Desktopová aplikace k minimalizaci logických funkcí

Daniel Ševčík

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Daniel Ševčík**
Osobní číslo:	**A16070**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Desktopová aplikace k minimalizaci logických funkcí**
Téma anglicky:	**A Desktop Application for Boolean Function Simplification**

Zásady pro vypracování:

1. **Seznamte se s programovacím jazykem C++ a knihovnami použitými v této práci.**
2. **Osvojte si pravidla Booleovy algebry a minimalizace logických funkcí pomocí Karnaughových map. V teoretické části práce je popište.**
3. **Navrhněte a naprogramujte algoritmus pro řešení Karnaughových map s podporou až osmi proměnných na vstupu.**
4. **V programu vytvořte přehledné uživatelské rozhraní s podporou automatické i manuální minimalizace a možností následného exportu výsledků do obecně používaných souborových formátů.**
5. **Otestujte vytvořený program na několika praktických příkladech a vyhodnoťte výsledky.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **HOLDSWORTH, Brian a Clive WOODS. Digital Logic Design. 4th ed. Boston: Newnes, 2002. ISBN 978-075-0645-829.**
2. **TIDWELL, Jenifer. Designing Interfaces. 2nd ed. Sebastopol (California): O'Reilly Media, 2010. ISBN 978-144-9379-704.**
3. **PRATA, Stephen. Mistrovství v C++. 4. aktualiz. vyd. Brno: Computer Press, 2013. ISBN 978-802-5138-281.**
4. **KUPHALDT, Tony R. Vol. IV – Digital – Electronics Textbook. All About Circuits [online]. [cit. 2018-11-26]. Dostupné z: https://www.allaboutcircuits.com/textbook/digital/**
5. **CORNUT, Omar. Dear ImGui: Bloat-free Immediate Mode Graphical User interface for C++ with minimal dependencies. GitHub [online]. [cit. 2018-11-26]. Dostupné z: https://www.github.com/ocornut/imgui**

Vedoucí bakalářské práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **21. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 21. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 15. 5. 2019

Daniel Ševčík, v. r.

.....

podpis diplomanta

ABSTRAKT

Předmětem této práce je návrh a tvorba desktopové aplikace k minimalizaci logických funkcí metodou Karnaughových map. Součástí aplikace je přehledné uživatelské rozhraní, které uživatele provede všemi dílčími kroky minimalizačního procesu. Program umožňuje provádět manuální i zcela automatickou tvorbu řešení s generováním výsledků v libovolném textovém formátu, včetně různých programovacích jazyků. Veškeré výsledky je možné ukládat do řady běžně používaných grafických a textových formátů. Aplikace cílí na využití v praxi i ve výuce či tvorbě podpůrných materiálů k ní určených.

Klíčová slova: minimalizace, logická funkce, Booleova algebra, Karnaughova mapa, grafické uživatelské rozhraní

ABSTRACT

The main goal of this thesis is to design and create a desktop application for Boolean function minimization using the Karnaugh map method. The application includes a graphical user interface which guides its user through the process of minimization. User can solve Karnaugh maps on his own or use the solver. Minimization results can be generated in arbitrary text format, including various programming languages. It is possible to save the results to common image and text file formats. The application aims for its practical use in various industries including education.

Keywords: minimization, Boolean function, Boolean algebra, Karnaugh map, graphical user interface

Rád bych poděkoval vedoucímu své bakalářské práce, panu Ing. Pavlu Pokornému, Ph.D, za jeho cenné rady a asistenci při jejím vypracovávání. Dále také Mgr. Barboře Jestřebské a celé své rodině za jejich podporu nejen v průběhu vypracovávání této práce, ale především po celou dobu mého studia.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Minimalizace logických funkcí nachází své každodenní využití v celé řadě technických oborů. Díky ní je možné dosáhnout jednoduššího technologického procesu i vyšší spolehlivosti logických systémů. [3]

Předmětem této bakalářské práce je návrh a následná realizace desktopové aplikace k minimalizaci logických funkcí. Práce se zabývá problematikou tvorby algoritmu pro minimalizaci metodou Karnaughových map s podporou až osmi proměnných na vstupu, který je nedílnou součástí vyvíjené aplikace. Krom využití plně automatického režimu minimalizace nabízí i možnost provést celou minimalizaci ručně.

Aplikace cílí na praktické využití v oblasti návrhu logických funkcí a obvodů, optimalizace zdrojového kódu a v neposlední řadě také na využití v akademické sféře pro podporu výuky problematiky minimalizace logických funkcí a tvorby podpůrných materiálů k výuce určených. K tomu slouží schopnost aplikace provádět export do řady grafických i textových souborových formátů včetně v praxi používaných programovacích jazyků.

Mou hlavní motivací pro volbu tohoto tématu byl nedostatek kvalitních nástrojů pro tvorbu a následnou minimalizaci logických funkcí umožňujících provádět grafický návrh a export dílčích prvků. Většina existujících aplikací se soustředí na jedinou činnost – automatickou minimalizaci.

I. TEORETICKÁ ČÁST

1 MINIMALIZACE LOGICKÝCH FUNKCÍ

Jak již bylo zmíněno v úvodu, minimalizace logických funkcí má své využití v celé řadě technických oborů. Její princip spočívá v hledání ekvivalentního funkčního tvaru v nejjednodušší možné podobě. [3]

V této kapitole budou představeny nejzákladnější pojmy související s minimalizací logických funkcí včetně dvojice prostředků určených k jejímu vykonávání - Booleovy algebry a Karnaughových map. V obou případech je důraz kladen primárně na vysvětlení dané problematiky prostřednictvím praktických příkladů.

1.1 Základní pojmy

Před nastíněním jednotlivých metod minimalizace je nutné definovat základní pojmy, které s minimalizací úzce souvisí. Jedním z nich je logická funkce. Tu lze popsat jako funkci sestávající z určitého počtu skupin logických proměnných spojených logickými operátory. V závislosti na použitých operátorech se jedná o funkci v součtovém nebo součinovém tvaru. Tyto tvary jsou duální, což znamená, že jedna forma po záměně logických operátorů a následné inverzi hodnot proměnných vyplývá z té druhé. Pokud všem kombinacím vstupních proměnných odpovídá výstupní logická hodnota, nazývá se funkce úplně zadanou. V opačném případě hovoříme o neúplně zadané funkci, jejímž nedefinovaným výstupním hodnotám se říká neurčitý stav. Ty bývají označovány znakem X. [1; 3]

Základní metodu strukturovaného popisu logické funkce představuje použití pravdivostní tabulky. Ta obsahuje řádek pro každou kombinaci proměnných a předepisuje jim funkční hodnotu. Počet těchto kombinací a potažmo i řádků pravdivostní tabulky odpovídá hodnotě 2^n, kde n je počet vstupních proměnných. Logická funkce je dána sumou kombinací proměnných vedoucích k funkční hodnotě jedna. Každá takováto kombinace se nazývá mintermem. Logická funkce daná sumou mintermů se nazývá funkcí v kanonickém součtovém tvaru. Opakem je kanonický součinový tvar funkce, který je daný součinem takzvaných maxtermů, což jsou kombinace vstupních proměnných, při nichž je výstup funkce roven nule. [2]

1.2 Booleova algebra

Anglický matematik George Boole (1815 – 1864) se zabýval hledáním způsobu, jakým by bylo možné přiřadit Aristotelově logice symbolickou formu. Boole v roce 1854 vydal odbornou publikaci s názvem *An Investigation of the Laws of Thought on Which are*

Founded the Mathematical Theories of Logic and Probabilities. Jejím prostřednictvím popsal vztahy mezi matematickými veličinami, jež mohou nabývat vždy jen jedné ze dvou přípustných hodnot, jedničky nebo nuly, které signalizují existenci či absenci určité podmínky. Tento matematický systém je známý pod názvem Booleova algebra. [1; 2]

1.2.1 Základní pravidla Booleovy algebry

Booleova algebra je, stejně jako každý jiný matematický systém, definována sadou základních definic. Tu tvoří tři elementární operace, které daly za vznik řadě doplňujících zákonů i ostatních operací. Jedná se o negaci, konjunkci (logický součin) a disjunkci (logický součet). V případě první jmenované platí, že negace proměnné je rovna jedné pouze tehdy, pokud je proměnná rovna nule a naopak. Chování prvků konjunkce a negace je znázorněno prostřednictvím pravdivostní tabulky (*Tabulka 1.1*). [1; 2]

Tabulka 1.1: Pravdivostní tabulka konjunkce a disjunkce dvou proměnných.

A	**B**	**A · B**	**A + B**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Předmětem práce není provádět minimalizaci pomocí Booleovy algebry, nýbrž metodou Karnaughových map. Z toho důvodu jsou do kapitoly zahrnuta pouze základní pravidla a operátory, které se v projektu mohou vyskytnout.

Na základě zmíněných operací byla odvozena řada zákonů, pomocí nichž je možné logické funkce upravovat – minimalizovat. V závislosti na tvaru funkce je nutné aplikovat odpovídající variantu zákonů, jejichž přehled je k vidění v tabulce (*Tabulka 1.2*). [3]

Tabulka 1.2: Definice zákonů Booleovy algebry. [18]

	Součtová forma	**Součinová forma**
Zákon komutativní	$A + B = B + A$	$A \cdot B = B \cdot A$
Zákon asociativní	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
Zákon distributivní	$A \cdot B + A \cdot C = A \cdot (B + C)$	$(A + B) \cdot (A + C) = A + (B \cdot C)$
Zákon idempotence	$A + A = A$	$A \cdot A = A$
Zákon vyloučeného třetího	$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$
Zákon agresivních hodnot	$A + 1 = 1$	$A \cdot 0 = 0$
Zákon neutrálních hodnot	$A + 0 = A$	$A \cdot 1 = A$
Zákon adsorpce	$A + A \cdot B = A$	$A \cdot (A + B) = A$
Zákon adsorpce negace	$A + \bar{A} \cdot B = A + B$	$A \cdot (\bar{A} + B) = A \cdot B$
Zákon dvojí negace	$\bar{\bar{A}} = A$	$\bar{\bar{A}} = A$
De Morganovy zákony	$\overline{A + B} = \bar{A} \cdot \bar{B}$	$\overline{A \cdot B} = \bar{A} + \bar{B}$

1.2.2 Příklad minimalizace pomocí Booleovy algebry

Princip minimalizace logických funkcí je vhodné nastínit na názorném příkladu. Základní funkční tvar byl převzat ze zdroje [18] a upraven dle potřeby, viz rovnice (1.1).

$$y = A\bar{B} + A\bar{C} + A\bar{B}D + ABCD \tag{1.1}$$

Jelikož se v rámci celého výrazu nenabízí přímá možnost úpravy jeho dílčích prvků, je nutné vybrat nejvhodnější kandidáty k vytknutí. Vybírají se prvky, na které bude možné aplikovat některý ze zákonů Booleovy algebry. Provede se vytknutí znázorněné rovnicí (1.2).

