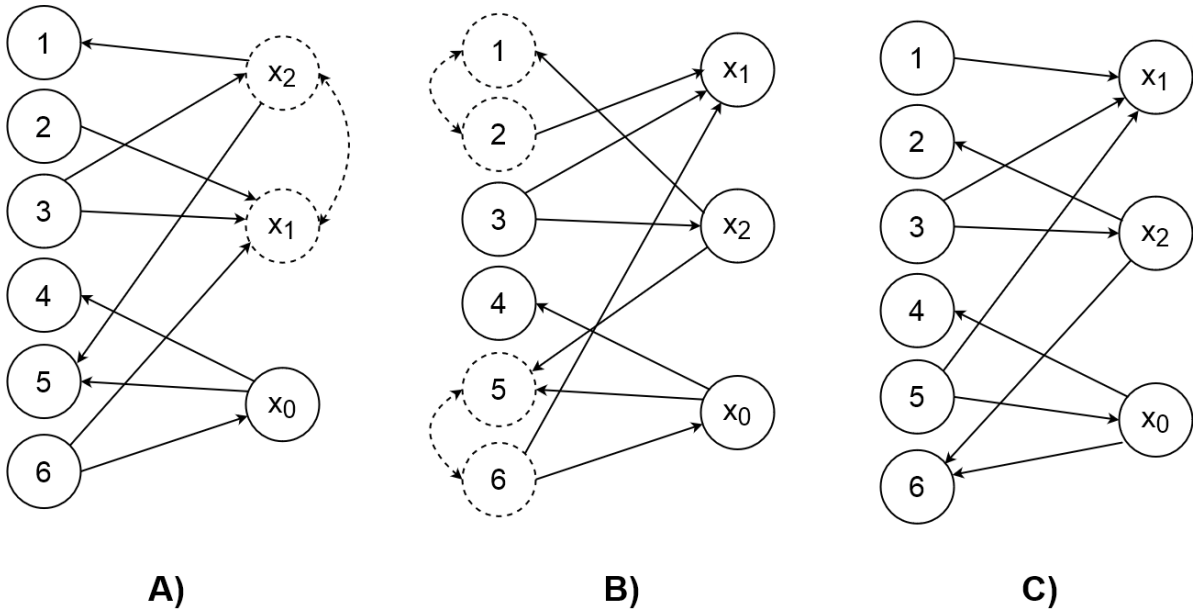
Tabuľka 5.2 – Ohodnotenie uzlov pre f_A z príkladu 5.1

Trieda hrany	Premenná			Ohodnotenie *{vstup, výstup}*		
	x_0	x_1	x_2	x_0	x_1	x_2
0	−	−	−	{0,0}	{0,0}	{0,0}
1	↓	↑	↓	{0,1}	{1,0}	{0,1}
2	↓↑	↑↑	↑↓	{1,1}	{2,0}	{1,1}
3	↑	↑	↑	{1,0}	{1,0}	{1,0}

Pri ľubovoľnom preusporiadaní premenných funkcie je možné pozorovať dva javy. Prvým je fakt, že bez ohľadu na permutáciu poradia, sú koncové indexové uzly 0 a $2^n - 1$ vždy na rovnakom mieste s rovnakými hranami. Oba uzly tvoria samostatné triedy hrán s počtom vstupujúcich alebo vystupujúcich hrán rovným 0 pre indexový uzol 0 alebo 1 pre indexový uzol $2^n - 1$. Koncové indexové uzly a ich triedy hrán je preto možné zanedbať a sústrediť sa iba na uzly medzi nimi. Druhým javom je zachovanie ohodnotenia hrán pri všetkých permutáciách poradia. Orientácie aj počet hrán medzi uzlami ostávajú nezmenené, mení sa len pozícia uzlov v grafe. Inými slovami, ľubovoľná B-funkcia a každá jej preusporiadaná verzia majú izomorfné grafy s rovnakým ohodnotením hrán.

Príklad 5.2: Na vstupe sú zadané 2 funkcie, $f_A = (10110011)$ s poradím premenných $X_A = \{x_0, x_1, x_2\}$ (rovnaká ako v príklade 5.1) a funkcia $f_B = (11010101)$, ktorá je jej preusporiadanou verziou s poradím premenných $X_B = \{x_0, x_2, x_1\}$. Zmeny v pravdivostnej tabuľke pri preusporiadaní premenných sú znázornené v tabuľke 5.3, zmeny v grafe na obrázku 5.2, tabuľka ohodnotenia je znázornená v tabuľke 5.4. Z grafu sú vynechané koncové indexové uzly. Označenie premenných je ponechané po preusporiadaní, t.j. premenná x_2 ostáva tou istou premennou aj na novej pozícii. Indexy riadkov ostávajú nezmenené, pri výmene v druhom kroku sa presunie len obsah riadku na pozíciu s novým indexom.

Obrázok 5.2 zobrazuje funkciu f_A (A) a jej postupnú zmenu na funkciu f_B (C). Presúvanie uzlov znamená presunutie hrán pripojených na daný uzol na uzol na cieľovej pozícii. Pri presunutí hrán pripojených na uzly premenných bolo presunuté aj označenie premenných (B) aby odrážalo zmenu poradia. Pri presúvaní indexových uzlov boli presunuté iba hrany (C), poradie indexových uzlov ostáva rovnaké aby odrážalo stav v usporiadanej pravdivostnej tabuľke. Inými slovami, presunutie obsahu riadku v pravdivostnej tabuľke prislúcha presunutiu hrán v grafe so zachovaním označenia indexových uzlov.



Obrázok 5.2 – Zmeny pri preusporiadaní premenných funkcie $\mathrm{f_A}$ *z* $\{\mathrm{x_0}, \mathrm{x_1}, \mathrm{x_2}\}$ *na* $\{\mathrm{x_0}, \mathrm{x_2}, \mathrm{x_1}\}$

Tabuľka 5.3 – Zmeny pri preusporiadaní f_A *z* $\{x_0, x_1, x_2\}$ *na* $\{x_0, x_2, x_1\}$

I	x_0	x_1	x_2	y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

➡

I	x_0	x_2	x_1	y
0	0	0	0	1
1	0	1	0	0
2	0	0	1	1
3	0	1	1	1
4	1	0	0	0
5	1	1	0	0
6	1	0	1	1
7	1	1	1	1

➡

I	x_0	x_2	x_1	y'
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Tabuľka 5.4 – Ohodnotenie uzlov pre f_A *a* f_B *z príkladu 5.2*

Trieda hrany	f_A						f_B					
	Premenná			Ohodnotenie			Premenná			Ohodnotenie		
	x_0	x_1	x_2	x_0	x_1	x_2	x_0	x_2	x_1	x_0	x_2	x_1
1	↓	↑	↓	{0,1}	{1,0}	{0,1}	↓	↓	↑	{0,1}	{0,1}	{1,0}
2	↓↑	↑↑	↑↓	{1,1}	{2,0}	{1,1}	↓↑	↑↓	↑↑	{1,1}	{1,1}	{2,0}

Z tabuľky 5.4 je zrejmé, že pri preusporiadaní premenných ostáva ohodnotenie uzlov rovnaké mení sa len poradie v akom sa vyskytujú v tabuľke (zmena poradia stĺpcov v časti *Ohodnotenie*). Pri zisťovaní ekvivalencie dvoch B-funkcií bude označenie premenných rozdielne a je potrebné

určiť, ktorá premenná v jednej funkcii prislúcha premennej v druhej funkcii. Párovanie premenných z dvoch rôznych funkcií je možné práve na základe ich ohodnotenia. V niektorých prípadoch však priradenie nemusí byť jednoznačné, napr. premenné x_0 a x_2 majú rovnaké ohodnotenie a je možné ich zaradiť do jednej skupiny. Premenná x_1 je však jednoznačne identifikovateľná svojim unikátnym ohodnotením.

5.1.2 Heuristika zmenšenia množiny poradí

Pomocou krokov v príkladoch 5.1 a 5.2 je možné definovať ohodnocovanie premenných ako samostatnú funkciu. Pre potreby všeobecného zápisu sú zavedené nové definície, ktoré nadväzujú na všeobecne známe definície B-funkcie v kapitole 2.1.

Definícia 5.1: Množina premenných X_n funkcie f, resp. množina kofaktorov je rozdelená do tried $C_{i=0}^{n}$ podľa početnosti premenných rovných 1 v kofaktoroch funkcie. Napríklad pre funkciu $f_{n=3}$ sú kofaktory 1. triedy funkcie $C_1 = \{f(001),\ f(010), f(100)\}$ a kofaktory 2. triedy $C_2 = \{f(011),\ f(101), f(110)\}$. Z definície vyplýva, že veľkosť triedy C_0 je vždy rovná 0 a veľkosť C_n je vždy rovná 1, pre zadanú funkciu to sú triedy $C_0 = \{\emptyset\}$ a $C_3 = \{f(111)\}$.

Definícia 5.2: Daná je taká ohodnocovacia funkcia ε, ktorá priradí každej premennej x_i množinu početnosti hodnôt 0 a 1 v takých kofaktorových triedach, v ktorých platí $x_i = 1$. Pre triedu C_1 je toto priradenie rovné $\varepsilon_1(x_i) = \{\sum_{x_i=1}(C_1), \sum_{x_i=0}(C_1)\}$, analogicky platí $\varepsilon_2(x_i) = \{\sum_{x_i=1}(C_2), \sum_{x_i=0}(C_2)\}$ atď. Výsledná hodnota ε je rovná $\varepsilon(x_i) = \{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n\}$.

Definícia 5.3: Premenné funkcie je možné zaradiť do skupín na základe hodnoty ε, premenné s rovnakou ε patria do tej istej skupiny, označené ako $S_x(f) = \{s_1 = [x_i, x_j, \dots], s_2 = [x_k, x_l, \dots], \dots\}$, pričom platí s_1: $[\varepsilon(x_i) = \varepsilon(x_j) = \varepsilon(\dots)]$; s_2: $[\varepsilon(x_k) = \varepsilon(x_l) = \varepsilon(\dots)]$, atď.

Na základe vyššie uvedených definícii je možné opísať heuristiku pre určenie ekvivalencie dvoch vektorov. Vzhľadom na možnosť existencie premenných s rovnakým ohodnotením sa jedná len o heuristiku, nie o algoritmus s konečným počtom krokov. Heuristika vychádza z predpokladu, že vstupné vektory sú ekvivalentné a snaží sa postupnými krokmi čo najrýchlejšie určiť nepravdivosť predpokladu. Vylučovaním nutných podmienok sa snaží dostať k zmenšenej množine poradí premenných, ktoré môžu viesť k ekvivalentným vektorom.

Postupným ohodnocovaním premenných vznikajú skupiny premenných s rovnakou hodnotou ε. Pre funkcie z príkladu 5.2 sú týmito skupinami $S_x(f) = \{[x_0, x_2], [x_1]\}$, prepísané hodnotami ε ako $S_\varepsilon = \{[\{0,1\}, \{1,1\}], [\{1,0\}]\}$.

Aby boli dve funkcie f_A a f_B, resp. ich vektory ekvivalentné, musia spĺňať novovytvorenú definíciu ekvivalencie opísanú ako:

1. Hodnoty koncových (okrajových) kofaktorov oboch funkcií musia byť identické, t.j. $f_A(00 \ldots 0) = f_B(00 \ldots 0)$ a zároveň $f_A(11 \ldots 1) = f_B(11 \ldots 1)$.

2. Hammingova váha oboch funkcií musí byť rovnaká, t.j. $H(f_A) = H(f_B)$.

3. Skupiny premenných oboch funkcií podľa ohodnocovacej funkcie ε musia byť identické, t.j. $S_\varepsilon(f_A) == S_\varepsilon(f_B)$.

4. Funkcia f_A je určená ako východzia funkcia. Premenné funkcie f_B sú preusporiadané tak, aby boli ich skupiny podľa ohodnocovacej funkcie v ronakom poradí ako v f_A. Pokiaľ nie je vektor funkcie f_B po preusporiadaní zhodný s vektorom funkcie f_A, pokračuje sa v preusporiadavaní premenných, avšak v tomto kroku sa vymieňajú už iba premenné v rámci skupiny.

5. Pokiaľ sú vyčerpané všetky kombinácie poradí premenných podľa skupín a ani jeden zo vzniknutých vektorov f_B nie je zhodný s vektorom f_A, je možné s istotou tvrdiť, že funkcie nie sú ekvivalentné.

V ideálnom prípade navrhnutá heuristika jednoznačne identifikuje pozíciu každej premennej jedinečným ohodnotením funkciou ε, v tomto prípade je zložitosť rovná $O(n)$. Z definície sú jasné, aj empiricky boli zistené prípady, kedy sú hodnoty ohodnocovacej funkcie ε rovnaké pre všetky premenné, zložitosť navrhovanej heuristiky preto ostáva $O(n!)$.