$$y = A\bar{B}(1 + D) + A(\bar{C} + BCD) \tag{1.2}$$

Vhodnost volby prvku $A\bar{B}$ spočívá v tom, že po vytknutí zůstane v závorkách logická jednička. Tam lze uplatnit zákon agresivních hodnot a celou závorku vyřadit. V případě zbylých hodnot se nabízí vytknutí prvku A, které vede k výskytu osamocené proměnné C v negované podobě a její normální formy uvnitř přidruženého podvýrazu BCD, kdy je možné aplikovat zákon adsorpce negace. Výsledek dosavadní činnosti lze vidět v rovnici (1.3).

$$y = A\bar{B} + A\bar{C} + ABD = A(\bar{B} + \bar{C} + BD) \tag{1.3}$$

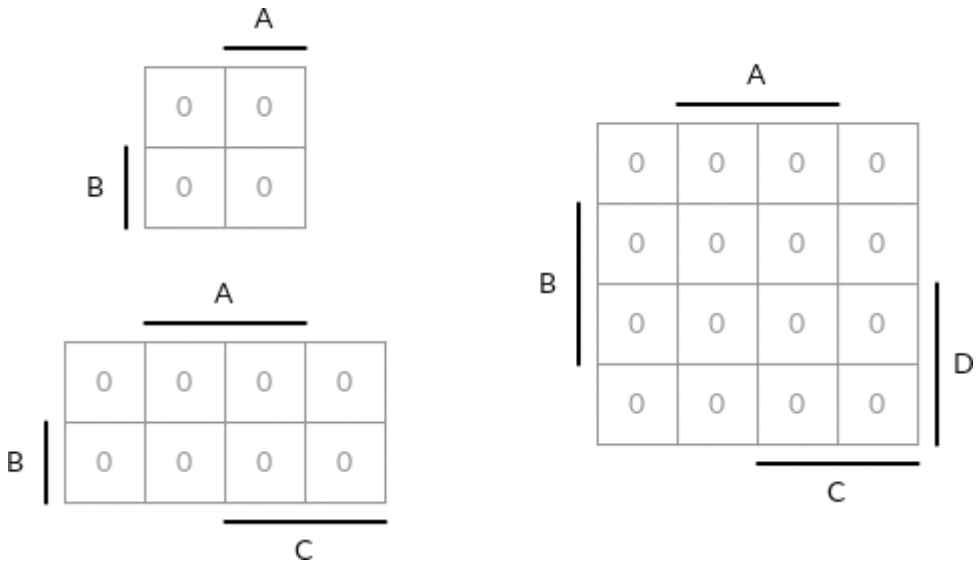
Ve tvaru rovnice (1.3) je možné si všimnout dvojice prvků $\bar{B}$ a BD, na něž jde stejně jako v případě minulého kroku aplikovat zákon adsorpce negace, čímž se výsledek ještě více upraví. Výsledkem je již dále neupravitelný a tím pádem plně zminimalizovaný tvar funkce y, viz rovnice (1.4).

$$y = A(\bar{B} + \bar{C} + D) \tag{1.4}$$

1.3 Karnaughovy mapy

S návrhem Karnaughových map přišel v roce 1953 americký telekomunikační inženýr Maurice Karnaugh při návrhu logiky spínacích obvodů v Bellových laboratořích. Karnaughovy mapy lze chápat jako nástroj pro grafické vyjádření logických funkcí. Nejvýznamnější je však jejich využití pro minimalizaci. To je umožněno rozložením proměnných do řádků a sloupců na základě Grayova kódu, kdy se sousední pole liší v hodnotě vždy jen jedné proměnné. Na základě toho je možné pojit sousední pole do takzvaných minimalizačních smyček, jejichž minterm lze vyčíst jako součin hodnot proměnných, které se napříč touto smyčkou nemění.

Proměnné a jejich hodnoty mohou být označeny několika způsoby, jedním z nich je označení pomocí čar. Čára s názvem proměnné pokrývá buňky, v nichž tato proměnná nabývá logické jedničky. Trojice různých podob Karnaughových map v závislosti na počtu vstupních proměnných je k vidění na obrázku (Obrázek 1.1). Ve většině případů platí, že je minimalizace jejich využitím snazší a rychlejší, než aplikace pravidel Booleovy algebry. To platí prakticky vždy při výskytu více než dvou vstupních proměnných. [1; 2; 3]



Obrázek 1.1: Příklad Karnaughových map.

V případě výskytu více než čtyř vstupních proměnných nastává situace, kdy je nutné Karnaughovu mapu dále rozdělit, aby bylo možné zahrnout i zbytek proměnných. V případě pěti proměnných je možné každou buňku diagonálně rozdělit na dvě, v případě šesti proměnných na čtyři a tak dále. Z hlediska algoritmizace se však nejedná o praktický postup a je proto výhodnější využít alternativy v podobě vytvoření $2^{(n-4)}$ dílčích map, kde *n* označuje počet vstupních proměnných.

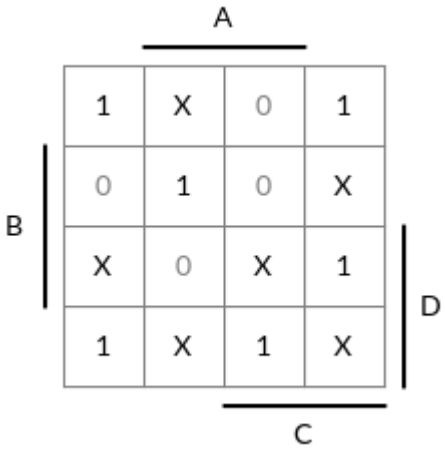
1.3.1 Základní pravidla Karnaughových map

Při provádění samotné minimalizace logických funkcí pomocí Karnaughových map je vhodné dodržovat následující pravidla, na jejichž základě celý proces staví:

- Všechny logické jedničky uvnitř mapy musí být zahrnuty v minimalizační smyčkách, přičemž se uvnitř nesmí nacházet byť jen jediná logická nula. V případě hledání inverzní funkce platí pravý opak - je nutné zahrnout všechny logické nuly a žádné logické jedničky. [1]
- Dohromady lze pojit pouze sousedící buňky. Těmi se rozumí i buňky nacházející se na protilehlých stranách mapy.
- Velikost každé minimalizační smyčky musí být rovna hodnotě 1, 2, 4, 8 či vyšší mocnině dvou.
- Řešením je vyhledání nejnižšího možného počtu co největších smyček. K tomu se využívá i schopnosti jejich vzájemného překrývání.
- Je možné zahrnout neurčité stavy zcela dle potřeby. To však musí vést ke zvětšení rozměrů smyčky.

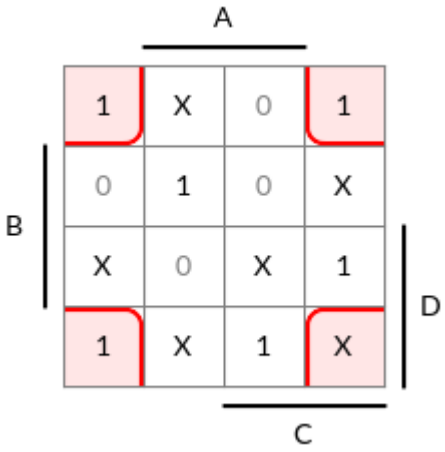
1.3.2 Příklady minimalizace pomocí Karnaughových map

Pro znázornění průběhu minimalizace byla zvolena dvojice náhodně vygenerovaných příkladů. Karnaughovu mapu prvního z nich je možné vidět na obrázku (Obrázek 1.2) a slouží k demonstraci minimalizace logické funkce o čtyřech vstupních proměnných se zahrnutím neurčitých stavů.



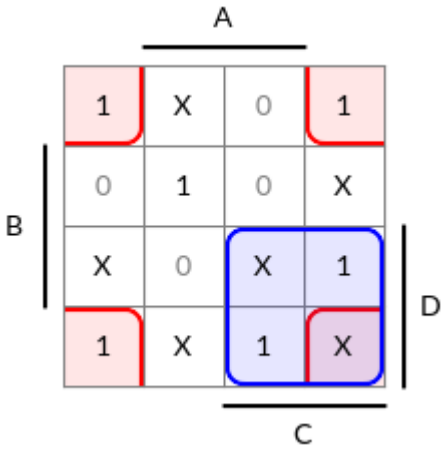
Obrázek 1.2: Podoba Karnaughovy mapy k minimalizaci.

Celý proces je vhodné započít vyhledáním minimalizační smyčky, která pokryje co nejvíce prvků obsahujících logickou jedničku. Je proto zvolena smyčka pokrývající čtyři prvky v rozích mapy, což je možné díky tomu, že spolu horizontální i vertikální strany mapy vzájemně sousedí.



Obrázek 1.3: Příklad minimalizace pomocí Karnaughovy mapy – krok 1 ze 3.

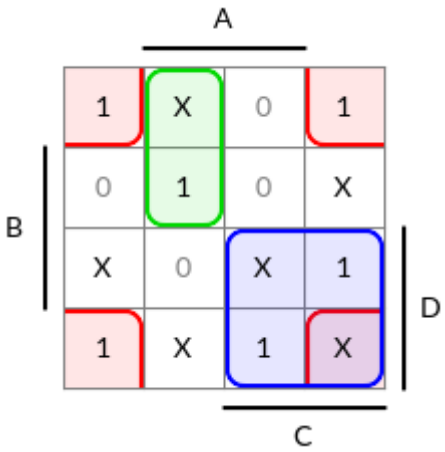
Na základě vytvořené smyčky je vyjádřen její minterm, který je dán součinem proměnných neměnících svou hodnotu napříč smyčkou. V tomto případě se jedná o tvar $\overline{AB}$. Stejným způsobem je možné najít i další smyčku, viz obrázek (Obrázek 1.4).



Obrázek 1.4: Příklad minimalizace pomocí Karnaughovy mapy – krok 2 ze 3.

Smyčky se mohou vzájemně překrývat, je tedy možné zahrnout čtveřici prvků v pravém dolním rohu mapy, čímž budou pokryty další dvě logické jedničky. Minterm nově vzniklé smyčky má tvar CD.

Jak je z obrázku (Obrázek 1.4) patrné, zbývá pokrýt už jen poslední prvek. Nejlepším způsobem je jeho zahrnutí zelenou smyčkou na obrázku (Obrázek 1.5), jejíž minterm má tvar $A\overline{CD}$.

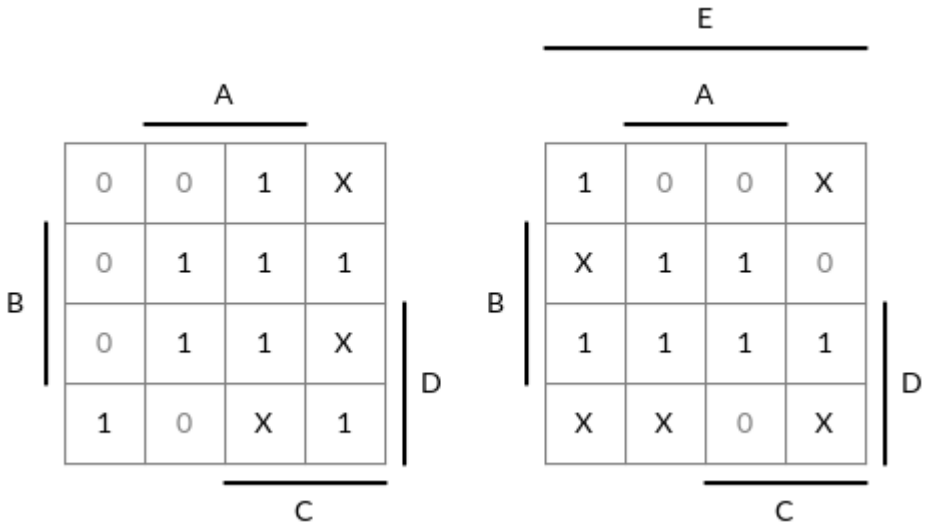


Obrázek 1.5: Příklad minimalizace pomocí Karnaughovy mapy – krok 3 ze 3.

Tímto způsobem byly zahrnuty všechny požadované prvky – logické jedničky. Nyní zbývá na základě vyjádřených mintermů vytvořit součtový tvar funkce jejich sečtením, viz výsledná rovnice (1.5).

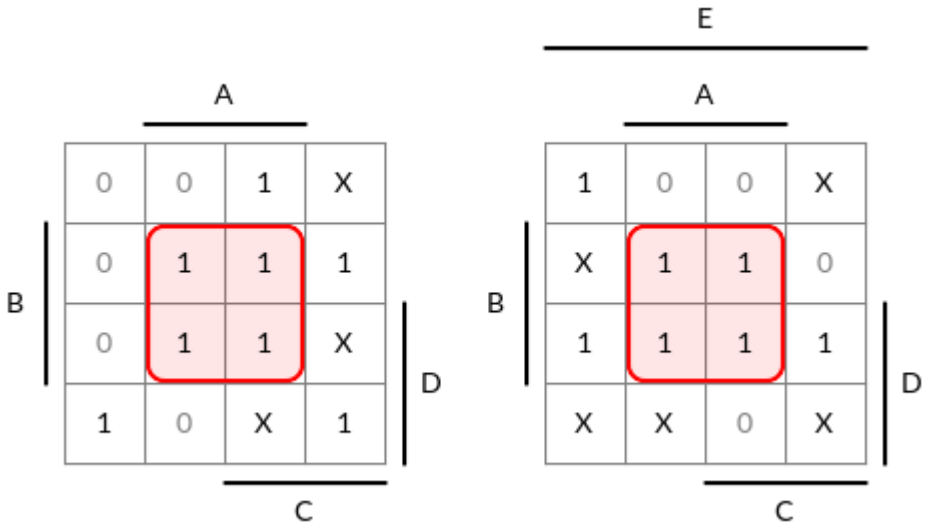
$$y = \overline{AB} + CD + A\overline{CD} \tag{1.5}$$

Obdobně proces minimalizace probíhá i v případě výskytu více vstupních proměnných. Jako příklad byla zvolena dvojice Karnaughových map na obrázku (Obrázek 1.6), na níž je možné nastínit průběh minimalizace napříč více mapami.



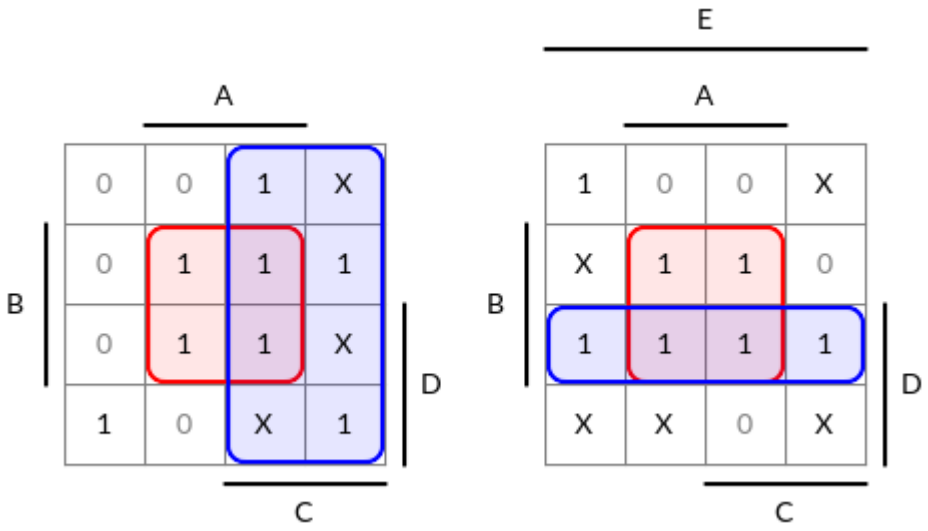
Obrázek 1.6: Podoba Karnaughových map k minimalizaci.

Samotná minimalizace má stále stejný průběh. Tentokrát je však nutné brát v potaz vhodnost minimalizačních smyček nacházejících se na obou mapách zároveň. V případě výskytu na jedné mapě by bylo nutné minterm smyčky dále spojit s hodnotou proměnné, pod kterou mapa spadá. V tomto případě se jedná o proměnnou E. Přímo uprostřed obou map se nabízí smyčka pokrývající čtveřici logických jedniček, celkem tedy osm prvků, jak je možné vidět na obrázku (Obrázek 1.7).



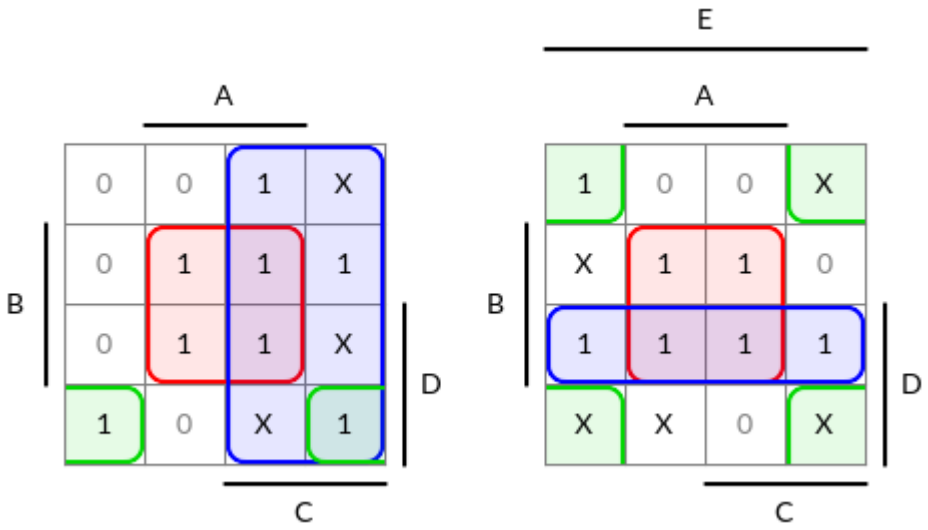
Obrázek 1.7: Příklad minimalizace pomocí Karnaughových map – krok 1 ze 3.

Její minterm má tvar AB. Jelikož se hodnota proměnné E napříč smyčkami mění, není nutné ji do výsledného tvaru zahrnout. Žádná další smyčka, nacházející se na obou mapách zároveň se již nenabízí. Vyberou se proto největší smyčky v rámci dílčích map, viz (Obrázek 1.8).



Obrázek 1.8: Příklad minimalizace pomocí Karnaughových map – krok 2 ze 3.

Minterm nově vzniklé smyčky uvnitř levé mapy má tvar AB. V případě pravé mapy pak smyčka nabývá tvaru BDE. Na obou mapách zbývá jediná logická jednička, provede se proto poslední krok spočívající v jejím zahrnutí (Obrázek 1.9).



Obrázek 1.9: Příklad minimalizace pomocí Karnaughových map – krok 3 ze 3.

Nyní je na řadě spojení získaných mintermů do součtového funkčního tvaru, stejně jako v minulém příkladu, čímž se po dodatečné úpravě získá konečný výsledek (1.6).

$$y = AB + C\bar{E} + BDE + \overline{AB}D\bar{E} + \overline{AB}E = \overline{AB}(E + D) + B(A + DE) + C\bar{E} \qquad (1.6)$$

2 TECHNOLOGICKÝ ZÁKLAD PRÁCE

Prostřednictvím této kapitoly budou nastíněny technologie, na jejichž základě lze vytvořit desktopovou aplikaci splňující zadané cíle. Nejprve bude představen zvolený programovací jazyk C++, dále několik datových formátů využívaných aplikací a vhodné varianty knihoven pro tvorbu grafických uživatelských rozhraní a zpracování obrazu využitím zvoleného programovacího jazyka.

2.1 Programovací jazyk C++

Programovací jazyk C++ pojí prvky procedurálního, objektově orientovaného a generického programování. O jeho vznik se zasloužil programátor dánského původu Bjarne Stroustrup již na samotném počátku osmdesátých let minulého století v Bellových laboratořích. Jeho původním záměrem bylo obohatit jazyk C o vlastnosti objektově orientovaného programování. V roce 1985 vydal Stroustrup knihu *The C++ Programming Language* a jazyk byl poprvé implementován do komerčního produktu. Stále však nebyl standardizován, k čemuž poprvé došlo až v roce 1998. Od té doby jazyk ušel dlouhou cestu a dočkal se řady menších i větších revizí standardu. Ta poslední nese název *C++17* a byla uvedena v roce 2017. Jazyk se již nedlouho po svém vzniku těšil velké oblibě a stal se tak jedním z nejrozšířenějších programovacích jazyků na světě. [4; 5]

Objektově orientované programování zdůrazňuje data namísto algoritmů, jež jsou upřednostňovány procedurálním paradigmatem. Umožňuje tvorbu znovupoužitelného a zapouzdřeného kódu i využití dědičnosti a polymorfismu. Generické paradigma naproti tomu představuje prostředek pro tvorbu abstraktního kódu a společných, typově nezávislých, úloh. Základ v podobě jazyka C s sebou přináší řadu výhod. Každý zdrojový kód psaný v programovacím jazyce C je platný i v C++, s čímž souvisí i plná kompatibilita s knihovnami naprogramovanými v jazyce C.

Důležitou složkou jazyka C++ je jeho standardní knihovna šablon známá pod označením *STL*. Ta je realizovaná metodou generického programování a čítá nespočet obecně použitelných struktur, tříd a algoritmů. Byla vyvinuta na počátku devadesátých let Alexanderem Stepanovem v laboratořích *Hewlett Packard* a v roce 1998 se stala významnou součástí standardu *C++98*. Díky tomu je nedílnou součástí všech implementací jazyka. Jedná se o kolekci šablon reprezentující homogenní úložné jednotky a prostředky pro práci s nimi, funkční objekty, nejrůznější algoritmy a další.

2.2 Datové formáty

Význam použití datových formátů tkví ve sjednocení struktury a obsahu jimi formátovaných souborů. Mají širokou škálu využití ve správě projektových, konfiguračních, překladových a mnohých jiných typů souborů, přičemž vyčnívají jejich přednosti v podobě čitelnosti a upravitelnosti lidmi i stroji, možnosti zachování zpětné kompatibility i kompatibility mezi platformami a rozšiřitelnosti. V této kapitole je blíže popsána trojice v projektu použitých datových formátů.