Príklad 5.3: Sú zadané dve funkcie f_A a f_B, úlohou je zistiť, či sú dané funkcie ekvivalentné. Dĺžka funkcií je rovná 32, počet premenných je teda $n = 5$. Premenné funkcie f_A budú označené $X_A = \{x_0, x_1, x_2, x_3, x_4\}$a premenné funkcie f_B označené ako $X_B = \{x_a, x_b, x_c, x_d, x_e\}$.

$$f_A = (10000101111101010001110100110111)$$
$$f_B = (10011101001100010001110101011111)$$

V prvom kroku podľa navrhovanej heuristiky je potrebné skontrolovať okrajové hodnoty funkcie, v tomto prípade platí $f_A(00000) = f_B(00000) = 1$ a $f_A(11111) = f_B(11111) = 1$.

V druhom kroku sa kontroluje Hammingova váha oboch funkcií, ktorá je rovná $H(f_A) = H(f_B) = 18$.

Premenné je možné s trochou cviku ohodnocovať priamo z vektora hodnôt, nie je potrebné vytvárať pravdivostnú tabuľku. Hodnoty ε pre premenné z oboch funkcií sú zobrazené v tabuľke 5.5, rozdelené do riadkov podľa tried kofaktorov pre prehľadnosť. Triedy C_0 a C_5 sú vynechané, ich rovnosť je kontrolovaná v prvom kroku heuristiky.

Tabuľka 5.5 – Hodnoty ε pre funkcie z príkladu 5.3

	$\varepsilon(x_0)$	$\varepsilon(x_1)$	$\varepsilon(x_2)$	$\varepsilon(x_3)$	$\varepsilon(x_4)$
f_A	{1,0} {3,1} {3,3} {4,0}	{0,1} {2,2} {3,3} {4,0}	{0,1} {2,2} {3,3} {4,0}	{0,1} {1,3} {4,2} {4,0}	{0,1} {2,2} {5,1} {4,0}
	$\varepsilon(x_a)$	$\varepsilon(x_b)$	$\varepsilon(x_c)$	$\varepsilon(x_d)$	$\varepsilon(x_e)$
f_B	{0,1} {1,3} {4,2} {4,0}	{1,0} {3,1} {3,3} {4,0}	{0,1} {2,2} {3,3} {4,0}	{0,1} {2,2} {3,3} {4,0}	{0,1} {2,2} {5,1} {4,0}

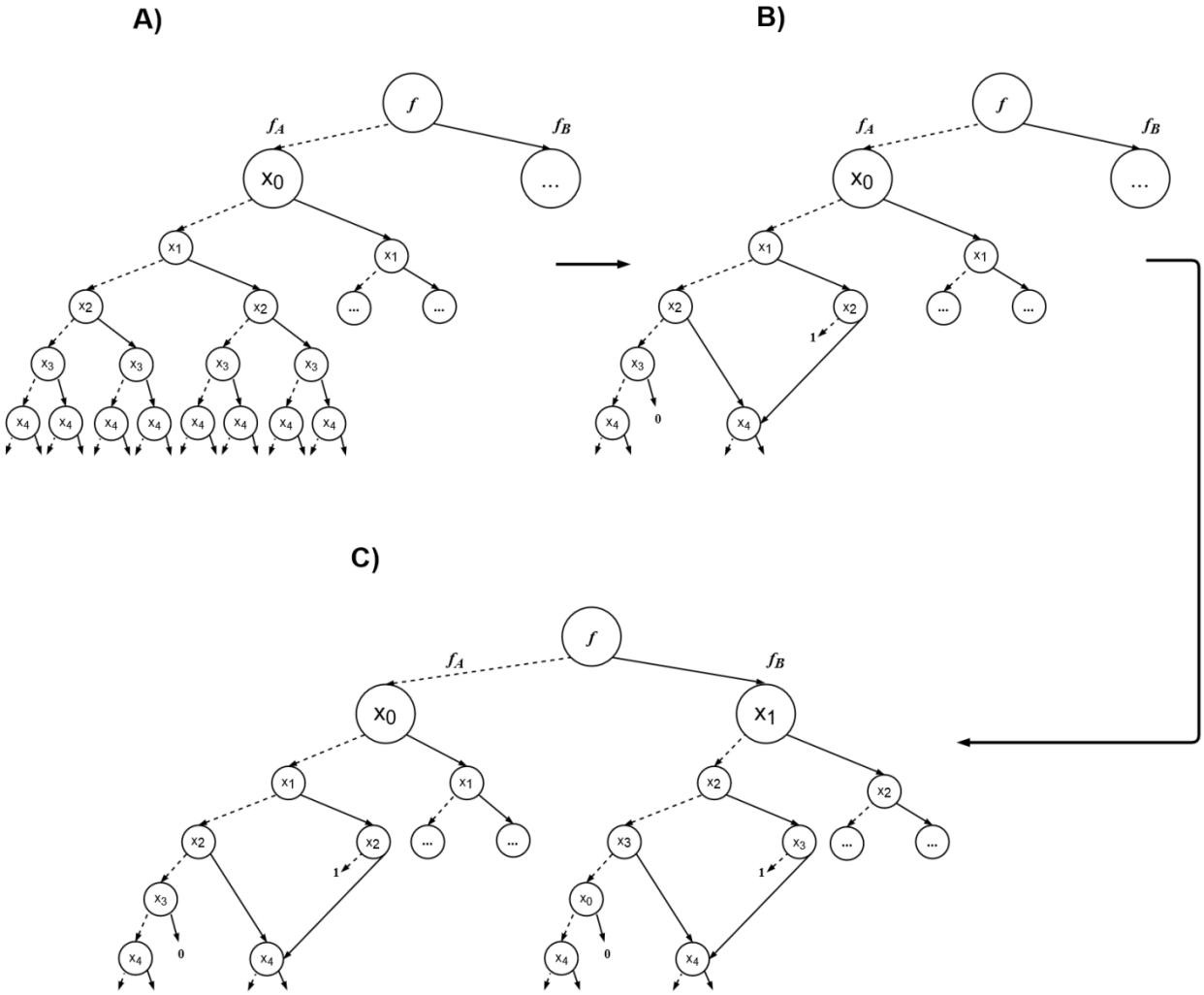
Na základe tabuľky 5.5 je možné zaradiť premenné do skupín. Funkcia f_A je zvolená za východziu funkciu , premenné funkcie f_B, resp. ich skupiny sú preto preusporiadané tak, aby sa premenné s rovnakou hodnotou ε nachádzali na tej istej pozícii ako vo funkcii f_A.

$$S(f_A) = \{[x_0], [x_1, x_2], [x_3], [x_4]\}$$
$$S(f_B) = \{[x_b], [x_c, x_d], [x_a], [x_e]\}$$

Zo vzniknutých skupín je jasné, že existujú iba dve poradia premenných funkcie f_B, ktorých vektor hodnôt može byť ekvivalentný s vektorom hodnôt f_A , t.j. poradia $X_{B1} = \{x_b, x_c, x_d, x_a, x_e\}$ a $X_{B2} = \{x_b, x_d, x_c, x_a, x_e\}$, získané permutovaním premenných na všetkých pozíciach v rámci skupiny. Pri usporiadaní X_{B1} sa zmení vektor f_B na vektor $f_{B1} = (10000101111101010001110100110111)$, ktorý je identický s vektorom f_A. Je preto možné tvrdiť, že vektor f_B je ekvivalentný s vektorom f_A.

V najhoršom prípade budú mať všetky premenné rovnakú hodnotu ε, vtedy existuje len jedna spoločná skupina premenných a veľkosť množiny všetkých poradí na preskúmanie je rovná $n!$. Existujú však prípady, ako je uvedené aj v príklade 5.3, kedy je možné nielen určiť ekvivalenciu vektorov, ale aj presne priradiť premenné jednej funkcie k ich náprotivkom v druhej funkcii.

V stave, v akom bola heuristika prezentovaná, vie teoreticky ušetriť čas pri optimalizácii diagramu. Ak by boli funkcie f_A a f_B z príkladu 5.3 výstupom napr. Shannonovej dekompozície funkcie f, je možné vykonať syntézu a redukciu diagramu len pre f_A a aplikovať identickú kópiu výsledného subdiagramu na f_B so zmeneným poradím premenných tak, ako je znázornené na obrázku 5.3. Premenné funkcie f_B sú ekvivalentné premenným funkcie f_A, je ale potrebné ich preusporiadať podľa poradia v skupine $S(f_B)$, nové poradie pre f_B je teda rovné $X_B = (x_1, x_2, x_3, x_0, x_4)$, z usporiadania skupín vyplýva rovnosť $x_b = x_0$; $x_c = x_1$; $x_d = x_2$; $x_3 = x_a$; $x_4 = x_e$.



Obrázok 5.3 – Syntéza subdiagramu f_A (A), jeho redukcia (B) a aplikovanie kópie s upraveným poradím premenných na f_B (C). Pre jednoduchosť sa uvádza len BDD bez použitia RV, niektoré uzly sú vynechané z priestorových dôvodov

Jedným z nedostatkov navrhovanej heuristiky je zvýšená výpočtová zložitosť. Pri optimalizácii je potrebné pre každú funkciu subdiagramu zisťovať ekvivalenciu so všetkými ostatnými funkciami na tej istej úrovni, čo neúmerne zvyšuje čas potrebný na syntézu optimalizovaného diagramu. Proti tomuto tvrdeniu je možné vzniesť dva protiargumenty. Pre každú funkciu je postačujúce určiť ohodnotenia premenných iba raz a zapamätať si výsledky pri ďalších výpočtoch. Druhým argumentom je fakt, že vyhodnotenie prvých dvoch krokov heuristiky je pomerne *lacné*, čo sa týka potrebných prostriedkov, a dostatočne rýchlo je možné určiť, či existuje možnosť ekvivalencie pre porovnávané funkcie. Matematicky je možné dokázať, že prvý krok zmenšuje počet funkcií, pre ktoré má význam zisťovať ekvivalenciu, na jednu štvrtinu (existujú 4 možné kombinácie okrajových hodnôt: {00,01,10,11}). Druhým pravidlom je taktiež eliminovaná podstatná časť zvyšných funkcií. Označením dĺžky vektora hodnôt ako l, dĺžky vektora bez okrajových hodnôt ako m (platí $m = l - 2$) a Hammingovej váhy funkcie ako h je možné vyjadriť veľkosť prehľadávanej množiny kombinačným číslom $\binom{m}{h}$ a veľkosť celej

množiny ako 2^m. Nevýhodou ostáva existencia dvoch identických subdiagramov s rôznym poradím premenných. Situácia sa nápadne podobá na predpoklady potrebné pre redukčné pravidlá I alebo S. Intuícia napovedá, že musí existovať praktické riešenie podobnej situácie, ktorým sa zaoberá nasledujúca kapitola.

5.2 Rozhodovací diagram s viacnásobnými premennými

Pri základných typoch DD, akými sú BDD alebo KFDD, je možné použiť na redukciu uzla s identickými potomkami redukčné pravidlo S. Za identických potomkov sa považujú také uzly, ktoré majú rovnakú vstupnú funkciu, rovnakú dekompozíciu a rovnaké vstupné poradie premenných. Práve posledné z vymenovaných kritérií je problémom pri redukcii voľných rozhodovacích diagramoch. Na obrázku 5.3, časť C), je zobrazená situácia, kde má koreňový uzol s označením f, potomkov s identickou štruktúrou diagramu, jediným rozdielom je rôzne vstupné poradie premenných do oboch potomkov.

V kapitole 2.2 je uvedených 6 implicitných pravidiel tvorby binárnych diagramov, pre pripomenutie sú znova uvedené pravidlá 3. a 4:

- Pravidlo 3: Každý uzol v diagrame reprezentuje práve jednu premennú.
- Pravidlo 4: Všetky cesty od koreňa k terminálnemu uzlu produkujú rovnaký vektor premenných.

Uvoľnením pravidla 3 a povolením viacnásobných premenných v jednom uzle sa automaticky ruší účinnosť pravidla 4. Pri syntéze diagramu je vstupom vektor hodnôt, vektor dekompozícií a vektor premenných. Na každej vrstve sa vstupná funkcia rozloží podľa premennej a dekompozície na prvej pozícii v prislúchajúcich vektoroch a do nižších vrstiev sa *posielajú* vektory bez práve použitej premennej alebo dekompozície. Napr. pri poradí $X_f = \{x_0, x_1, x_2\}$ a $DTL_f = \{S, pD, nD\}$ sa na premennú x_0 použije dekompozícia S a do nižších vrstiev diagrame pokračuje ako vstup poradie $X_1 = \{x_1, x_2\}$ a $DTL_1 = \{pD, nD\}$ atď. Povolením viacerých premenných pre každý uzol je potrebné zabezpečiť prešírenie všetkých poradí do všetkých uzlov na nižších vrstvách diagramu. Z algoritmického hľadiska je každý uzol diagramu dátovou štruktúrou, ktorá udržiava okrem iného údaje:

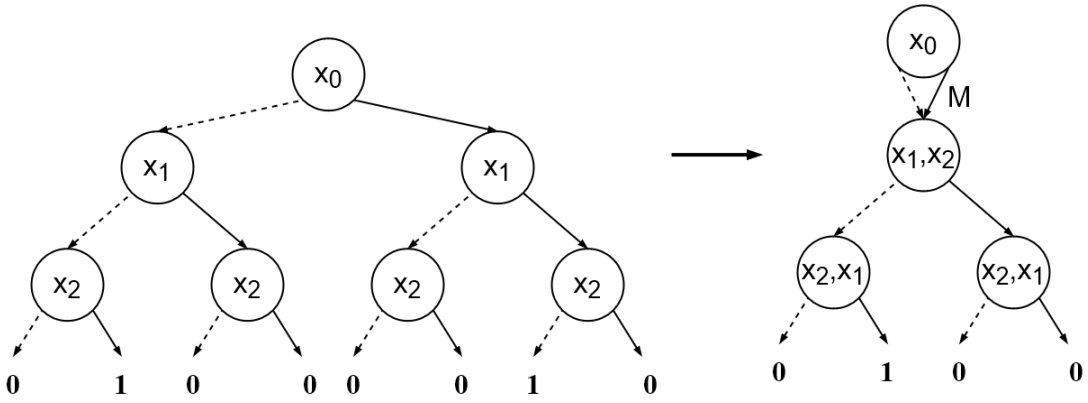
- Premenná *p*, jedna inštancia konkrétneho dátového typu.
- Dekompozícia *d*, jedna inštancia konkrétneho dátového typu.
- Vektor hodnôt *v*, pole konkrétneho dátového typu.