2.2.1 JSON

Označení *JSON* nese formát primárně určený pro přenos a zpracování dat. Zkratka *JSON* v překladu znamená JavaScriptová objektová notace, z čehož je patrné, že staví na základech právě tohoto programovacího jazyka. Jedním z cílů formátu je jednoduchost čtení a zápisu dat, což se týká nejen jejich zpracování a generování strojem, ale především samotným uživatelem. To je jedním z hlavních důvodů, proč se v moderním světě stal *JSON* tolik oblíbeným. [6]

Struktura formátu staví na základě dvou univerzálně používaných datových struktur, objektů a polí. Pod označením objekt se skrývá neseřazená množina párů klíč-hodnota, představující jeho dílčí prvky. Klíčem je v tomto případě vždy neprázdný řetězec nesoucí název prvku. Samotnou hodnotu tvoří data libovolného typu, ať už se jedná o reálnou, celočíselnou nebo logickou konstantu, řetězec, pole hodnot či další vnořený objekt. Díky tomu je možné data strukturovat zcela libovolně dle potřeby.

Datový formát *JSON* se v dnešní době řadí mezi ty nejrozšířenější a je nativně podporován řadou programovacích jazyků. C++ mezi ně ovšem nepatří, a to ani ve svém nejmodernějším standardu *C++17*. Je tedy nutné využít některou z existujících externích knihoven, které se liší nabídkou funkcí, strukturou, výkonem či jinými vlastnostmi. Patří mezi ně knihovny *JSON++*, *rapidjson*, *libjson* a mnohé další.

2.2.2 XML

O vývoj a standardizaci datového formátu *XML* neboli rozšiřitelného značkovací jazyka se postarala v roce 1996 standardizační organizace *World Wide Web Consortium*. Klade si za cíl jednoduchost, čitelnost a snadný návrh dokumentů, použitelnost napříč internetem a podporu v rámci široké škály aplikací. Formát vychází ze značkovacího jazyka *SGML*, na jehož kořenech staví i jazyk pro tvorbu webových stránek *HTML*. [7]

Každý dokument se skládá z elementů, které mohou přímo obsahovat data nebo se dále větvit. Ty jsou ohraničeny dvojicí značek a je možné jim přiřadit libovolný počet unikátních atributů. Neexistují však žádné předdefinované značky ani atributy, je potřeba definovat vlastní. Z toho důvodu se formát nazývá rozšiřitelným, jelikož není omezen žádnou množinou přípustných prvků. Dokument může obsahovat i komentáře v podobě speciálně označených elementů. Standard popisuje jen základní prvky jazyka, návrh struktury celého dokumentu proto leží čistě v rukou uživatele.

XML si zakládá na plné podpoře kódování znaků *Unicode* pro podporu tvorby obsahu ve zcela libovolném mluveném jazyce. To se týká nejen ukládaných dat, ale také názvů atributů, značek a ostatních prvků dokumentu. Specifikace *XML 1.0* také zahrnuje podporu šifrování, jmenných prostorů, digitálních podpisů a dalších zajímavých vlastností. Jazyk dal za vznik stovkám dalších formátů založených na jeho syntaxi. Mezi ně se řadí například grafický formát *SVG* či otevřený formát dokumentů pro kancelářské aplikace *ODF*. Využití tak nachází v širokém spektru různých odvětví. Stal se však terčem kritiky kvůli zbytečné komplikovanosti a redundantnosti zápisu dat, a hlavně komplikovanosti mapování složitějších *XML* struktur do struktury stávajících programovacích jazyků.

2.2.3 CSV

Zkratka *CSV* (čárkami oddělené hodnoty) označuje jednoduchý souborový formát určený pro práci s tabulkovými daty a jejich ukládání i přenos mezi řadou různých tabulkových procesorů a jiných aplikací. Jedná se o dlouhá léta používaný formát, který ovšem dosud nebyl standardizován. Vyskytuje se proto v několika různých variantách, což je dáno neexistencí jednotné specifikace. Ty se nejčastěji odlišují použitým oddělovačem. Existuje například varianta zvaná *TSV* (tabulátorem oddělené hodnoty), která využívá tabulátor namísto čárky. [8]

Největší výhoda formátu spočívá v naprosté triviálnosti tvorby i zpracování souborů. Jejich struktura má pouze několik základních pravidel. Jednotlivé sloupce tabulky jsou vždy oddělovány čárkou, řádky pak běžným odřádkováním. Textový obsah buňky je možné ohraničit uvozovkami, což má za následek ignoranci případného výskytu oddělovače uvnitř ní. Za nesprávně formátovaný lze považovat soubor s neodpovídajícím počtem sloupců na každém řádku, jiné nesrovnalosti nelze z hlediska struktury formátu definovat, jelikož veškeré znaky krom uvozovek a oddělovačů jsou považovány za reprezentaci dat.

V dnešní době již formát *CSV* není natolik aktuálním a často bývá nahrazen formáty modernějšími, které umožňují tvorbu podrobnější struktury souboru. Stále je však velmi rozšířený a používaný především v nenáročných případech zpracování tabulkových dat.

2.3 Grafický formát SVG

SVG (škálovatelná vektorová grafika) je grafickým formátem založeným na univerzálním značkovacím jazyce *XML* (viz kapitola 2.2.2), jehož doplňuje o sadu použitelných značek a atributů sloužících k vykreslování obrázků. O jeho vývoj a následnou standardizaci se postarala stejná organizace, jako v případě datového formátu *XML* - *World Wide Web Consortium*. Poprvé byl standardizován v roce 1999 a v současnosti existuje ve verzi *SVG 1.1*, která byla uvedena v roce 2011. Textový základ souborů umožňuje jejich snadné prohledání, indexování, kompresi či další zpracování jejich obsahu. Díky tomu je též možné obrázek otevřít, upravit či případně celý vytvořit prostřednictvím textového editoru. *SVG* patří mezi technologie navržené ke spolupráci s webovými technologiemi. Fakt, že se jedná o vektorový formát znamená, že je obsah škálovatelný do libovolných rozměrů bez ztráty kvality a je tedy vhodný i k tisku ve vysoké kvalitě. [9]

Obsah založený na formátu *SVG* se může skládat ze široké škály prvků, které umožňují provádět komplexní kompozici obrázků a diagramů. Mezi podporované funkce patří jednoduché vykreslování všech základních obrazců včetně podpory jejich pokročilého maskování a aplikace základních transformací. Je však nutné brát v potaz, že některé prvky mohou různé aplikace interpretovat různě, podobně jako tomu bývá v případě webových technologií. Formát je podporován valnou většinou webových prohlížečů, především díky jeho zahrnutí do standardu *HTML 5*, vektorových grafických editorů i jiných aplikací.

2.4 Knihovny pro tvorbu uživatelských rozhraní

Volba knihovny pro tvorbu grafických uživatelských rozhraní je velmi podstatná, neboť právě na ní závisí směr, kterým se celý projekt, a obzvláště pak jeho zdrojový kód, bude ubírat. Z důvodu požadavku na přenositelnost programu nepřipadá v úvahu využití systémových knihoven, byla proto zvolena trojice vhodných alternativ.

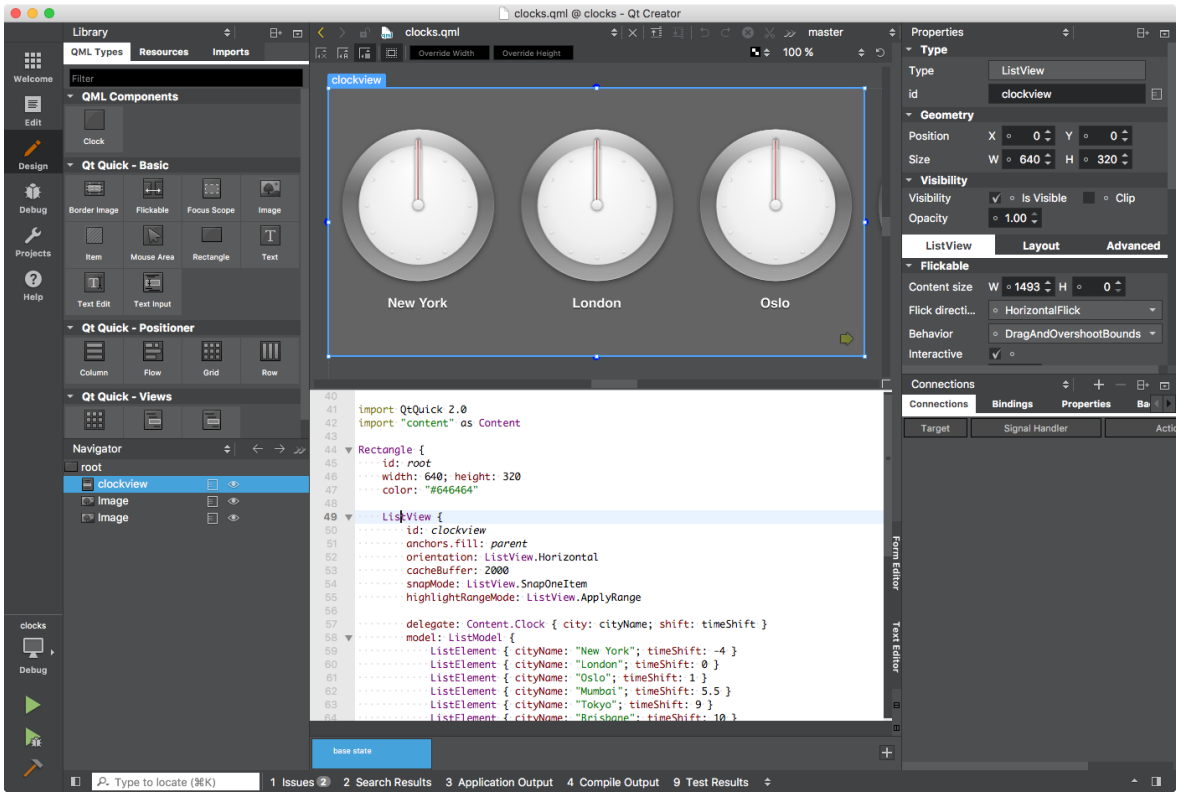
2.4.1 Qt

Knihovna *Qt* nabízí nepřeberné množství technologií, na jejichž základě je možné stavět nejrůznější druhy interaktivních aplikací. Je běžně používaná v oblasti tvorby desktopových

a mobilních uživatelských rozhraní, ale také v automobilovém průmyslu, automatizaci, ve zdravotnictví a mnohých dalších odvětvích. Počátek vývoje knihovny sahá do roku 1990, kdy jej zahájila norská společnost *Trolltech* - nynější *The Qt Company*. Během své dosavadní existence se osvědčila v rámci nepřeberného množství projektů a mezi její uživatele patří společnosti *AMD*, *Autodesk*, *Siemens*, *Wolfram Research* a mnohé další. [10]

Qt si zakládá na vysoké spolehlivosti, stabilitě a výkonu. Je vysoce modulární, což umožňuje využít jen ty části knihovny, které jsou doopravdy potřeba. S přihlédnutím k jejím rozměrům se jedná o vítaný bonus. Ke zefektivnění práce programátora a dalšímu rozšíření schopností jazyka C++ zahrnuje knihovna nástavný *preprocessor MOC*, který se spouští ještě před samotným zahájením překladu kódu. To umožňuje uvnitř něj využít speciálně označených prvků, mezi něž patří i tak zvané signály a sloty. Jejich prostřednictvím lze intuitivně reagovat na nejrůznější události. [11]

Pro tvorbu desktopových uživatelských rozhraní knihovna nabízí vedle jejich přímého zápisu do zdrojového kódu aplikace také dvojici nástrojů určených k této činnosti. První z nich, *Qt Designer*, je součástí integrovaného vývojového prostředí *Qt Creator* a umožňuje provádět vizuální návrh a kompozici uživatelských rozhraní z předem připravených prvků. K tomu je možné využít i podpůrných nástrojů pro testování programu v nejrůznějších podmínkách. Jeho výstupem je zdrojový kód v jazyce C++ využívající mechanismu signálů a slotů. Druhým nástrojem v nabídce je *Qt Quick Designer* (Obrázek 2.1), s jehož pomocí lze vytvářet uživatelská rozhraní využitím modulu *Qt Quick* a s ním souvisejícím kódu v jazyce *QML*. Pro zbylou část zdrojového kódu aplikace se dá použít jazyk C++. *QML* je deklarativní jazyk založený na syntaxi podobné formátu *JSON*. Slouží primárně k popisu vizuálních komponentů uživatelského rozhraní a jejich vzájemné interakce.