Navrhovaný postup mení prvý udržiavaný údaj na:

- Vektor premenných *p*, pole konkrétneho dátového typu.

Príklad 5.4: Na vstupe je zadaná funkcia $f = (01000010)$ s poradím $X = \{x_0, x_1, x_2\}$, pre jednoduchosť je použitá len Shannonova dekompozícia, t.j. $DTL = \{S, S, S\}$. Rozložením f podľa premennej x_0 vzniknú dve funkcie pre ľavého a pravého potomka rovné $f_L = (0100)$ a $f_P = (0010)$ s poradím $X_L = X_P = \{x_1, x_2\}$. Už na prvý pohľad je jasné, že f_P je ekvivalentné s f_L prehodením poradia na $X'_P = \{x_2, x_1\}$ s prislúchajúcou úpravou vektora hodnôt na $f'_P = (0100)$. Prevod BDD na DD s viacnásobnými premennými v uzloch pre funkciu f je znázornený na obrázku 5.4.

Konceptuálne je opísaná zmena jednoduchá. Vstupom pre každý uzol ostáva vektor hodnôt a prislúchajúce poradie premenných, prípadne množina poradí, ktorými je možné daný vektor dosiahnuť. Pri dodržaní pravidla jednej dekompozície na uzol zostáva zachovaná množina výstupných hodnôt aj vnútorná štruktúra uzla, ktorý je predlohou pre konštrukciu logických obvodov. Novým problémom je potreba udržiavania informácie o tom, ktorá premenná sa vyskytuje na ktorej ceste od koreňa k terminálnym uzlom. Existuje viacero možností ako sa s daným problémom vysporiadať.



Obrázok 5.4 – Prevod BDD z príkladu 5.4 na diagram s viacnásobnými premennými. Prísmeno M predstavuje tzv. označenie MVDD

Jedným z riešení zachovania jednoznačnej cesty je značenie hrán. Na obrázku 5.4 je označená pravá hrana diagramu písmenom *M*, ktoré určuje výber druhej množiny poradí premenných vo všetkých uzloch nižších vrstiev. Pri prechádzaní diagramom a skladaní cesty, resp. vektora poradí, sa vyberajú premenné z množín poradí podľa označenia hrany. Štandardne platí výber prvej premennej z každého vektora premenných. Pri skladaní zoznamu od koreňa po nulového potomka (najviac vľavo) sa do úvahy berú premenné v poradí $x_0 - x_1 - x_2$, pretože ľavá hrana nemá priradenú značku. Skladaním cesty k jednotkovému potomkovi (najviac vpravo) sa docieli postupnosť premenných $x_0 - x_2 - x_1$, pretože pravý potomok koreňa má priradenú značku *M*, ktorá platí na všetky uzly v nižších vrstvách, teda aj na posledný neterminálny uzol v pravej časti diagramu.

Navrhovaný diagram bude označený ako rozhodovací diagram s viacnásobnými premennými v uzloch (MVDD – Multi-Variable Decision Diagram). V čase písania tejto práce je už zaužívaná skratka MDD používaná pre diagramy s viacerými výstupmi (Multi-valued DD), ktoré stále nachádzajú uplatnenie v praktickej oblasti [50] aj s ich voľnou verziou v [51] (jeden zo zoznamov zaužívaných skratiek je uvedený aj v [2]). MDD povoľuje na výstupe viac ako dve (viac ako štyri v prípade RV) výstupne hodnoty pre každý uzol z preddefinovanej množiny.

MVDD je priamo z definície formou FrDD a je kompatibilný s oboma používanými typmi dekompozície – Shannon a Davio, čo má za následok aj kompatibilitu s redukčnými pravidlami s jednou malou modifikáciou. Redukčné pravidlo typu S by mohlo byť podľa základnej definície použité na uzol x_0 v MVDD na obrázku 5.4 (diagram vpravo), uzol spĺňa všetky podmienky – použitá dekompozícia je Shannon a ľavý potomok je rovný pravému. Redukciou uzla by boli odstránené aj jeho hrany a tým by sa stratila aj informácia o zmene poradia. Redukčné pravidlo S je preto rozšírené o jednu podmienku a pre jeho aplikáciu na uzol *u* v MVDD musí platiť:

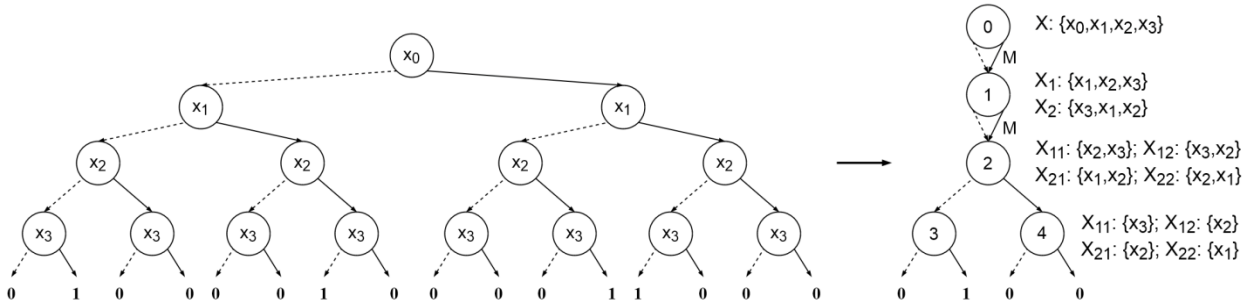
- Na uzol u je použitá Shannonova dekompozícia, $dek(u) = S$.
- Ľavý potomok je rovný pravému potomkovi, ľ$avý(u) = pravý(u)$.
- Vstupné premenné do ľavého aj pravého potomka sú identické (obidve hrany vystupujúce z uzla u sú bez označenia MVDD).

MVDD je taktiež kompatibilný s RV, ktorá je z poradia premenných odstránená ešte pred syntézou diagramu. Vektor reziduálnych funkcií je vygenerovaný pred syntézou diagramu a pozícia RV ostáva nemenná v každej fáze optimalizácie, úprava poradia pomocou MVDD preto môže prebehnúť bez ovplyvnenia výsledných hodnôt. Kompletne redukovaný MVDD zachováva plnú informáciu vstupnej funkcie a predstavuje kanonickú formu DD.

Postupným skladaním skratiek podľa použitej optimalizácie, napr. usporiadania (O - Ordered), redukcie (R - Reduced) alebo RV (RVi – Residual Variable in), a podľa použitej dekompozície (S, pD, nD) sa docieli jedna z možných kombinácií rozhodovacieho diagramu, medzi ktoré patria okrem iných aj OMVDD, ROMVDD, ROMVBDD, ROMVFDD, ROMVKFDD, RORViMVBDD, RORViMVFDD a RORViMVKFDD. Pre potreby tejto práce sa za MVDD považuje RORViMVKFDD pokiaľ nie je uvedené inak, t.j. v diagrame sú použité všetky dekompozície spolu s RV, diagram je usporiadaný a redukovaný.

Príklad 5.5: Na vstupe je zadaná funkcia $f = (0100001000011000)$ s poradím $X = \{x_0, x_1, x_2, x_3\}$, pre jednoduchosť je použité $DTL = \{S, S, S, S\}$. Rozložením f podľa premennej x_0 vzniknú dve funkcie $f_L = (01000010)$ a $f_P = (00011000)$ s poradím $X_L = X_P = \{x_1, x_2, x_3\}$. Funkcia f_P je ekvivalentná s funkciou f_L prehodením poradia na $X'_P = \{x_3, x_1, x_2\}$, MVDD pre f_L je opísaný v príklade 5.4. Prevod BDD na MVDD pre f (bez použitia redukčných pravidiel) je znázornený na obrázku 5.5. Označenie používaného poradia v MVDD je zobrazené

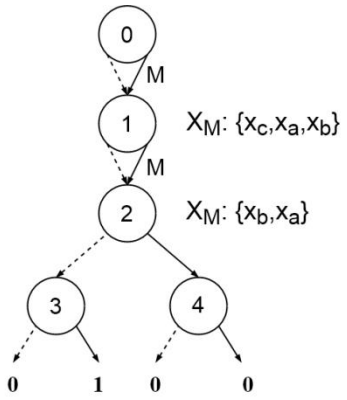
vedľa uzlov ako zoznam premenných, vnútri uzlov sa nachádzajú ich indexy. Dekompozíciou funkcie podľa premennej x_0 a preusporiadaním premenných jednotkovej časti (opísané vyššie ako f_L a f_P) sa prechádza do uzla s indexom 1. Použitie MVDD logiky na pravého potomka je označené značkou *M*. Uzol 1 má dve možné vstupné poradia X_1 a X_2. Jeho dekompozícia a preusporiadanie sú opísané v príklade 5.4, na jednotkového potomka je opäť použitá logika MVDD a vznikajú dve nové poradia pre každé vstupné poradie. Keďže uzol 1 mal dve vstupné poradia, uzol 2 má štyri vstupné poradia, dve pre každé poradie vstupujúce z uzla 1.



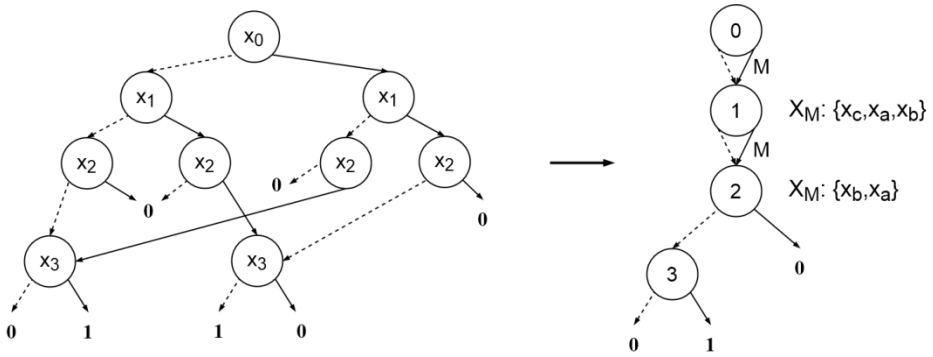
Obrázok 5.5 – Prevod BDD z príkladu 5.5 na MVDD

Uchovávanie všetkých možných poradí premenných tak, ako je znázornené na obrázku 5.5, je zavádzajúco komplikované. Existuje však jednoduchší (z pohľadu množstva udržiavanej informácie) spôsob zápisu, pre každý uzol je postačujúce udržiavať symbolické preusporiadanie pre označenú hranu. Vstupom pre celú funkciu f zostáva poradie $X = \{x_0, x_1, \dots, x_{n-1}\}$, vstupom pre každý uzol nižších vrstiev (od potomkov koreňového uzla po rodičov terminálnych uzlov) je symbolické poradie $X_S = \{x_a, x_b, \dots\}$ rovné poradiu X bez prvej premennej (tá je použitá v rodičovskom uzle). Pri uzloch na druhej úrovni by teda platilo $X_S = \{x_a = x_1, x_b = x_2, \dots\}$ atď. V štandardnom zápise základných typov DD sa v každom uzle uvedie prvá premenná zo vstupujúceho zoznamu a zvyšok sa pošle do nižších vrstiev. Pri MVDD sa bude predpokladať podobný, tentokrát symbolický, vstup a ak bude jedna zo vstupujúcich hrán označená značkou *M*, pri uzle bude uvedené zmenené symbolické poradie premenných X_M, východzie poradie X_S sa zo zápisu môže vynechať, jeho prítomnosť sa automaticky predpokladá. Vyššie opísané značenie pre MVDD z príkladu 5.5 je zobrazené na obrázku 5.6.

Pre porovnanie je na obrázku 5.7 uvedený redukovaný BDD aj redukovaný MVDD pre f z príkladu 5.5. Z oboch diagramov je možné vyskladať úplne definovanú funkciu f. Z obrázku 5.7 jasne vidno pridanú hodnotu MVDD oproti BDD pri redukcii veľkosti diagramus. Zatiaľ čo redukovaný BDD má 9 uzlov, redukovaný MVDD má veľkosť iba 4 uzly, čo predstavuje zmenšenie veľkosti o viac ako 55%. Ešte vyššiu mieru redukcie by prinieslo pridanie ďalších optimalizačných metód, napr. RV. Samozrejme, takto vysoká miera redukcia nebude dosiahnutá vo všetkých prípadoch a v niektorých z nich môže dôjsť k zvýšeniu výpočtového času (vyhodnocovaním ekvivalencie) bez akéhokoľvek zvýšenia miery redukcie.



Obrázok 5.6 – Iný zápis MVDD pre f z príkladu 5.5



Obrázok 5.7 – Redukovaný BDD a MVDD pre f z príkladu 5.5

V úvode práce sa spomína, že oblasťou, v ktorej našli DD najväčšie uplatnenie, je návrh logických obvodov. BDD uzly je možné priamo transformovať na určité logické členy, napríklad na 2:1 multiplexor, a táto vlastnosť ostáva zachovaná aj pri MVDD uzloch, pribúda však potreba zakomponovať možnosť viacerých vstupných poradí do jedného uzla. Rozhodnutie, ktoré poradie, resp. ktorá premenná je vstupom pre daný uzol, je možné vykonať viacerými spôsobmi, pri menších počtoch premenných to môže byť multiplexor, ktorého kontrolné vstupy sú premenné vyšších vrstiev, pri väčších počtoch je možné navrhnúť novú časť obvodu, ktorá preusporiada premenné do nového poradia. Riešenie záleží od viacerých faktorov akými je reprezentovaný prvok/prvky obvodu, zvolená architektúra, prípadne nároky na veľkosť, oneskorenie alebo spotrebu.

5.3 Prínos MVDD

V úvode kapitoly je predstavená nová heuristika na určenie ekvivalencie dvoch plne definovaných B-funkcií o dĺžke 2^n. Cieľom bolo vyvinúť novú metódu použiteľnú pri optimalizácii rozhodovacích diagramov. Ekvivalencia B-funkcií je oblasť, ktorá nie je v počítačových vedách žiadnym nováčikom, významné uplatnenie našla napríklad pri riešení splniteľnosti B-funkcie, tzv. SAT riešiče (SATisfiability solver). V použitých zdrojoch pre kapitolu sú uvedené dve metódy [46] [47], pri ktorých bolo poukázané na nápadnú podobnosť s predstavenou heuristikou, najmä pri prvej z nich. Metóda opísaná v [47] vychádza z termov B-

funkcie a podľa nej zaraďuje premenné do tried, zatiaľ čo heuristika navrhnutá v tejto práci vychádza z pravdivostnej tabuľky funkcie a podkladom pre ňu je teória grafov. Podobnosť s inými, už existujúcimi metódami, nie je vylúčená, dokonca je pravdepodobné, že niektoré metódy stavajú na rovnakých princípoch. Výskum v tejto oblasti bol však zámerne minimalizovaný pre zabezpečenie nového (*čerstvého*) pohľadu na problematiku bez rizika ovplyvnenia zaužívanými postupmi alebo krokmi, ktoré sú považované za samozrejmosť.

Zložitosť navrhovanej metódy je v kapitole vyjadrená v dvoch krokoch. Prvým je pomer prehľadávanej množiny B-funkcií oproti celkovému počtu všetkých možných funkcií. Veľkosť prehľadávanej množiny je určená kombinačným číslom $\binom{m}{h}$, ktoré predstavuje všetky unikátne kombinácie logických hodnôt 0 a 1 pri zachovaní ich početnosti vo funkcii, m je rovné dĺžke funkcie bez okrajových hodnôt a h predstavuje Hammingovu váhu funkcie. Veľkosť množiny všetkých možných funkcií je vyjadrená ako súčet všetkých kombinácií početností 0 a 1 pre danú dĺžku m, vyjadrené súčtom $\sum_{i=0}^{m}\binom{m}{i} = 2^m$. Pomer veľkostí je teda možné vyjadriť ako $\binom{m}{h}/2^m$. V najhoršom prípade je $h = (m/2)$. Druhým krokom vyjadrenia zložitosti je počet možných poradí, pre ktoré je potrebné preusporiadať vektor hodnôt pre potvrdenie ekvivalencie funkcií. V tomto prípade zložitosť záleží od vlastností funkcie, t.j. od možnosti jednoznačne identifikovať premenné podľa hodnôt funkcie. V ideálnom prípade je každá premenná unikátna a postačuje jediné preusporiadanie, v najhoršom prípade sú premenné ohodnotené rovnako (pomocou definovanej ohodnocovacej funkcie ε) a prehľadávaná množina poradí má pri počte premenných n veľkosť $n!$.

V druhej časti kapitoly je navrhnutý nový typ rozhodovacieho diagramu s viacnásobnými premennými v uzloch, označený skratkou MVDD. Pri redukovaní KFDD a úpravách hodnôt evolučného algoritmu bol pomerne často pozorovaný výskyt ekvivalentných funkcií na tej istej úrovni. Prirodzene sa naskytla otázka – *keď už existuje riešenie tejto funkcie, prečo ho nie je možné aplikovať na funkciu s iným poradím?* Intuitívnou odpoveďou bolo presmerovanie hrany na uzol so známym riešením a overiť kompatibilitu s existujúcimi optimalizačnými metódami už bolo triviálne. Takýto postup ale priniesol jeden problém, ktorý ostáva otvorený pre rôzne implementačné postupy – problém zachovania informácie o poradí. V kapitole sú uvedené dva spôsoby, ako sa s touto prekážkou vysporiadať pri DD ako dátovej štruktúre, udržiavaním kompletného zoznamu všetkých vstupných poradí v každom uzle alebo udržiavaním iba symbolického preusporiadania v prípade uzlov so vstupujúcou označenou hranou.

Navrhované metódy poskytujú zaujímavé výsledky napriek tomu, že teoretická zložitosť najhoršieho prípadu ostáva nezmenená. Zjednodušene povedané, aj keď sa pripúšťa možnosť zbytočného skomplikovania a predĺženia optimalizácie diagramu (ako vo všetkých nových metódach), je prínos novej heuristiky v kombinácií s MVDD zjavný a ťažko zanedbateľný, najmä pri redukcii počtu uzlov DD. MVDD bolo aplikované ako ďalšia úroveň optimalizácie nad KFDD z kapitoly 4, z logického hľadiska predstavuje nadstavbu nad optimalizáciou KFDD.

6 Experimentálne výsledky

Na overenie prínosu navrhnutých metód bola použitá sada testovacích obvodov LGSynth93 [20] z dôvodu možnosti porovnania s podobnými prácami. V kapitole je opísaný graduálny prínos optimalizačných metód postupne od jednoduchého BDD, cez KFDD až po MVDD, vždy s použitím RV. Hlavným cieľom bola optimalizácia veľkosti DD redukciou počtu uzlov a popritom boli sledované sekundárne parametre ako spotreba alebo APL. Na porovnanie sú uvedené výsledky z prác, ktoré sa zaoberajú hlavne redukciou BDD.

Parametre vybraných obvodov (funkcií) zo sady LGSynth93 sú uvedené v tabuľke 6.1. Pri obvodoch s viacerými výstupmi boli funkcie optimalizované samostatne a za výslednú hodnotu sa považuje súčet hodnôt dosiahnutých pri každej funkcii samostatne.

Tabuľka 6.1 – Počet vstupných a výstupných funkcií testovacích obvodov

Testovací obvod	Počet vstupných premenných	Počet výstupných funkcií
cm150a	21	1
mux	21	1
t481	16	1
parity	16	1
cm151a	12	2
cm152a	11	1
sao2	10	4
9sym	9	1
sqrt8	8	4
rd84	8	4
misex1	7	7
inc	7	9
5xp1	7	10
con1	6	2
xor5	5	1
squar5	5	8
rd53	5	3
majority	5	1

Testovanie prebiehalo niekoľkonásobným behom algoritmu a zaznamenávaním najlepších výsledkov. Správnosť implementácie a dosiahnutého diagramu bola overovaná ručné, pri väčších diagramoch boli použité voľne dostupné nástroje na overenie výsledných vektorov hodnôt pre neoptimalizovaný aj optimalizovaný diagram (sledovalo sa zachovanie výstupných hodnôt).

Navrhnuté metódy a heuristiky boli implementované v jazyku Python. Syntéza a následná redukcia DDs pre testovacie obvody boli vykonané na stroji so 16GB operačnej pamäte a CPU Intel Core i7-4700MQ 2.4GHz s operačným systémom Windows 8.1.

6.1 Porovnanie RViBDD a RViKFDD

Poradie premenných je preukázateľne faktor s najväčším vplyvom na výslednú redukciu diagramu. Riešením tohto problému sa zaoberá široká škála prác, z ktorých časť je opísaná v skorších kapitolách. Významný vplyv má však aj zvolená dekompozícia vstupnej funkcie. Použitie iba Shannonovej dekompozície je *najjednoduchším* spôsobom optimalizácie. Porovnanie jednotlivých dekompozícií použitých samostatne na vybrané obvody spolu s RV bolo uverejnené v [52]. Priemerná redukcia pre Shannonovu dekompozíciu bola 84,70%, pre pozitívne Davio 73,59% a pre negatívne Davio 84,08%. Samostatne dosahuje Shannon lepšie výsledky v porovnaní s Davio dekompozíciami, čo môže byť ďalším dôvodom (okrem jednoduchosti použitia) pre jeho rozšírenú popularitu. Najlepšie výsledky však boli dosiahnuté použitím kombinácie všetkých troch dekompozícií spolu, miera redukcie dosiahla zlepšenie o 87,24% oproti neredukovanému diagramu. Poradia dekompozícií pre premenné boli vygenerované použitím EA tak, ako je to opísané v kapitole 4.

Vzhľadom na stav publikovaných prác, kde väčšinu predstavuje optimalizácia BDD a KFDD je spomenuté len teoreticky, má význam porovnávať práve tieto dva typy diagramov. KFDD v sebe zahŕňa kombináciu všetkých typov dekompozícií. Z dostupných výsledkov sú pre KFDD známe [53], kde bolo použité posúvanie premenných a porovnanie OKFDD proti OBDD alebo relatívne nová práca [44], kde boli porovnané 4 rôzne algoritmy na hľadanie optimálneho DTL. Obe spomenuté práce žiaľ používajú iné testovacie obvody. Porovnanie BDD a KFDD spolu s RV (t.j. RViBDD a RViKFDD) je zobrazené v tabuľke 6.2.

Tabuľka 6.2 – Porovnanie prínosu RV v BDD a KFDD z pohľadu redukcie počtu uzlov

Testovací obvod	PSO	MMA	Sifting	BDD	RViBDD	RViKFDD	Zlepšenie [%]		
							RViBDD / BDD	RViKFDD / BDD	RViKFDD / RViBDD
cm150a	32	-	33	32	31	31	3,13	3,13	0,00
mux	32	-	33	32	31	31	3,13	3,13	0,00
cm151a	32	-	34	32	30	30	6,25	6,25	0,00
sao2	91	85	92	103	96	96	6,80	6,80	0,00
9sym	-	33	33	33	31	25	6,06	24,24	19,35
sqrt8	33	33	42	35	32	32	8,57	8,57	0,00
rd84	-	59	59	71	64	46	9,86	35,21	28,13
misex1	36	36	41	62	49	44	20,97	29,03	10,20
inc	79	61	68	96	81	81	15,63	15,63	0,00
5xp1	68	68	82	76	59	59	22,37	22,37	0,00
con1	16	15	18	15	12	11	20,00	26,67	8,33
squar5	37	37	38	47	34	31	27,66	34,04	8,82
rd53	-	23	23	29	24	18	17,24	37,93	25,00
Priemer	45,60	45,00	45,85	51,00	44,15	41,15	12,90	19,46	7,68

V tabuľke 6.2 sú uvedené výsledné počty uzlov pre redukované a usporiadané BDD, RViBDD a RViKFDD. Stĺpce *PSO* [12], *MMA* [13] a *Sifting* [21] zobrazujú zlepšenie dosiahnuté pri redukcii BDD, uvedené sú len pre porovnanie hodnôt s inými prácami (pri niektorých obvodoch boli dosiahnuté menšie hodnoty v iných prácach napriek tomu, že v tu prezentovanej práci boli preskúmané všetky možné poradia premenných, pravdepodobne pôjde o iný spôsob vyjadrovania veľkosti). Stĺpec *RViBDD* je identický s hodnotami uvedenými v práci [33]. Obsahovo je tento zdroj podmnožinou optimalizácií opísaných v tejto práci a rovnaké výsledky boli dosiahnuté otestovaním navrhnutej metodiky s použitím iba Shannonovej dekompozície.