Obrázek 2.1: Nástroj Qt Quick Designer. [19]

Knihovna *Qt* se pyšní obsáhlou dokumentací a patří mezi ty nejlépe dokumentované projekty, a to díky své četné komunitě uživatelů. Samotná tvorba uživatelských rozhraní je jen jedním z mnoha modulů knihovny jako celku. Její součástí je také interpret jazyka *JavaScript*, vestavěný webový prohlížeč a široká škála modulů pro připojení k databázím, zobrazování audia a videa, pro podporu tvorby hardwarově akcelerovaných grafických aplikací a mnoho dalších. Pro tvorbu menší aplikace však *Qt* může představovat přílišnou zátěž. Aneb jak se říká, méně je někdy více.

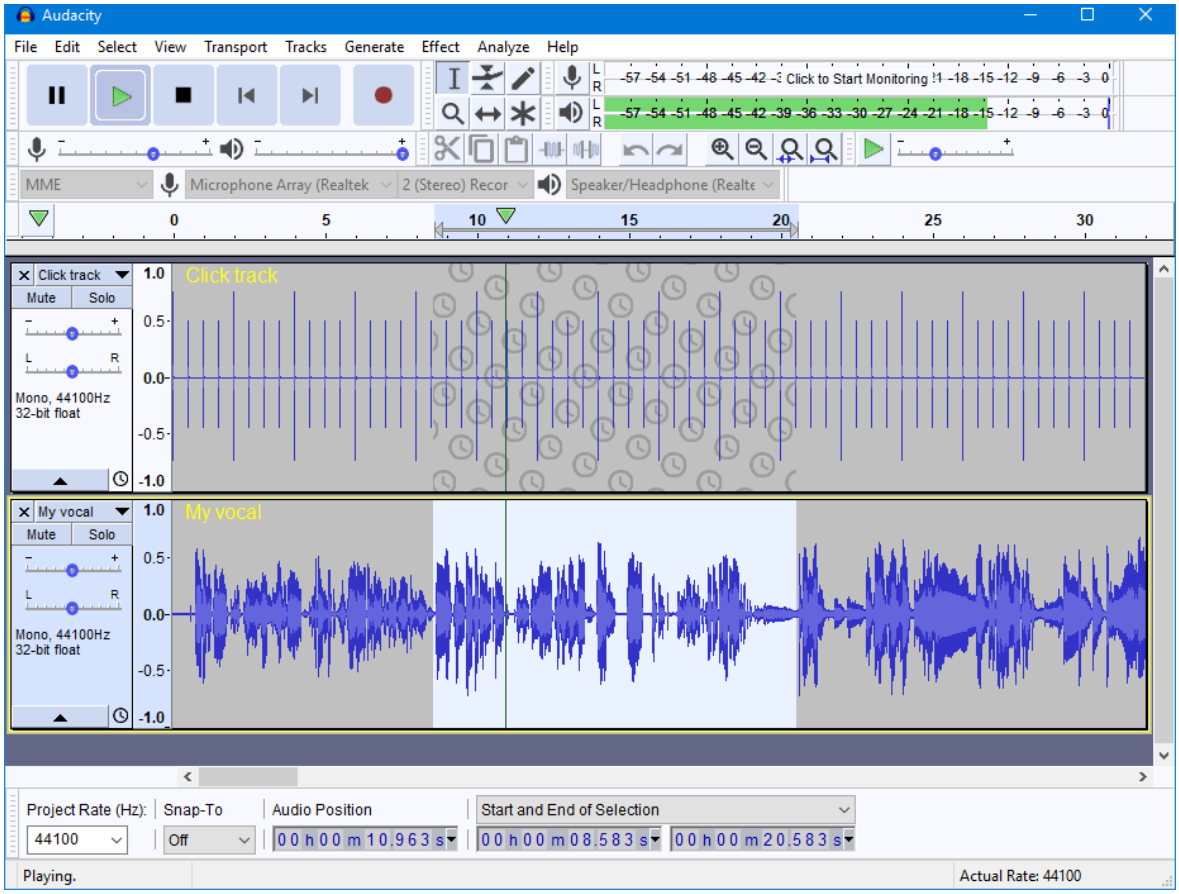
2.4.2 wxWidgets

Počátek vývoje knihovny *wxWidgets* se datuje do roku 1992 a je aktivní dodnes. Jejím autorem je Julian Smart, který v té době působil v ústavu aplikací umělé inteligence Edinburské univerzity. Projekt původně sloužil jako nástroj pro tvorbu multiplatformních aplikací pro systémy *Windows* a *Unix*. V průběhu let se okruh jeho podporovaných platforem rozrostl o *macOS* a některé další. Je dokonce možné využít knihovnu *Qt* jako platformovou základnu, což umožňuje dále rozšířit působnost navrhované aplikace. Mezi nejznámější uživatele *wxWidgets* se řadí společnosti *AMD*, *NASA*, *Xerox* a jiné. [12]

Nejvýraznější vlastností, kterou se knihovna *wxWidgets* odlišuje od konkurence, je její využití nativních systémových knihoven namísto vykreslování vlastních prvků. Vytvořená aplikace snáze zapadne do cílového prostředí. Koncovému uživateli to umožní snazší orientaci v uživatelském rozhraní, jelikož bude pracovat s prostředky, na které je z ostatních aplikací zvyklý. Zvolený přístup ovšem nese i určité nevýhody. Stále je nutné počítat s možnou nekonzistencí prvků napříč platformami. Hlavní nevýhodou je ovšem dostupnost pouhé podmnožiny prvků k tvorbě rozhraní, což je dáno tím, že si knihovna zakládá na využití nativních systémových prvků. Zbytek prvků je nutné simulovat, k čemuž ale knihovna nabízí prostředky. Díky tomu je možné repertoár knihovny v případě potřeby rozšířit. Ve většině případů se jedná o méně využívané prvky, ty nejčastěji používané se vyskytují na většině platforem. Rovněž se nabízí i široká paleta již existujících rozšíření.

Uživatelská rozhraní lze navrhovat přímo ve zdrojovém kódu aplikace nebo prostřednictvím několika externích nástrojů. Jedním z nich je nástroj pro vizuální návrh prostředí zvaný *wxFormBuilder*. Dále se nabízí možnost využití na jazyce *XML* založeného formátu *XRC*, který umožňuje návrh dialogů, nabídek či nástrojové lišty čistě v textové podobě. Soubor v tomto formátu dokáže knihovna v případě potřeby načíst a vytvořit z něj odpovídající prvky nebo dokonce vygenerovat C++ kód, který je následně možné zahrnout přímo do zdrojového kódu aplikace.

Součástí knihovny je objemná dokumentace a množství příkladů, z nichž se dá vycházet při seznamování se s její strukturou. V nabídce je i řada modulů nad rámec tvorby samotného uživatelského rozhraní. Mezi ně patří například modul pro zpracování zvuku a videa, modul pro tisk, vykreslování *HTML* kódu, a mnohé další. Stejně jako v případě ostatních větších knihoven pro tvorbu uživatelského rozhraní jsou na programátora kladeny výrazné požadavky na strukturu kódu. Největší nevýhodou *wxWidgets* je v tomto směru zastaralost její struktury a kódu. Při práci s ní je nutné počítat s hojným využitím maker a jiných v dnešní době již příliš nepoužívaných prostředků. To je mimo jiné dáno její letitou existencí.



Obrázek 2.2: Aplikace založená na knihovně wxWidgets. [20]

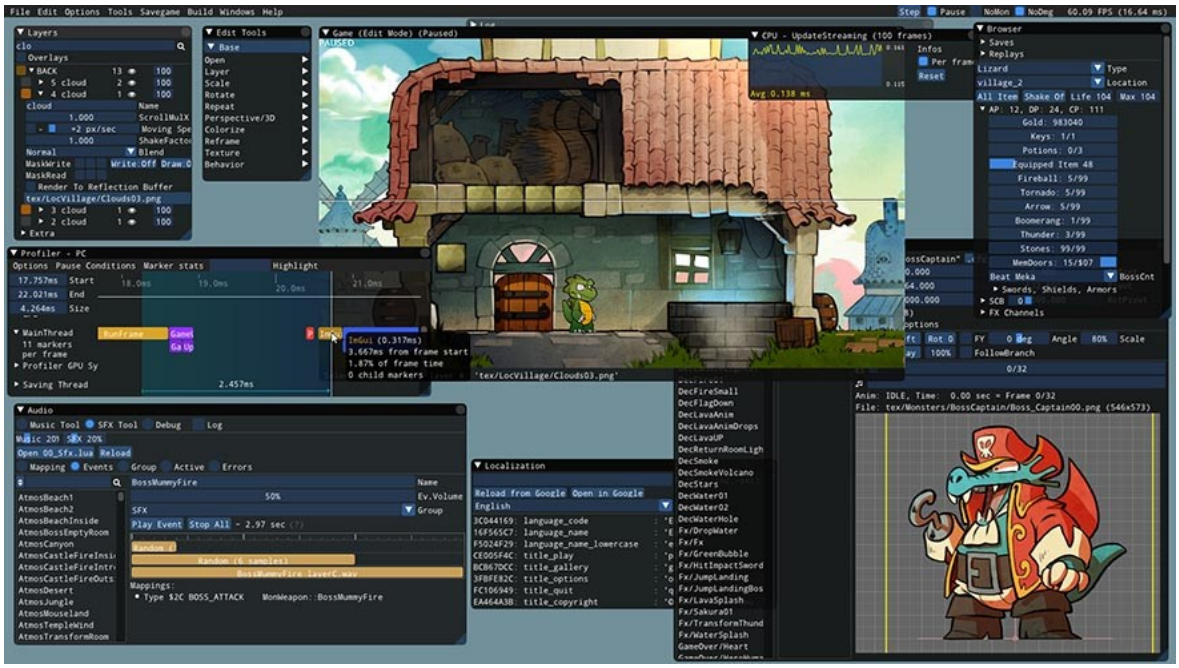
2.4.3 Dear ImGui

Dear ImGui je moderní a minimalistická knihovna pro tvorbu uživatelských rozhraní. Původně byla určená pro tvorbu nástaveb existujících aplikací, především videoher. To umožňuje její jednoduché a rychlé začlenění do stávajícího zdrojového kódu. Její vývoj je financován převážně herními vývojářskými studii, mezi které se řadí *Activision Blizzard*, *Bethesda*, *Valve* a jiné. V minulosti však na jejím základě vznikla řada samostatných nástrojů včetně jednoho od společnosti *Intel*. [13]