V stĺpci *Zlepšenie* je uvedený percentuálny pomer zlepšenia pre použitie RV v BDD a taktiež percentuálne zlepšenie RViKFDD proti BDD aj RViBDD. Zatiaľ čo RV mala zjavný prínos v každom z vybraných obvodov, pridanie nových typov dekompozície prinieslo zlepšenie iba v 6 z 13 zobrazených obvodov. Priemerná pridaná hodnota RV použitej na BDD je na úrovni 12,9%. V práci [32] je uvedené priemerné zrýchlenie pri syntéze RViBDD oproti BDD 40,7%. V práci uvedenej v tomto dokumente nebol čas syntézy sledovanou veličinou vzhľadom na fakt, že výrazne záleží od implementácie a použitého hardvéru. V priemere však možno očakávať podobné zrýchlenie, pretože obe práce stoja na rovnakých teoretických základoch a rozdiel v použitom hardvéri by sa mal pomerovo prejaviť v oboch implementáciách. Čas syntézy a redukcie v súčasnosti prestáva byť až tak obmedzujúcim problémom, najmä vďaka jednoduchšej paralelizácii a dostupnosti výpočtových prostriedkov v klastrových platformách.

Zlepšenie v KFDD sa objavuje najmä pri funkciách s určitou symetriou, akou je napr. obvod 9sym. Priemerný prínos použitia všetkých dekompozičných pravidiel a RV je 19,46% pri porovnaní oproti redukovanému a usporiadanému BDD. Najväčšia pozorovaná zmena je pri obvode rd84 kde použitie oboch optimalizácií prinieslo zlepšenie až o 35,21%, z čoho až 25,35% je prínos kombinovania dekompozícií. Najmenšie pozorované hodnoty boli pri obvodoch s najväčším počtom vstupných premenných, kedy RV priniesla zmenšenie veľkosti iba o 1 uzol (o 2 v prípade cm151a) v porovnaní s BDD. Okrem týchto obvodov je najmenšia pridaná hodnota mierne vyššia ako 6%.

V obvodoch, v ktorých použitie Davio dekompozícií malo vplyv, je najnižším percentuálnym zlepšením RViKFDD oproti RViBDD hodnota 8,33%; v extrémnych prípadoch akým je napr. obvod rd84, je dosiahnuté zlepšenie až 28,13%. Z výsledkov je možné usúdiť, že keď už bude mať pridanie ďalších dekompozícií vplyv, je možné očakávať zlepšenie aspoň v okolí 16,64%. V nadpolovičnej väčšine však dominuje, resp. dosahuje najlepšie výsledky, Shannonova dekompozícia a preto je celkový priemer 7,68%. Z výsledkov uvedených v tabuľke jasne vyplýva pridaná hodnota použitia všetkých typov dekompozícií ako aj pridaná hodnota reziduálnej premennej. Určité ušetrenie výpočtového času by mohlo priniesť klasifikovanie obvodov pred redukciou na kontrolné a dátové funkcie a podľa rozdelenia určiť dekompozíciu. Pridaná hodnota RV je zjavná v oboch skupinách.

6.2 Spotreba a priemerná dĺžka výstupnej cesty v RViKFDD

V kapitole 4.3 je opísaná optimalizácia KFDD z pohľadu spotreby a APL reprezentovaného obvodu. Teoretická spotreba jednotlivých uzlov je vyjadrená ako súčet pravdepodobností výstupnej hodnoty 1, t.j. pravdepodobnosť prepnutia v uzle obvodu. Spotreba je určovaná rovnako pre všetky tri typy dekompozície, konkrétna spotreba záleží od zvolenej architektúry uzla, preto boli pri meraniach všetky uzly rovnocenné. APL diagramu je vypočítané podľa postupu opísaného v [42]. Koreňovému uzlu je priradená pravdepodobnosť výstupu 1 a každému ďalšiemu uzlu je postupne priradená polovičná pravdepodobnosť jeho rodiča. V prípade viacerých referencií na jeden uzol je jeho pravdepodobnosť vyjadrená ako súčet polovíc všetkých jeho rodičov. Výsledná hodnota APL diagramu je súčet všetkých pravdepodobností v diagrame.

Pre každý obvod bola sledovaná hlavne veľkosť vyjadrená počtom uzlov, teda pri EA funkcia vhodnosti prihliadala iba na výslednú veľkosť diagramu. Keďže pre niektoré obvody existuje viacero poradí premenných alebo dekompozícií dosahujúcich rovnaké hodnoty, boli uchovávané všetky výsledky s najnižším počtom uzlov (veľkosťou) a k nim prislúchajúce hodnoty spotreby a APL. Namerané hodnoty pre RViKFDD (redukované a usporiadané) vybraných obvodov sú zobrazené v tabuľke 6.3.

Tabuľka 6.3 – Spotreba a APL pre RViKFDD

Testovací obvod	Veľkosť	Spotreba			APL		
		Minimum	Maximum	Rozdiel [%]	Minimum	Maximum	Rozdiel [%]
cm150a	31	11,58	15	22,80	3,49	5	30,20
mux	31	11,44	11,44	0,00	3,49	3,49	0,00
t481	17	6,72	6,72	0,00	4,11	4,11	0,00
parity	15	0,83	0,83	0,00	2	2	0,00
cm151a	30	10,5	12,36	15,05	8	8	0,00
cm152a	11	4,38	4,84	9,50	3,5	3,5	0,00
sao2	96	29,79	29,79	0,00	11,48	11,48	0,00
9sym	25	7,83	7,83	0,00	6,89	6,89	0,00
sqrt8	32	10,97	12,82	14,43	11,13	11,63	4,30
rd84	46	10,23	12,94	20,94	14,09	19,11	26,27
misex1	44	14,26	17,08	16,51	18,66	20,72	9,94
inc	81	26,21	33,32	21,34	23,22	36,22	35,89
5xp1	59	27,82	29,5	5,69	21,66	23,34	7,20
con1	11	3,34	4,08	18,14	5,31	5,44	2,39
xor5	4	0,71	1,09	34,86	1,88	1,88	0,00
squar5	31	6,91	13	46,85	13,25	20,25	34,57
rd53	18	4,74	6,27	24,40	8,38	9,13	8,21
majority	6	1,87	2,34	20,09	2,38	3,13	23,96
Priemer	32,67	10,56	12,29	15,03	9,05	10,85	10,16

Stĺpce *Spotreba* aj *APL* sú rozdelené na 3 časti – minimum a maximum hodnôt z diagramov s najnižším počtom uzlov (viac riešení s rovnakou mierou redukcie) a ich rozdiel uvedený v percentách. Priemerný rozdiel spotreby pri meraných obvodoch je 15,03%. Táto hodnota výrazne poukazuje na dobre známu črtu DD - dosiahnutie najmenšieho počtu uzlov v diagrame nemusí znamenať dosiahnutie najlepších hodnôt aj v ostaných parametroch. Z 18 obvodov 5 dosiahlo rovnakú minimálnu a maximálnu spotrebu, čo môže byť vysvetlené dvomi javmi – bolo nájdené jediné riešenie s najnižším počtom uzlov (obvod t481) alebo všetky nájdené riešenia majú symetrické diagramy (9sym). Vynechaním obvodov s jediným najnižším riešením alebo obvodov so symetrickými riešeniami (t.j. obvodov s rozdielom spotreby rovným 0) sa dosiahne priemerná úspora spotreby 20,82%. Najväčší pozorovaný rozdiel sa objavil pri obvode squar5, kde bol rozdiel spotreby až 46,85%. Obe tieto hodnoty jasne poukazujú na potrebu optimalizácie DD s ohľadom na viacero parametrov naraz, jednoduchá redukcia počtu uzlov nie je vo väčšine prípadov postačujúca pri návrhu obvodov. Dokonca aj pri obvodoch s väčším počtom vstupov, akým je napr. aj cm150a, je rozdiel pozorovanej spotreby až 22,80%.

Pri APL je primerný rozdiel medzi najvyššími a najnižšími nameranými hodnotami 10,16%, zahŕňajúci všetky sledované obvody. 8 z 18 obvodov nepreukazuje zlepšenie APL pri rôznych usporiadaniach, odstránením obvodov s nulovým rozdielom sa priemerná zmena v APL zvyšuje na 18,29%. Najväčší, až 35,89-percentný rozdiel bol pozorovaný pri obvode inc. Táto hodnota naznačuje výskyt viacerých uzlov odkazujúcich sa na uzol alebo uzly s vysokou pravdepodobnosťou prechodu. Pri obvodovej realizácii by to znamenalo existenciu prvku, ktorý môže byť nepomerne viac vyťažený ako ostatné prvky v obvode, čím by sa zvýšila šanca jeho skorého opotrebovania alebo skratu.

Je nutné podotknúť, že najmenšia nameraná spotreba nemusí nutne znamenať aj najmenšie APL. Z uvedených obvodov mali iba obvody mux, t481, parity, sao2 a 9sym jednoznačné riešenie pri všetkých sledovaných parametroch. Pri ostatných obvodoch je potrebné priradiť váhu sekundárnym parametrom podľa riešeného problému a adekvátne upraviť funkciu vhodnosti.

6.3 Porovnanie BDD, KFDD a MVDD s reziduálnou premennou

Optimalizácia určovaním ekvivalentných uzlov v diagramoch je opísaná v kapitole 5. Navrhnutá heuristika spolu s novým typom diagramu boli testované ako vrstva optimalizácie pridaná nad RViKFDD. Ekvivalencia uzlov, resp. ich vstupných funkcií, sa zisťuje počas syntézy diagramu. Pri nájdení zhody je presmerovaná referencia rodiča na už existujúci uzol, ku ktorému je navyše poznačené preusporiadané poradie premenných. Jedným zo zaujímavých porovnaní by možno bolo aj porovnanie MVDD pri rôznych typoch dekompozícií, cieľom práce je však prekonávať najlepšie riešenia, nie zahltiť čitateľa množstvom výsledkov.

Namerané hodnoty zlepšenia redukcie DD pri použití MVDD sú zobrazené v tabuľke 6.4. V stĺpcoch *RViBDD, RViKFDD* a *RViMVDD* je uvedený počet uzlov redukovaného diagramu, v stĺpci (stĺpcoch) *Zlepšenie* je uvedené percentuálne zlepšenie redukcie.

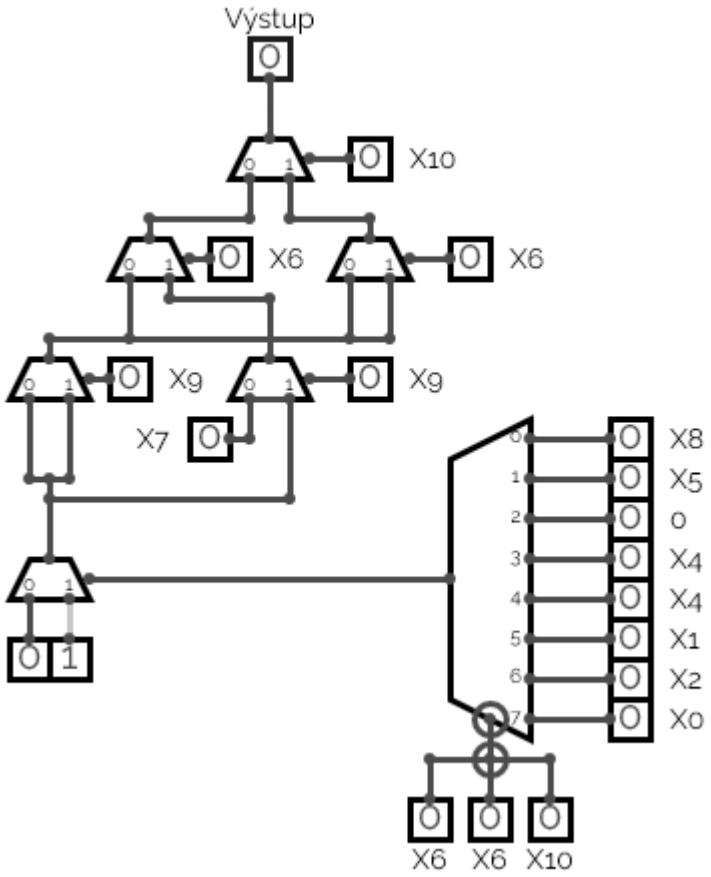
Tabuľka 6.4 – Porovnanie redukcií diagramov s RV pre BDD, KFDD a MVDD

Testovací obvod	RViBDD	RViKFDD	RViMVDD	Zlepšenie [%]	
				RViMVDD / RViBDD	RViMVDD / RViKFDD
cm150a	31	31	23	25,81	25,81
mux	31	31	22	29,03	29,03
t481	36	17	16	55,56	5,88
parity	29	15	15	48,28	0,00
cm151a	30	30	11	63,33	63,33
cm152a	14	11	6	57,14	45,45
sao2	96	96	94	2,08	2,08
9sym	31	25	25	19,35	0,00
sqrt8	32	32	30	6,25	6,25
rd84	64	46	46	28,13	0,00
misex1	49	44	36	26,53	18,18
inc	81	81	75	7,41	7,41
5xp1	59	59	56	5,08	5,08
con1	12	11	10	16,67	9,09
xor5	7	4	4	42,86	0,00
squar5	34	31	30	11,76	3,23
rd53	24	18	18	25,00	0,00
majority	6	6	6	0,00	0,00
Priemer	37	32,67	29,06	26,13	12,27

RViMVDD dosahuje výraznejšie zlepšenie ako samotné RViKFDD. V priemere je veľkosť MVDD oproti BDD (RV je implicitne daná) menšia o 26,13%, teda viac ako o štvrtinu, pričom sa dosiahlo zlepšenie u všetkých testovaných obvodoch okrem obvodu majority, ktoré má pri 5 vstupných premenných diagram so 6 uzlami. Toto zlepšenie v sebe zahŕňa pridanú hodnotu Davio dekompozícií aj zisťovania ekvivalencií uzlov. Priemerná pridaná hodnota samotnej ekvivalencie je 12,27% pre všetky obvody. Odstránením obvodov, pri ktorých MVDD neprinieslo zlepšenie (nulové hodnoty), narastá priemerná pridaná hodnota na 18,4%.

Z výsledkov vyplýva dôležitosť zisťovania ekvivalentných uzlov v diagrame a ich pomerne častý výskyt, avšak je potrebné zvážiť pomer ceny a dosiahnutých výsledkov. Pri obvodoch s nižším počtom vstupov, t.j. obvod sao2 a všetky obvody pod ním v tabuľke (do 10 vstupných premenných), sa javí dosiahnutie hraničných hodnôt redukcie. Optimalizácia MVDD priniesla až na jednu výnimku zlepšenie veľkosti diagramu menej ako 10%. Vo väčšine prípadov bol odstránené jeden až dva uzly z diagramu a pri takýchto *malých* zlepšeniach je otázne, či je vhodné zvyšovať zložitosť optimalizáci a aj štruktúry reprezentovaného obvodu za cenu tak nízkej redukcie. Naproti tomu pri obvodoch s väčším počtom vstupov (od cm152a vyššie, viac ako 10 premenných) bolo dosiahnuté priemerné zlepšenie redukcie o 33,9%, v priemere bola teda odstránená viac ako tretina už redukovaného obvodu.

Pri obvodoch cm150a a mux bol dosiahnutý počet uzlov redukovaného MVDD o dva a jeden uzol v uvedenom poradí väčší ako je počet vstupných premenných. Pri obvode t481 je počet uzlov redukovaného MVDD rovný počtu vstupov, pri obvodoch parity (rovnaké aj pri KFDD) a cm151a je veľkosť diagramu o jeden uzol menšia ako počet vstupov. Najzaujímavejším výsledkom je však obvod cm152a, pri ktorom má redukovaný KFDD rovnaký počet uzlov ako vstupov a redukovaný MVDD (oba DD s RV) má iba 6 uzlov pri 11 vstupných premenných. V skutočnosti hodnota troch premenných v tomto obvode rozhoduje o jednej z ôsmych premenných použitej na jediný uzol v diagrame. Multiplexorový strom pre MVDD je znázornený na obrázku 6.1, na ktorom vidno použitie šiestich 2:1 multiplexorov a jeden 8:1 multiplexor, ktorý rozhoduje o premennej v uzle na poslednej úrovni diagramu.



Obrázok 6.1 – RViMVDD pre obvod cm152a

Pri obvodoch cm150a a mux boli objavené poradia s najnižšou redukciou rovnaké pre BDD, KFDD aj MVDD. Ak sú problémom pamäťové a výpočtové nároky, je pravdepodobné dosiahnutie pozitívnych výsledkov aj vykonaním optimalizácie len pre BDD a aplikovaním MVDD iba na najlepšie nájdené riešenie (alebo skupinu riešení). Optimalizáciu je teda možné vykonať v troch krokoch – optimalizácia BDD, pridanie nových dekompozícií na najlepšie nájdené výsledky a nakoniec určenie ekvivalentných subdiagramov.

6.4 Spotreba a priemerná dĺžka výstupnej cesty v RViMVDD

Spotreba a APL pre MVDD boli odhadnuté rovnakým spôsobom ako pri KFDD. Výpočet APL zohľadňuje všetky referencie v diagramoch, teda aj tie, ktoré vznikli preusporiadaním premenných v jednom uzle a presmerovaním potomka rodičovského uzla na už existujúci ekvivalentný uzol. Namerané hodnoty sú zobrazené v tabuľke 6.5.

Tabuľka 6.5 – Spotreba a APL pre RViMVDD

Testovací obvod	Veľkosť	Spotreba			APL		
		Minimum	Maximum	Rozdiel [%]	Minimum	Maximum	Rozdiel [%]
cm150a	23	11	11	0,00	4,75	4,75	0,00
mux	22	8,07	8,07	0,00	3,28	3,28	0,00
t481	16	7,15	7,15	0,00	5,21	5,21	0,00
parity	15	6,83	7	2,43	2	2	0,00
cm151a	11	4,5	4,5	0,00	5,5	5,5	0,00
cm152a	6	2,02	2,02	0,00	7,13	7,13	0,00
sao2	94	29,79	29,79	0,00	11,48	11,48	0,00
9sym	25	6,89	6,89	0,00	10,78	11,59	6,99
sqrt8	30	8,93	12,11	26,26	10,38	10,97	5,38
rd84	46	16,67	19,39	14,03	14,09	19,11	26,27
misex1	36	11,11	13,97	20,47	16,44	17,5	6,06
inc	75	21,72	28,46	23,68	25,91	31,53	17,82
5xp1	56	27,82	29,5	5,69	21,66	23,34	7,20
con1	10	3,18	3,55	10,42	5,38	5,38	0,00
xor5	4	1,34	1,5	10,67	1,88	1,88	0,00
squar5	30	5,86	12,5	53,12	14,25	21,63	34,12
rd53	18	5,69	7,09	19,75	8,38	9,13	8,21
majority	6	1,38	2,5	44,80	2,38	3,13	23,96
Priemer	29,06	10,00	11,50	12,85	9,49	10,81	7,56

Rozdiel v priemernej spotrebe testovaných obvodov dosiahol hodnotu 12,85%. Najväčší pozorovaný rozdiel bol pri obvode squar5, až 53,12%. Pri nenulových rozdieloch je priemerný rozdiel v spotrebe 21,03%, takmer všetky tieto hodnoty sú však dosiahnuté pri menších obvodoch, kde môže aj malý počet uzlov spôsobiť veľký percentuálny rozdiel.

Priemerný rozdiel v APL pri všetkých obvodoch je 7,56%, kde bola opäť vačšina pozorovaných rozdielov pri menších obvodoch. Najväčší percentuálny rozdiel 34,12% preukazoval opäť obvod squar5. Vysoké rozdiely pozorované v spotrebe a APL, aj keď len pri relatívne malých obvodoch, poukazujú na dôležitosť optimalizácie sekundárnych parametrov. Obvod majority taktiež vykazuje vysoké rozdiely v spotrebe a APL iba pri 6 uzloch redukovaného diagramu. Väčšina obvodov s rovnakou maximálnou aj minimálnou hodnotou spotreby aj APL mala iba jediné najlepšie riešenie, aj v prípade symetrických funkcií.

6.5 Porovnanie BDD, KFDD a MVDD z pohľadu viacerých parametrov

V predchádzajúcich podkapitolách boli uvedené hodnoty spotreby a APL pre KFDD a MVDD s reziduálnou premennou. Práca [33] sa zaoberala optimalizáciou rovnakých parametrov pre RViBDD. Porovnanie hodnôt všetkých 3 typov diagramov sa nachádza v tabuľke 6.6. Stĺpce *Veľkosť RVi...* predstavujú najmenší dosiahnutý počet uzlov a stĺpce *Spot. min.*, resp. *APL min.*, označujú minimálnu nameranú spotrebu, resp. APL.

V poslednom riadku tabuľky je zobrazený postupne klesajúci trend minimálnych hodnôt. Zatiaľ čo priemerná minimálna spotreba pri RViBDD dosahovala hodnotu okolo 14,33, pri RViKFDD je to už 10,56 a pri RViMVDD dokonca rovných 10. Je nutné opäť podotknúť, že Davio uzly môžu mať mierne komplikovanejšiu štruktúru v závislosti od zvolenej architektúry, praktické implementácie preto môžu dosahovať iné pomery hodnôt ako prezentované teoretické hodnoty. Porovnanie APL je agnostické k rozdielom v architektúre členov obvodu. Priemerné namerané rozdiely sú však pri APL zanedbateľne malé, najväčší rozdiel 9,5% nastáva pri porovnaní priemerného APL pre RViBDD a RViKFDD. Väčšie priemerné APL pre RViMVDD môže byť spôsobené zvýšeným počtom referencií na uzly s veľkou pravdepodobnosťou prechodu.

Tabuľka 6.6 – Porovnanie minimálne spotreby a APL pre BDD, KFDD a MVDD

Testovací obvod	Vstup	Výstup	Veľkosť RVi...			RViBDD		RViKFDD		RViMVDD	
			BDD	KFDD	MVDD	Spot. min.	APL min.	Spot. min.	APL min.	Spot. min.	APL min.
cm150a	21	1	31	31	23	11,63	3,06	11,58	3,49	11	4,75
mux	21	1	31	31	22	15,38	3,06	11,44	3,49	8,07	3,28
t481	16	1	36	17	16	14,86	4,15	6,72	4,11	7,15	5,21
parity	16	1	29	15	15	14,5	15	0,83	2	6,83	2
cm151a	12	2	30	30	11	11,25	5,25	10,5	8	4,5	5,5
cm152a	11	1	14	11	6	7	3,25	4,38	3,5	2,02	7,13
sao2	10	4	96	96	94	23,52	10,33	29,79	11,48	29,79	11,48
9sym	9	1	31	25	25	10,22	7,13	7,83	6,89	6,89	10,78
sqrt8	8	4	32	32	30	13,43	9,94	10,97	11,13	8,93	10,38
rd84	8	4	64	46	46	24,52	22,36	10,23	14,09	16,67	14,09
misex1	7	7	49	44	36	18,91	16,75	14,26	18,66	11,11	16,44
inc	7	9	81	81	75	30,06	20,75	26,21	23,22	21,72	25,91
5xp1	7	10	59	59	56	27,82	21,66	27,82	21,66	27,82	21,66
con1	6	2	12	11	10	5,26	4,25	3,34	5,31	3,18	5,38
xor5	5	1	7	4	4	3,5	4	0,71	1,88	1,34	1,88
squar5	5	8	34	31	30	13,7	15,38	6,91	13,25	5,86	14,25
rd53	5	3	24	18	18	10,1	11,25	4,74	8,38	5,69	8,38
majority	5	1	6	6	6	2,35	2,38	1,87	2,38	1,38	2,38
Priemer	9,94	3,39	37,00	32,67	29,06	14,33	10,00	10,56	9,05	10,00	9,49

Z tabuľky 6.6 je možné usúdiť množstvo záverov. Existujú prípady, kedy použitie Davio dekompozícií prinieslo rovnakú veľkosť diagramu (obvodu) ale za cenu mierne vyššej spotreby a hodnoty APL, napr. cm150a alebo sao2. Naproti tomu v mux je pri rovnakej veľkosti obvodu spotreba nižšia o viac ako 25% a APL menšie o zhruba 14%, podobný jav sa opakuje aj v cm151a. V niektorých prípadoch bola pri zachovaní veľkosti dosiahnutá nižšia spotreba ale zvýšené APL, napr. inc alebo sqrt8. Jediný obvod so zachovaním veľkosti, APL a so znížením spotreby sa pri KFDD javí obvod majority. Obvody s redukciou veľkosti vo väčšine prípadov priniesli aj zmenšenie sekundárnych parametrov, napr. t481, 9sym, cm152a alebo rd84. Pri obvode parity bol dokonca rozdiel obrovský, hodnoty spotreby a APL klesli z 14,5 a 15 na hodnoty 0,83 a 2.

Pri porovnaní MVDD oproti KFDD sa vyskytli rovnaké javy ako pri porovnaní KFDD vs. BDD, objavilo sa zmenšenie jedného parametra za cenu zvýšenia druhého (cm150a, cm152a, 9sym), zmenšenie oboch parametrov (mux, cm151a, sqrt8) alebo dokonca zväčšenie oboch parametrov (t481), prípadne zmena iba jedného zo sledovaných parametrov k horšiemu (parity). Pri drvivej väčšine MVDD sa však dosiahla redukcia veľkosti. Podobné hodnoty APL pri KFDD a MVDD napovedajú, že uzol z diagramu síce zmizne ale jeho funkcionalita je zachovaná.