Knihovna *Dear ImGui* se od ostatních liší již ve svých základech – samotnou metodou tvorby uživatelských rozhraní. Ta probíhá výhradně prostřednictvím zdrojového kódu navrhované aplikace na základě metody zvané *IMGUI - Immediate Mode Graphics User Interface*. Její myšlenkou je přenechání vlastnictví stavů potřebných k vizualizaci prostředí a interakci s ním aplikaci, což činí její vývoj více transparentním a méně náchylným k chybám. Hlavní motivací tohoto přístupu je předcházení frekventovaných změn stavů a jejich zbytečné synchronizaci. Stavy v podobě veškerých textových řetězců a

jiných hodnot jsou knihovně v každém cyklu předávány ze strany aplikace. To umožňuje vytvářet vysoce dynamický obsah. [14]

Dear ImGui je samostatná a její jádro nezávisí na žádných externích knihovnách. Po jejím boku je nabízena řada implementací její vykreslovací i systémové vrstvy, které je možné volit zcela libovolně. Ty lze využít při budování samostatných aplikací. Výhodou je také fakt, že programátorovi nabízí prostředky, které sama využívá pro vykreslování uživatelského rozhraní a umožňuje mu tak jednoduše tvořit vlastní obsah. Největší nevýhodou knihovny je absence její dokumentace v běžném slova smyslu. Její autor apeluje na využití komentářů uvnitř zdrojového kódu a přiložených příkladů k jejímu osvojení. To ve spojení s naprosto minimálním výskytem návodů tvoří poměrně vysokou bariéru vstupu. Je to dáno především mládím knihovny, v budoucnu se situace při jejím širším využití jistě zlepší.



Obrázek 2.3: Aplikace založená na knihovně Dear ImGui. [21]

2.5 Knihovny pro zpracování obrazu

Aby aplikace byla schopná činit grafický export, je nutné zvolit vhodné vykreslovací jádro, které umožní tvorbu rastrových obrázků na základě jejich zadání ze strany kódu. Spolehlivých knihoven pro tvorbu obrazu umožňujících jeho kvalitní výstup je v jazyce C++ je poměrně málo, proto byli vybráni jen dva nejvhodnější kandidáti – *OpenCV* a *Magick++*.

2.5.1 OpenCV

Knihovna *OpenCV* je primárně zaměřená na využití v oblastech počítačového vidění a strojového učení. Sestává z více než dvou a půl tisíce vysoce optimalizovaných algoritmů, které je možné použít k široké škále různých činností. Mezi ně se řadí například detekce objektů či obličejů, kategorizace lidské činnosti na základě videa, vyhledání podobných obrázků v rámci databáze a mnohé další. Jednotlivé algoritmy jsou psány tak, aby bylo možné je použití v aplikacích běžících v reálném čase. Knihovna se těší široké oblibě a je využívána při provádění rozmanitých činností v rámci vládních organizací, výzkumných ústavů i nadnárodních korporací, mezi něž patří společnosti *Google*, *Microsoft*, *IBM*, *Intel* a mnohé další. [15]

Jednou ze základních schopností knihovny *OpenCV* je tvorba a zpracování obrázků. Jedná se o naprosto nepatrný zlomek jejího obsahu, právě na něm však staví podstatná část algoritmů. Knihovna je vysoce modulární, umožňuje tedy využít pouze ty části, které jsou opravdu potřeba. Pro účely vykreslování a následného ukládání obrázků postačí modul jádra ve spojení s moduly *Image Processing* a *Image Codecs.* Modul jádra je kompaktní a definuje pouze základní datové struktury a operace nad nimi používané ostatními moduly. Především se jedná o optimalizovanou práci s vícerozměrnými poli či propojení s externími knihovnami. Modul *Image Processing* slouží ke tvorbě a zpracování obrazu a obsahuje podporu vykreslování geometrických obrazců včetně jejich vyhlazování a podpory maskování. Dále nabízí možnost aplikování obrazových transformací, přemapování barev, rozmazání či zostření obrazu, tvorbu histogramů a další funkce. Data obrázků jsou v paměti uložena v barevném prostoru *BGRA* oproti běžně používanému *RGBA*, což se týká i definic barev ve zdrojovém kódu. Modul *Image Codecs* slouží pouze k načítání a ukládání obrázkových souborů v celé řadě rastrových formátů, mezi něž patří často používané formáty *PNG*, *JPEG*, *TIFF*, *WEBP*, *HDR* a mnohé další. O vnitřním formátu souboru pak knihovna rozhoduje dle koncovky souboru.

2.5.2 Magick++

Magick++ představuje aplikační programovací rozhraní nástroje pro zpracování obrazu *ImageMagick*, jehož historie sahá až do konce 80. let 20. století. Jeho hlavním účelem je tvorba, provádění úprav a další zpracování obrázků. Pyšní se podporou více než dvou set souborových formátů nastřádaných za celou dobu jeho existence. Jedná se o rastrové i vektorové grafické formáty, většina z nich však již není aktuální. Samozřejmostí je podpora

moderních a v dnešní době nejpoužívanějších formátů, ať už s průhledností nebo bez ní. Ukládání do vektorových formátů je ovšem řešeno pouhým vložením rastrového obrázku, namísto jednotlivých obrazců, ze kterých byl výsledek složen. Nejedná se tudíž o vektorový výstup, který umožňuje libovolnou změnu rozměrů bez ztráty kvality. Nástroj bývá často využíván k pouhému převodu grafických souborových formátů. Samotná podpora souborových formátů je realizována prostřednictvím dynamicky načítaných knihoven. Díky tomu lze vyřadit nepotřebné formáty pouhým nezahrnutím jejich dynamických knihoven po boku hlavního spustitelného souboru programu. [16]

Nejdůležitější vlastností knihovny je její schopnost vykreslování obrazců. Umožňuje kompozici obrázku prostřednictvím řady podporovaných geometrických obrazců, včetně polygonů a křivek, s podporou vyhlazování jejich okrajů, pokročilého maskování, využití vyplňovacích vzorů a dalších. Mimo to nabízí celou řadu ostatních operací, například aplikaci filtrů pro rozmazání či zostření obrazu, tvorbu animací nebo automatické ořezání obrázku tak, aby zahrnoval jen samotný obsah bez přebytečných pixelů na jeho okrajích. Třešničkou na dortu je podpora zpracování až terapixelových obrázků, což umožňuje optimalizované jádro využívající několik nezávislých výpočetních vláken. Knihovna je tedy velmi dobře optimalizovaná a nečiní jí problém ani takto náročná činnost.

II. PRAKTICKÁ ČÁST

3 ALGORITMUS PRO MINIMALIZACI LOGICKÝCH FUNKCÍ

Tato kapitola se zaměřuje na popis navrženého algoritmu pro minimalizaci logických funkcí, který se skládá celkem ze čtyř hlavních kroků popsaných v odpovídajících podkapitolách. Ke své činnosti využívá obou metod minimalizace nastíněných v teoretické části práce.

3.1 Tvorba minimalizačních smyček

Základem ideálního pokrytí Karnaughových map je nalezení vhodné kombinace minimalizačních smyček. Důležité je zvolit správnou metodu rozložení samotných map pro případ, že bude minimalizována funkce o více než čtyřech vstupních proměnných. Právě od toho se odvíjí veškerá činnost algoritmu. Vzhledem k nutnosti podporovat až osm proměnných na vstupu funkce je nutné počítat s řadou možných komplikací. Proto jsem zvolil dle mého názoru tu nejpraktičtější variantu z hlediska návrhu algoritmu, konkrétně rozdělení dat do několika zcela nezávislých map, jejichž počet se odvíjí od počtu proměnných na vstupu. Pro čtyři či méně proměnné postačí jediná Karnaughova mapa, při větším množství však počet map odpovídá hodnotě $2^{(n-4)}$, kde n označuje počet vstupních proměnných. Díky tomu je možné aplikovat totožný algoritmus na každou mapu zvlášť bez ohledu na celkový počet proměnných a na ostatní mapy.

Algoritmus svou činnost započíná vytvořením všech validních kombinací minimalizačních smyček pro každou Karnaughovu mapu zvlášť. Pro každý prvek, v němž se nachází logická jednička je vytvořena základní smyčka, z níž se jejím rekurzivním rozšiřováním do všech směrů postupně získají všechny možné kombinace. Z vygenerovaných smyček se v navazujících krocích vybírají pouze ty nejvhodnější, a to na základě několika kritérií. Nejprve je však nutné seznam nalezených smyček oprostit od duplicitních prvků, aby nedošlo k jejich zbytečnému zpracování. Po získání všech unikátních prvků je vzhledem k faktu, že jsme s každou mapou až doposud nakládali zvlášť, nutné spojit totožné smyčky napříč více mapami do jednoho prvku a označit mapy, na nichž se nachází.

Nejpodstatnějším krokem celého procesu je výběr minimalizačních smyček ze všech nabízených. Rozhodování, která smyčka bude nebo nebude použita, probíhá na základě dvou kritérií. Tím hlavním je přídavek prvků na všech mapách, na nichž se může daná smyčka nacházet. Přídavkem se rozumí počet všech dosud nezahrnutých polí obsahujících pravdivostní hodnotu jedna, které by v případě jejího využití nově zahrnuty byly. Takto se prohledá celý seznam dostupných prvků a vybere se z nich první smyčka s nejvyšším přídavkem, tedy za předpokladu, že byla nalezena alespoň jedna taková, která pokrývá

nejméně jedno nové pole. Druhé kritérium přichází na řadu až v případě, že jsou nalezeny dvě či více smyček se stejným přídavkem. Z těch je nutné vybírat na základě jejich velikosti násobené počtem map, na nichž se nachází, jelikož je potřeba brát v potaz, že smyčku opakující se na více mapách je možné dále minimalizovat. Od ní se odvíjí komplexnost vygenerovaného výrazu, jelikož platí, že čím je větší plocha smyčky, tím méně proměnných bude v jejím výsledném tvaru figurovat a je tak vhodnějším kandidátem než menší smyčka.