6.6 Zhodnotenie dosiahnutých výsledkov

V kapitole je uvedené porovnanie dvoch optimalizačných prístupov, ktoré ako podklad využívajú kombináciu všetkých možných poradí pri menej ako sedem vstupných premenných a evolučné algoritmy pri väčšom počte vstupov. Všetky opísané diagramy využívajú prínos RV, aj keď je explicitne uvedená. Jedine tabuľka 6.2 obsahuje v stĺpcoch *PSO*, *MMA*, *Sifting* a *BDD* diagramy bez použitej RV, ostatné tabuľky majú RV danú. Prvým prístupom je využívanie všetkých dekompozícií s dokázaným vplyvom na redukciu [5]. V diagramoch sa využíva kombinácia všetkých troch typov dekompozície s generovaním DTL pomocou EA tak, ako je to opísané v kapitole 4. Druhým predstaveným prístupom je DD s viacnásobnými premennými v uzloch označený ako MVDD, ktorý je formou voľného rozhodovacieho diagramu. Primárnym sledovaným parametrom bola veľkosť obvodu vyjadrená ako počet uzlov diagramu reprezentujúceho daný obvod. V 18 testovaných obvodoch prinieslo použitie všetkých typov dekompozície zlepšenie veľkosti v 10 obvodoch a MVDD v 17 obvodoch oproti najpoužívanejšiemu BDD. MVDD dosiahlo lepšie výsledky v 12 obvodoch oproti KFDD. Pri diagramoch boli zároveň s optimalizovaním veľkosti pozorované aj sekundárne parametre – spotreba a APL. Namerané hodnoty ukazujú, že so zvyšovaním redukcie dochádza k variabilným zmenám v sekundárnych parametroch, vo väčšine prípadov k miernemu zlepšeniu ale vyskytujú sa aj prípady, kedy došlo k zhoršeniu jedného alebo oboch parametrov. Rozdiel v prínose oboch typov diagramu je výraznejší pri porovnaní s RViBDD. Porovnanie KFDD a MVDD, resp MVKFDD, nie je až také výrazné, v priemere sa jedná o zlepšenie vo veľkosti a spotrebe na úrovni zhruba 11% a 5% a zhoršenie APL o 4,8%. Napriek tomu dosiahlo MVDD výraznejšiu mieru redukcie pri dvoch tretinách porovnávaných obvodov.

7 Záver

Optimalizácii rozhodovacích diagramov (DD) sa venuje množstvo prác. DD sú fundamentálnou dátovou štruktúrou v Booleovej algebre a našli široké uplatnenie vo viacerých oblastiach informatických vied. Ich pretrvávajúca popularita je podložená vedeckými výstupmi z minulosti ale aj z posledných rokov. Jednou z oblastí, ktorá najviac profituje z ich využívania, je návrh obvodov, ale s nemalým významom našli uplatnenie aj v sieťovej bezpečnosti, klasifikácii dát alebo v poslednej dobe vo veľmi populárnej oblasti umelej inteligencie.

Optimalizácia DD na teoretickej úrovni má za následok priamu alebo nepriamu optimalizáciu v reprezentovanom obvode. Väčšina dostupných prác sa orientuje hlavne na optimalizáciu veľkosti diagramu vyjadrenej počtom uzlov. Menší diagram spravidla znamená lacnejší a rýchlejší obvod, preto sa kladie dôraz hlavne na tento parameter. Okrem veľkosti je možné v obvode optimalizovať viacero iných parametrov, napríklad vedľajšími sledovanými parametrami môžu byť spotreba obvodu alebo jeho symetria, prípadne možnosti riadenia výstupnej cesty.

Hlavným z faktorov vplývajúcich na redukciu diagramu je usporiadanie vstupných premenných Booleovej funkcie. Usporiadanie premenných patrí medzi NP-problémy, preto je nutné využívanie algoritmov a heuristík na zjemnenie tohto problému. Medzi najpopulárnejšie metódy sa radia evolučné algoritmy (EA), ktoré dosahujú vynikajúce výsledky za zlomok času. Druhým faktorom v poradí najväčších vplyvov sa ukazuje použitie rôznych dekompozičných pravidiel na vstupnú funkciu. Existujú 3 dekompozície s dokázaným prínosom – Shannonova, pozitívne a negatívne Davio. Výber vhodnej dekompozície je námetom na samostatný výskum a v súčasnosti neexistuje jednoznačné pravidlo, ktoré by rozhodlo o použití dekompozície. Tretím faktorom sú známe postupy, ktoré majú menší ale stále zásadný vplyv na redukciu, napríklad reziduálne funkcie premennej definovanej v priamom aj komplementárnom tvare. Poslednou skupinou faktorov sú nové, zatiaľ neobjavené alebo neotestované optimalizačné postupy, často štandardom v inej oblasti ale ešte neaplikované na DD.

Hlavným cieľom práce bola optimalizácia DD z pohľadu viacerých parametrov na teoretickej úrovni. Primárnym parametrom všetkých optimalizačných postupov bola veľkosť diagramu. Sekudnárnymi sledovanými parametrami boli teoretické hodnoty spotreby a dĺžka výstupnej cesty v diagrame. Za týmto účelom boli navrhnuté nasledovné tézy práce:

- Vyladenie konfiguračných hodnôt EA pre vektor premenných aj vektor dekompozícií pri Kroneckerových funkcionálnych rozhodovacích diagramoch (KFDD).
- Overenie prínosu reziduálnej premennej v KFDD z pohľadu viacerých parametrov.
- Návrh novej heuristiky na vyhodnotenie ekvivalencie dvoch Booleových funkcií.

- Návrh nového typu diagramu s viacnásobnými premennými (MVDD) schopného redukovať ekvivalentné štruktúry v diagrame bez ohľadu na použité poradie.

V prvej časti práce je opísaný prínos v optimalizácii KFDD a všetky problémy s tým spojené, akými sú napríklad generovanie vektorov poradí alebo dekompozícií. Sú odvodené reziduálne funkcie pre všetky typy dekompozície, ktoré zhodou okolností poskytujú rovnaké výstupné hodnoty. Opísané sú aj rôzne postupy aplikovania reziduálnej premennej (RV) na diagram. KFDD s RV bol optimalizovaný pomocou EA. Pre dosiahnutie čo najvhodnejších vstupných nastavení EA bolo preskúmaných niekoľko intervalov hodnôt. Výsledné nastavenia, pri ktorých prestala s ich rastúcou hodnotou pribúdať ich pridaná hodnota, sú opísané v kapitole 4. V tejto kapitole sú odvodené aj zložitosti jednotlivých častí optimalizácie, napríklad pre n premenných je zložitosť usporiadania premenných rovná $n!$, ktorá platí rovnako pre BDD aj pre KFDD. Pri KFDD však pribúda problém zvolenia dekompozície, ktorý túto zložitosť násobí hodnotou 3!.

Pomerne často sa v diagramoch vyskytujú uzly, ktorých funkcie by boli po preusporiadaní premenných jednej z nich identické a diagramy by mali identickú štruktúru. Riešenie ekvivalencie B-funkcií je známy problém, ktorým sa zaoberá viacero prác. Cieľom práce však bolo vyvinúť novú metódu, preto bola táto časť výskumu zámerne minimalizovaná a práca sa sústreďuje na návrh originálnej metódy s ohľadom na DD. Ako druhý prínos práce je predstavená heuristika, ktorej princípom je predpoklad ekvivalencie funkcií a snaha o čo najrýchlejšie vyvrátenie tohto predpokladu. Inými slovami, heuristika očakáva, že funkcie sú ekvivalentné a niekoľkými krokmi, ktoré sú zoradené podľa rýchlosti ich vyhodnotenia, sa snaží dokázať nerovnosť funkcií po preusporiadaní. Prvé kroky heuristiky vyradia väčšinu funkcií z vyhodnocovania a posledné kroky sa snažia napárovať premenné oboch funkcií pomocou navrhnutej ohodnocovacej funkcie ε. Zložitosť navrhnutej heuristiky je odvodená ako pomer prehľadávanej množiny funckií oproti celkovej množine, pri dĺžke l a Hammingovej váhe h vektora hodnôt je zložitosť rovná $\binom{l-2}{h}/2^{l-2}$. Ohodnocovacia funkcia ε túto zložitosť teoreticky ešte viac znižuje, pre toto zníženie však nie je možné odvodiť všeobecný vzťah.

Pre maximálne využitie potenciálu predchádzajúcej heuristiky bol navrhnutý nový typ diagramu. Aby bolo možné reprezentovať dve rôzne funkcie s rôznymi poradiami jednou štruktúrou diagramu, je potrebné povolenie viacnásobného vstupu, resp. viacnásobných premenných v uzloch. Pre takýto krok neexistujú žiadne teoretické prekážky, je nutné len overiť kompatibilitu s najpoužívanejšími optimalizačnými metódami. V kapitole 5 je opísané použitie použitie základných redukčných pravidiel aj RV v novom type diagramu nazvanom rozhodovací diagram s viacnásobnými premennými - MVDD. Navrhnutý diagram nesie úplnú informáciu obsiahnutú vo vstupnej funkcii a po redukcii predstavuje kanonickú formu B-funkcie.

Pri všetkých vykonaných testoch boli sledované tri uvedené parametre obvodov – veľkosť, spotreba a priemerná dĺžka výstupnej cesty (APL). Na testovanie bola použitá voľne dostupná sada LGSynth93 Benchmark [20], ktorá je štandardom v oblasti. Pre oba opísané diagramy,

KFDD aj novonavrhnutý MVDD, boli odvodené funkcie na výpočet teoretickej spotreby obvodu pomocou prepínania aj APL. Namerané hodnoty boli porovnané s RViBDD opísanom v [33].

Vo všetkých diagramoch v kapitole 6 bola použitá RV pokiaľ nie je uvedené inak. Priemerné zlepšenie miery redukcie v upravenom KFDD oproti BDD bolo na úrovni 11,7%, pri novom MVDD oproti BDD to bolo až 21,46% a pri porovnaní MVDD oproti KFDD sa redukcia zlepšila o 11,05%. Z nameraných hodnôt je možné usúdiť, aj keď v iba veľmi voľnej interpretácii, že MVDD predstavuje zhruba rovnaký krok v optimalizácii veľkosti akým je použitie viacerých dekompozícií. Zlepšenie v odhadovanej spotrebe je pri oboch diagramoch porovnateľne veľké, s rozdielom medzi nimi iba niečo málo nad 5%, pri APL sa rozdiel medzi nimi pohybuje do 5%. Tento problém nie je ťažiskom práce, preto nebol ďalej riešený.

Z výsledkov uvedených v kapitole 6 je možné odvodiť dva najvýznamnejšie závery, na ktoré sa práca snažila poukázať. Prvým je fakt, že redukcia veľkosti diagramu nemá vždy iba pozitívny vplyv na sekundárne parametre diagramu. V niektorých prípadoch mala redukcia za následok zhoršenie spotreby (viacero potomkov na jednotkovej hrane uzla ako na nulovej), v iných zhoršenie priemernej výstupnej cesty (viaceré referencie na uzol vysokou pravdepodobnosťou prechodu) a dokonca sa objavili aj prípady, kedy boli zhoršené oba parametre naraz. Príklady spomenutých situácií sú uvedené v kapitole experimentálnych výsledkov.

Druhým záverom a hlavným prínosom práce je evidentná pridaná hodnota určovania ekvivalencie uzlov. Štandardné redukčné pravidlá typu I, S a D sa aplikujú iba v prípade, kedy sú identické vstupné funkcie a z nich vyplývajúca identická štruktúra diagramu. Ekvivalentné funkcie môžu taktiež dosiahnuť rovnakú štruktúru diagramu a preto je návrh diagramu využívajúceho túto skutočnosť iba prirodzeným rozšírením. Existujúce redukčné pravidlá sú schopné redukovať identické časti diagramu, MVDD je schopné redukovať aj ekvivalentné časti.

Práca môže byť zaujímavým podkladom pre ďalšie rozšírenie výskumu v oblasti a poukazuje na viacero existujúcich alebo nových problémov. Medzi existujúce problémy možno zaradiť sledovanie iných parametrov ako len spotrebu a APL, napríklad riadenie výstupnej cesty, rozloženie uzlov alebo rýchlosť vykonávania obvodu. Hodnoty spotreby by bolo najvhodnejšie porovnať pri konkrétnej architektúre prvkov, prípadne porovnať viacero architektúr medzi sebou. MVDD využíva uzly s viacnásobnými premennými, zaujímavou by mohla byť práca zaoberajúca sa uzlami podporujúcimi viacero dekompozícii naraz. Z čisto matematického hľadiska môže byť prínosom objavenie novej metódy dekompozície a jej vplyvu na optimalizáciu DD. Za zmienku stojí aj preskúmanie kompatibility MVDD s diagramami s potlačenou nulou alebo použitie negovaných hrán. Jedným z možných rozšírení MVDD je preskúmanie použitia redukčného pravidla S na označený uzol v MVDD, t.j. uzol, ktorého potomkovia majú rozdielne poradie premenných. Prácu je tiež možné rozšíriť o kompatibilitu s neúplne definovanými funkciami, pridaním efektívnejšej heuristiky na určenie ekvivalencie B-funkcií alebo objavením novej funkcie vhodnosti pre určovanie poradí dekompozičných pravidiel.

Zoznam použitej literatúry

[1] Bryant, R.: "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Transactions on Computers, vol. C-35, no. 8, pp. 677–691, 1986.

[2] Bryant, R.: "Binary decision diagrams and beyond: enabling technologies for formal verification", Proceedings of IEEE International Conference on Computer Aided Design (ICCAD), Nov. 1995.

[3] Pistek, P. New multiplexer-based switching circuits synthesis methods. Information Sciences and Technologies, 2015, 7.1/2: 19.