V posledním kroku jsou všechny spojené a již vytříděné smyčky zpětně rozděleny zvlášť pro každou mapu, na níž se nachází. Činí se tak proto, aby bylo možné s nimi snadno manipulovat v rámci ruční minimalizace. Rovněž dochází k aktualizaci jejich mintermů, aby s nimi bylo možné dále pracovat. Každá minimalizační smyčka je definována bitovým polem, které označuje její pokrytí prvků mapy. Cílem je aktualizovat interní programový tvar všech smyček. Ten sestává ze dvou bitových polí označujících existenci proměnných a jejich hodnoty uvnitř smyčky, které jsou na počátku vynulovány. Algoritmus aplikovaný na každou smyčku zvlášť svou činnost započíná ověřením výskytu dílčích proměnných na horizontální a vertikální ose. V případě, že je velikost smyčky na dané ose rovna velikosti mapy, víme, že se hodnoty všech proměnných na této ose mění a nebudou tedy součástí mintermu smyčky. V opačném případě dochází k provedení bitového součinu s předdefinovanými maskami výskytu proměnných, jejichž bity korespondují s místy označenými čarou v grafické podobě mapy. Na základě výsledku dochází k rozhodnutí, zdali může být daná proměnná vyřazena nebo ne. Pokud je výsledek roven nule nebo hodnotě bitového pole pokrytí smyčky, pak se proměnná ve výsledku objeví, a to s pravdivostní hodnotou nula, pokud byl výsledek nulový, v opačném případě jí náleží hodnota logické jedničky. Pokud se ovšem výsledek nerovná ani jedné ze zmíněných hodnot, znamená to, že se hodnota proměnné napříč danou smyčkou mění a je možné ji z mintermu vyřadit. Při výskytu více než jedné Karnaughovy mapy ve funkci je tento výsledek pomocí bitového součtu spojen s výsledným mintermem mapy, který se generuje totožným způsobem před generováním mintermů prvků v rámci mapy.

3.2 Dodatečná minimalizace

Vygenerováním ideální kombinace minimalizačních smyček proces minimalizace ani zdaleka nekončí. Pro získání úplného výsledku je potřeba na jejich základě vytvořit funkční tvar, na něj dodatečně aplikovat základní pravidla Booleovy algebry a umožnit také vytýkání proměnných. Je nutné brát v potaz, že součástí aplikace je i možnost provádět zcela manuální

minimalizaci. Při ní má uživatel plnou kontrolu nad tvorbou minimalizačních smyček. Smyčky, nacházející se na více mapách, jsou z důvodu jednoduchosti ovládání aplikace oddělené a ve výsledku figurují jako dva či více nezávislých podvýrazů, které je možné v drtivé většině případů dále zjednodušit.

Prvním krokem je tedy aplikace základních pravidel Booleovy algebry. Není nutné zahrnout pravidla složitější, v podobě De Morganových zákonů, jelikož není cílem algoritmu celý výraz minimalizovat pomocí Booleovy algebry, ale jen dodatečně poupravit výsledek již provedené minimalizace metodou Karnaughových map. Stále je nutné brát v potaz, že uživatel může provést minimalizaci manuálně a dopustit se tak určitých nedodělků. Ze všeho nejdříve je nutné odstranit všechny duplicitní prvky. Následuje promazání všech samotných proměnných, které se v seznamu podvýrazů nachází v normální i negované podobě, využitím zákona o vyloučení třetího. Nejdůležitější vlastností celého algoritmu je schopnost vytýkání proměnných za účelem další rekurzivní minimalizace. Je proto vyplněn seznam četnosti výskytů jednotlivých proměnných v negované i normální podobě napříč zpracovávaným výrazem. Ze seznamu je následně vybrána proměnná s nejčastějším výskytem, pokud se tedy vyskytuje více než v jednom podvýrazu. Pokud ano, je vytknuta, a všechny podvýrazy, v nichž se nachází, jsou přesunuty do výrazu nového, na který se aplikuje celý algoritmus znovu. Dochází tedy k rekurzivnímu volání, po jehož dokončení se výsledek spojí s vytknutou proměnnou a uloží se zpět do hlavního výrazu. Vše zakončuje aplikace zákonů o adsorpci a adsorpci negace, které umožňují výraz dále zjednodušit v případě, že se osamocená proměnná nachází i v ostatních výrazech.

3.3 Generování výrazů

Předposledním krokem celého minimalizačního procesu je vygenerování textové podoby jeho výsledného výrazu. Ta je sestavena na základě uživatelem zvoleného výstupního formátu, který je dán sadou textových řetězců, definujících tvar elementárních konstrukcí v podobě konjunkce, disjunkce, rovnosti a dalších. Na místo speciálně označených podřetězců jsou pak do těchto tvarů vkládány textové reprezentace průběžně generovaných podvýrazů.

Flexibilita navrženého substitučního mechanismu umožňuje generování mnohem komplexnějších formátů, než by tomu bylo v případě pouhé konkatenace textových řetězců. Jednou z motivací pro návrh komplexnějšího systému byla podpora nástroje *Microsoft Word* v podobě matematických rovnic. To si žádá generování kompletního *MathML* kódu, který

se ihned po vložení do programu převede na rovnici. Pro generování výstupu lze využít libovolný z následujících formátů:

- Algebraický formát
- Programovací jazyk
    - *C/C++/C#/Java*
    - *Python*
    - *Structured Text*
    - *VHDL*
- Značkovací jazyk
    - *LaTeX*
    - *MathML*
- Libovolný uživatelem definovaný formát

3.4 Validace výrazů

Proces validace představuje způsob, jakým je možné ověřit, zda je vygenerovaný výraz kompletní a svými pravdivostními hodnotami odpovídá zadané funkci, kterou by jím mělo být možné spolehlivě nahradit. Největší význam validace spočívá ve schopnosti upozornit uživatele na neúplnost výsledku manuální minimalizace. Významnou roli ovšem sehrála i při vývoji a následném testování minimalizačního algoritmu.

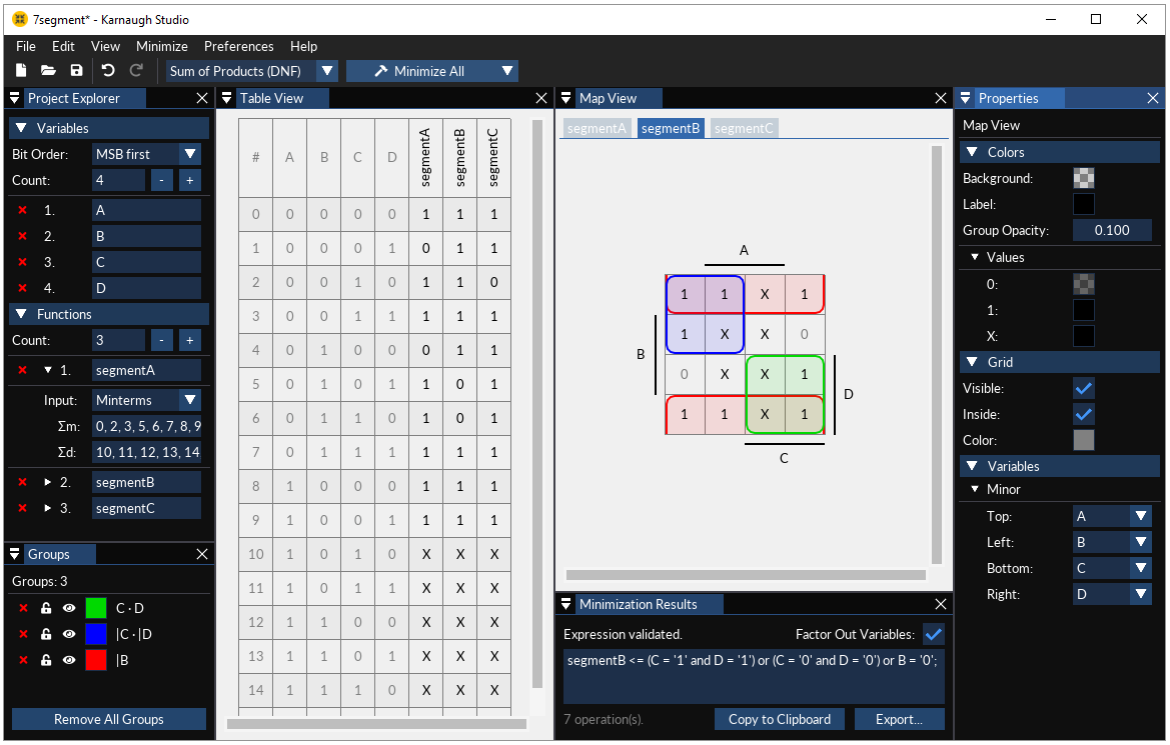
K validaci dochází po každém sestavení výstupního výrazu, tedy i při exportování výsledků minimalizace do souboru, kdy je ovšem možné validaci z důvodu úspory času deaktivovat. Jedná se o časově poměrně náročný proces, jehož činnost spočívá v postupné rekonstrukci pravdivostních hodnot z vytvořeného výrazu a jejich porovnání s hodnotami funkce původní. Pro zpracování logických výrazů jsem využil knihovnu *MathExpr*. Ta umožňuje zpracovávat matematické i logické výrazy v textové podobě využitím Dijkstrova algoritmu *shunting-yard*. Její nevýhodou je velká časová náročnost zpracování složitějších výrazů, čemuž se ale nelze zcela vyhnout. Doba validace je závislá na počtu proměnných i celkovém počtu podvýrazů nacházejících se v právě zpracovávaném výrazu.

Ze všeho nejdříve se znovu vygeneruje ověřovaný výraz, tentokrát však ve formátu na bázi jazyka C++, který je možné použít pro zpracování knihovnou *MathExpr*. Z důvodu zamezení případného vzniku konfliktů i navýšení výkonu následuje substituce názvů proměnných za písmena v rozsahu A až H. Činí se tak postupně od nejdelších názvů po

nejkratší, aby nedošlo k nechtěnému nahrazení části názvu proměnné, který obsahuje celé jméno jiné proměnné. Každá kombinace hodnot vstupních proměnných se následně doplní do výrazu upraveného pro potřeby validace a vyhodnotí se pravdivostní hodnota, která je porovnána s odpovídající funkční hodnotou neupravené vstupní funkce. Rovnost se nekontroluje pouze v případě, že je hodnota vstupní funkce pro tento prvek nastavena na neurčitý stav. Výraz je označen jako validní, pokud každý z jeho určitých stavů odpovídá stavu původní funkce.

4 UŽIVATELSKÉ ROZHRANÍ

Obsahem této kapitoly je detailní popis všech dílčích částí vytvořeného uživatelského rozhraní a jejich činnosti. Navržená aplikace nese název *Karnaugh Studio* a její podobu je možné vidět na obrázku (Obrázek 4.1) či ve větším formátu v rámci přílohy č. 1.