[4] Deb, Arighna, et al. "Synthesis of Optical Circuits Using Binary Decision Diagrams." Integration, the VLSI Journal, vol. 59, 11 May 2017, pp. 42–51.

[5] Wille, R., Niemann, P., Zulehner, A., Drechsler, R.: Decision diagrams for the design of reversible and quantum circuits. In 2018 International Symposium on Devices, Circuits and Systems, IEEE, 2018.

[6] Technology Roadmap for Semiconductors: Design. 2015: https://www.semiconductors.org/resources/2015-international-technology-roadmap-for-semiconductors-itrs/

[7] Nagy, B.: Optimal Boolean Programming with Graphs. Acta Polytechnica Hungarica, 2019, 16.4.

[8] Becker, B., and R. Drechsler. "How Many Decomposition Types Do We Need? [Decision Diagrams]." Proceedings the European Design and Test Conference. ED&TC 1995, 1995.

[9] Amarú, L., Gaillardon, P.E., De Micheli, G.: "Majority-Inverter Graph: A New Paradigm for Logic Optimization", Computer-Aided Design of Integrated Circuits and Systems IEEE Transactions on, vol. 35, pp. 806-819, 2016, ISSN 0278-0070.

[10] Amarú, L; Gaillardon, P.E.; De Micheli, G. Bds-Maj: A Bdd-Based Logic Synthesis Tool Exploiting Majority Logic Decomposition. In: Proceedings of the 50th Annual Design Automation Conference. Acm, 2013. P. 47.

[11] Bolling, B., Wegener, I.: Improving the variable ordering of OBDDs is NP-complete. In: IEEE Transactions on Computers, IEEE, 1996, vol. 45, no. 9, pp. 993-1002.

[12] Gerov, R., Jovanovic, Z.: Parameter Estimation Method for the Unstable Time Delay Process. Acta Polytechnica Hungarica, 2019, 16.3.

[13] P. Pistek, M. Kolesár, and K. Jelemenská. Optimization of multiplexer trees using modified truth table. In International Conference on Applied Electronics, pages 265–268. IEEE, 2010.

[14] Ebendt, R., Fey, G., Drechsler, R.: Advanced BDD Optimization (1. ed.). Netherlands: Springer, 2005, 222p, ISBN 978-0-387-25453-1.

[15] Rice, M., Kulhari, S.: A Survey of Static Variable Ordering Heuristics for Efficient BDD/MDD Construction. University of California, Technical Report, 2008.

[16] Friedman, S. J., Supowit, K.J.: Finding the optimal variable ordering for binary decision diagrams. In: IEEE Transactions on Computers. IEEE, 1990, vol. 39, no. 5, pp. 710-713.

[17] A. Mitra and S. Chattopadhyay. Variable ordering for shared binary decision diagrams targeting node count and path length optimisation using particle swarm technique. IET Computers & Digital Techniques, 6(6):353–361, 2012.

[18] Rehan, S., Bansal, M.: Performance Comparison among Different Evolutionary Algorithms in terms of Node Count Reduction in BDDs. In: International Journal of VLSI and Embedded Systems – IJVES, Technical Journals, 2013, vol. 4, no. July 2013, pp. 491-496.

[19] Marculescu, R., Marculescu, D., Pedram, P.: . Efficient power estimation for highly correlated input streams. In: Design Automation Conf., June 1995.

[20] McElvain, K.: LGSynth93 Benchmark Set: Version 4.0. http://www.cbl.ncsu.edu:16080/benchmarks/LGSynth93/.

[21] Rudell, R.: Dynamic variable ordering for ordered binary decision diagrams. In Int'l Conf. on CAD, pages 42–47. IEEE, 1993.

[22] Bern, J., Gregov, J., Meinel, C., Slobodova, A.: Boolean Manipulation With Free Bdd's.First Experiment a1 Results., In: FB IV – Informatik, Universität Trier, IEEE 1994

[23] A. U. Hassen, S. A. Khokhar and B. Amin, "Synthesis of Compact Crossbars for in-Memory Computing using Dynamic FBDDs," *2018 IEEE 18th International Conference on Nanotechnology (IEEE-NANO)*, 2018, pp. 1-4, doi: 10.1109/NANO.2018.8626401.

[24] Kisun, K., Taekyoon, A., Sang-Yeoul, H., Chang-Seung, K., Ki-Hyun, K. : Low-Power Multiplexer Decomposition by Suppresing Propagation of Signal Transitions. In: *The 2001 IEEE International Symposium on Circuits and Systems - ISCAS 2001*. Sydney, Australia: IEEE, 2001, vol. 5, pp. 85-88.

[25] Drechsler, R., Theobald, M., Becker, B. : Fast OFDD-Based Minimization of Fixed Polarity Reed-Muller Expressions. In: *IEEE Transactions on Computers*. IEEE, 1996, vol. 45, no. 11, pp. 1294-1299.

[26] Ebendt, R. Drechsler, R.: Exact minimisation of path-related objective functions for binary decision diagrams. In: *IEEE Proceedings - Computers and Digital Techniques*. IEEE, 2006, vol. 153, no. 4, pp. 231-242.

[27] Brudaru, O., Ebendt, R. , Furdu, I.: Optimizing variable ordering of BDDs with double hybridized embryonic genetic algorithm, In: *Proc. of The 12th Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing - Synasc 2010*, pp. 167-173.1

[28] Jacobi, R.P., Trullemans, A.M. : Generating prime and irredundant covers for binary decision diagrams. In: *3rd European Conference on Design Automation*. Brussels, Belgium, IEEE, 1992, pp. 104-108.

[29] Mishchenko, A. : An introduction to zero-suppressed binary decision diagrams. Technical report, 2001. [Online: August 2021]. Dostupné na: Alan Mishchenko – publications, Portland State University: https://people.eecs.berkeley.edu/~alanmi/publications/2001/tech01_zdd_.pdf

[30] Drechsler, R., Höreth, S. : Manipulation of* BMDs. In: *Asia and South Pacific Design Automation Conference – ASP-DAC 1998*. Yokohama, Japan, IEEE, 1998, pp. 433-438.

[31] T. Sasao: "Ternary decision diagrams. Survey," *Proceedings 1997 27th International Symposium on Multiple- Valued Logic*, 1997, pp. 241-250, doi: 10.1109/ISMVL.1997.601404.

[32] M. Maruniak and P. Pištek, "Binary decision diagram optimization method based on multiplexer reduction methods," *2013 International Conference on System Science and Engineering (ICSSE)*, 2013, pp. 395-399, doi: 10.1109/ICSSE.2013.6614698.

[33] Maruniak, M., Pištek, P.: A New Multiplexer-based Multi-objective Digital Circuit Synthesis Method. In: *EUROMICRO DSD/SEAA 2014*, 27. - 29. August 2014, Verona, Italy, IEEE..

[34] Hamming, R.: The Bell System Technical Journal, In: *Error Detecting and Error Correcting Codes*, Bell System Technical Journal 1950

[35] Lúčanský J.: Optimalizácie multiplexorových stromov. Bratislava: FIIT STU, 2015, Diplomová práca, Vedúci: Pištek, P., pp. 82.

[36] Chaudhury, S., Dutta, A.: Algorithmic Optimization of BDDs and Performance Evaluation for Multi-level Logic Circuits with Area and Power Trade-offs, *Circuits and Systems*, 2011, vol. 2, no 3, pp. 217-224.

[37] Dua, T., Rajput, A.: 2:1 Multiplexer Using Different Design Styles: Comparative Analysis. JoARB (2020) STM Journals 2020, pp. 5-13.

[38] M. G. Karpovsky, R. S. Stankovic and J. T. Astola, "Construction of linearly transformed planar BDD by Walsh coefficients," *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, 2004, pp. IV-517, doi: 10.1109/ISCAS.2004.1329054.

[39] P. W. C. Prasad, M. Raseen, A. Assi, A. Elchouemi and S. M. N. A. Senanayake, "Minimum average path length in BDDs based on static variable ordering," *48th Midwest Symposium on Circuits and Systems, 2005.*, 2005, pp. 716-719 Vol. 1, doi: 10.1109/MWSCAS.2005.1594201.

[40] R. Ebendt, W. Gunther and R. Drechsler, "Minimization of the expected path length in BDDs based on local changes," *ASP-DAC 2004: Asia and South Pacific Design Automation Conference 2004 (IEEE Cat. No.04EX753)*, 2004, pp. 866-871, doi: 10.1109/ASPDAC.2004.1337716.

[41] Keren, O.: Reduction of Average Path Length in Binary Decision Diagrams by Spectral Methods, Computers, IEEE Transactions, 2008, vol.57, no.4, pp.520-531.

[42] Nagayama, S., Mishchenko, A., Sasao, T., Butler, J., T.: Minimization of Average Path Length in BDDs by Variable Reordering, In: Proc. 12th Int'l Workshop Logic and Synthesis, 2003, pp. 1-8.

[43] J. T. Butler, T. Sasao and M. Matsuura, "Average path length of binary decision diagrams," in *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1041-1053, Sept. 2005, doi: 10.1109/TC.2005.137.

[44] X. Huang *et al.*, "Dynamic Minimization of Bi-Kronecker Functional Decision Diagrams," *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1-9.

[45] M. Fujita, H. Fujisawa and N. Kawato, "Evaluation and improvement of Boolean comparison method based on binary decision diagrams," *[1988] IEEE International Conference on Computer-Aided Design (ICCAD-89) Digest of Technical Papers*, 1988, pp. 2-5, doi: 10.1109/ICCAD.1988.122450.

[46] A. Petkovska, M. Soeken, G. De Micheli, P. Ienne and A. Mishchenko, "Fast hierarchical NPN classification," *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1-4, doi: 10.1109/FPL.2016.7577306.

[47] Z. Huang, L. Wang, Y. Nasikovskiy and A. Mishchenko, "Fast Boolean matching based on NPN classification," *2013 International Conference on Field-Programmable Technology (FPT)*, 2013, pp. 310-313, doi: 10.1109/FPT.2013.6718374.

[48] M. M. Doijade and D. B. Kulkarni, "Overview of sequential and parallel SAT solvers," *International Conference on Information Communication and Embedded Systems (ICICES2014)*, 2014, pp. 1-4, doi: 10.1109/ICICES.2014.7033875.

[49] D. G. Corneil and C. C. Gotlieb: An Efficient Algorithm for Graph Isomorphism. J. ACM 17, 1 (Jan. 1970), 51–64. DOI:https://doi.org/10.1145/321556.321562

[50] W. V. Parada, F. A. Giraldo and M. I. Londoño R, "MDD based case tool for Automatic Generation of ChatBot," *2019 XLV Latin American Computing Conference (CLEI)*, 2019, pp. 1-7, doi: 10.1109/CLEI47609.2019.235059.

[51] Changhee Kim, L. Lavagno and A. Sangiovanni-Vincentelli, "Free MDD-based software optimization techniques for embedded systems," *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*, 2000, pp. 14-18, doi: 10.1109/DATE.2000.840009.

[52] Lucansky, J., Pistek, P., & Maruniak, M.: The residual variable in decision diagrams. *Acta Polytechnica Hungarica*, *17*(5), 189–208, 2020, doi: 10.12700/aph.17.5.2020.5.10

[53] R. Drechsler and B. Becker, "Ordered Kronecker functional decision diagrams-a data structure for representation and manipulation of Boolean functions," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 10, pp. 965-973, Oct. 1998, doi: 10.1109/43.728917.

[54] Knuth, D. (2005). *The art of computer programming, volume 4, fascicle 1 : Bitwise tricks & techniques ; Binary decision diagrams*.

Príloha A – Publikácie autora

Publikácie medzinárodnej úrovne s medzinárodným rozpoznaním (A)

[1] Lúčanský, Ján - Pištek, Peter - Maruniak, Marián. The Residual Variable in Decision Diagrams. In Acta Polytechnica Hungarica. Vol. 17, no. 5 (2020), s. 189-208. ISSN 1785-8860 (2020: 1.806 - IF, Q3 - JCR Best Q, 0.277 - SJR, Q2 - SJR Best Q). V databáze: WOS: 000532417700010 ; SCOPUS: 2-s2.0-85086874902.

Publikácie medzinárodnej úrovne (B)

[1] Kristel, Patrik - Lúčanský, Ján - Kotuliak, Ivan. Traffic Optimization in Anonymous Network. In CNSM 2017. Proceedings of 13th International Conference on Network and Service Management, Tokyo, Japan, Nov. 26 - Nov. 30, 2017. 1. vyd. Piscataway : IEEE, 2017, S. 1-5. ISBN 978-3-901-882-98-2. V databáze: WOS: 000427961400008 ; SCOPUS: 2-s2.0-85046680522.

Publikácie na lokálnych fórach a študentských konferenciách (D)

[1] Lúčanský, Ján. Optimization of decision diagrams. In *IIT.SRC 2015, Student Research Conference: 11th Student Research Conference in Informatics and Information Technologies, Bratislava, April 23, 2015*. 1. vyd. Bratislava : Nakladateľstvo STU, 2015, S. 333-338. ISBN 978-80-227-4342-6..