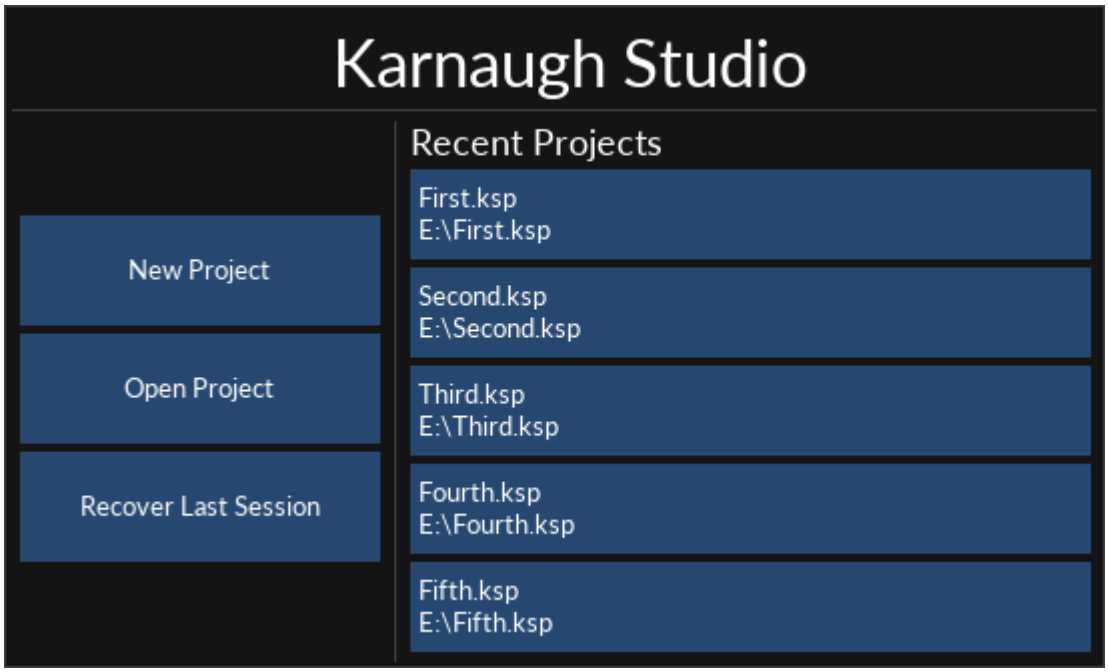
Obrázek 4.1: Aplikace Karnaugh Studio.

K jejímu vzniku posloužil nejmodernější standard programovacího jazyka C++, *C++17,* ve spojení s knihovnou pro tvorbu grafických uživatelských rozhraní *Dear ImGui*. Ta byla zvolena na základě řady kritérií, především z důvodu tvorby dynamického obsahu aplikace i minimálním dopadu na strukturu zdrojového kódu. Pro zpracování projektových a konfiguračních souborů byl zvolen datový formát *JSON* a pro tvorbu obrázkových souborů léty prověřená knihovna *Magick++*.

4.1 Uvítací obrazovka

První věcí, která na uživatele po spuštění programu čeká, je uvítací obrazovka. Její smysl spočívá v urychlení přístupu k nejčastějším operacím, které uživatel po startu běžně provádí. Nejužitečnější funkcí je rychlé načtení některého z nedávno upravovaných projektů. Jejich seznam se nachází v pravé části okna a sestává nanejvýš z pěti položek. Každá z nich je realizována formou tlačítka nesoucího název a kompletní cestu k souboru projektu. Kliknutím na něj dojde k okamžitému načtení zvoleného projektu.

Mezi další prvky okna se řadí tlačítka pro načtení existujícího projektu ze souboru, pro tvorbu nového projektu a pro obnovení předchozí relace, jež uvede program do stavu, ve kterém jej uživatel při posledním ukončení zanechal. Konečná podoba okna a jeho rozložení je znázorněna na obrázku (Obrázek 4.2).

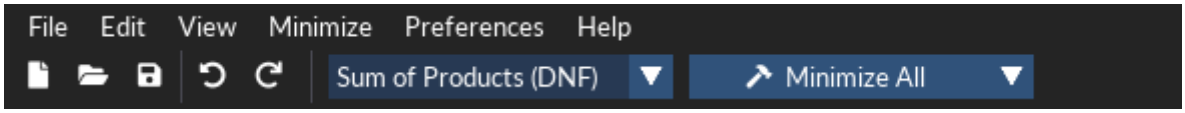


Obrázek 4.2: Uvítací obrazovka aplikace Karnaugh Studio.

Ke zobrazení uvítací obrazovky nedochází vždy po startu *Karnaugh Studio*. Prostřednictvím uživatelského nastavení je možné ji zcela deaktivovat. K jejímu zobrazení nedojde ani v případě, že byl program spuštěn se zadaným projektovým souborem, neboť rovnou dochází k jeho načtení. Toho lze dosáhnout například přímým otevřením takového souboru v průzkumníku systému *Windows* za předpokladu, že uživatel v minulosti provedl asociaci přípony souborů *ksp* s aplikací *Karnaugh Studio*.

4.2 Hlavní nabídka a panel nástrojů

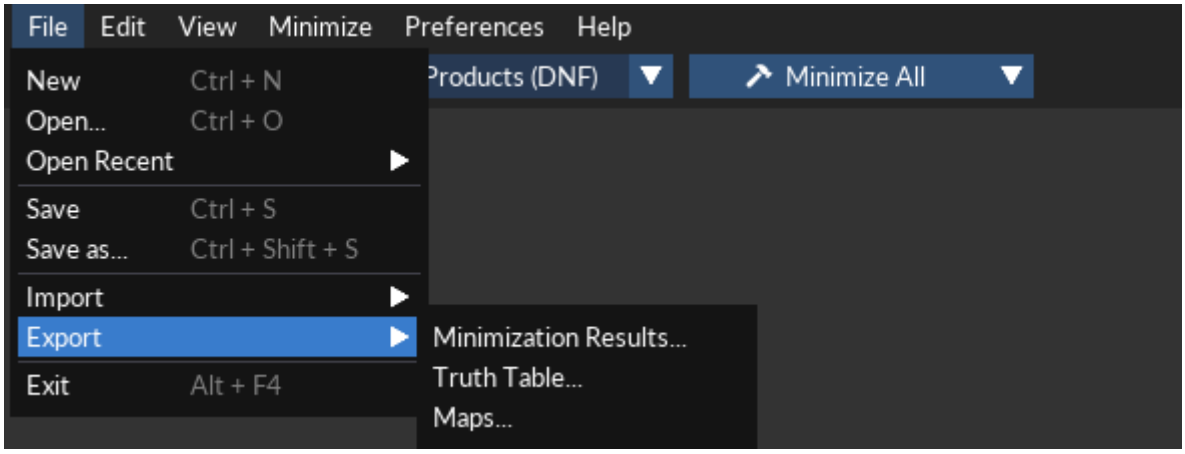
Hlavní nabídka představuje nedílnou součást prakticky každé desktopové aplikace. Jejím prostřednictvím může uživatel vykonávat celou řadu rozličných činností, ať už se jedná o práci s projektem nebo okny uživatelského rozhraní, provedení automatické minimalizace či cokoliv jiného. Většině těchto akcí je přiřazena klávesová zkratka, pomocí které je možné akci vyvolat, aniž by byl uživatel nucen vstoupit do hlavní nabídky. Ta je rozdělená do šesti jasně definovaných podnabídek. Jak je možné vidět na obrázku (Obrázek 4.3), patří mezi ně sekce *File*, *Edit*, *View*, *Minimize*, *Preferences* a *Help*.



Obrázek 4.3: Hlavní nabídka a panel nástrojů.

4.2.1 Nabídka File

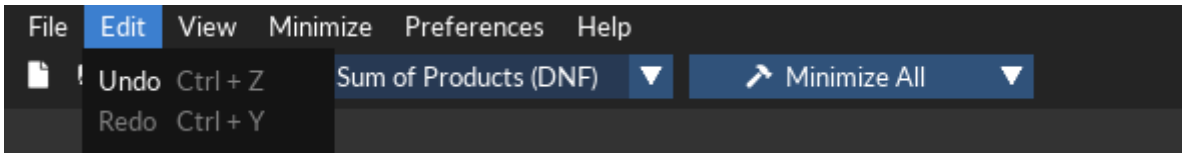
Nabídka *File* čítá celkem osm položek, z nichž tři se dále větví. Příkladem toho může být položka *Export*, která je k vidění spolu se zbytkem nabídky na obrázku (Obrázek 4.4). Po najetí na ni se zobrazí trojice dalších prvků označujících exporty výsledků minimalizace, pravdivostní tabulky a Karnaughových map. Jejich jedinou činností je otevření příslušného dialogového okna, prostřednictvím něhož může uživatel provést zvolený typ exportu. Stejným způsobem funguje i položka *Truth Table*, která se jako jediná skrývá v sekci *Import*. Dále má uživatel možnost vytvořit nový projekt nebo jej načíst ze souboru. To lze provést dvěma způsoby. Prvním z nich je výběr položky *Open Recent*, kdy se objeví podnabídka se seznamem čítajícím až deset nedávno upravovaných projektů, které je možné jediným kliknutím načíst. Druhým způsobem je zvolení položky *Open*, která zobrazí systémový dialog pro výběr souboru a umožní tak uživateli načíst zcela libovolný projekt. V případě nesprávného formátu vybraného souboru se zobrazí chybové hlášení, které na tuto skutečnost uživatele upozorní. Stejně jako položka *Open* funguje i *Save As*, která umožňuje stávající projekt uložit do libovolného souboru. Poněkud odlišně funguje položka *Save*. Ta projektový soubor ukládá do posledního zvoleného umístění. V případě, že projekt nebyl nikdy předtím uložen, funguje položka *Save* stejně jako *Save As*. Poslední prvek nabídky, *Exit*, slouží k ukončení aplikace.



Obrázek 4.4: Nabídka File.

4.2.2 Nabídka Edit

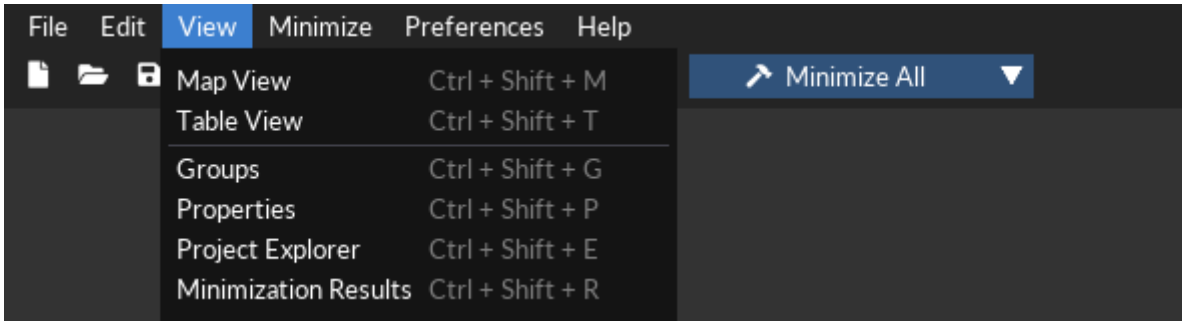
Druhá sekce hlavní nabídky se nazývá *Edit* (Obrázek 4.5). Jejím obsahem je dvojice prvků pro vrácení a obnovení uživatelem dříve provedené akce. Ty lze aktivovat pouze v případě, že je možné se daným směrem pohybovat v historii stavů projektu. Do ní jsou ukládány veškeré prováděné změny. Prostřednictvím položky *Undo* je možné načíst některý z předchozích stavů. Ve skutečnosti však dochází pouze k posunutí ukazatele aktivního stavu. Kliknutím na *Undo* se ukazatel přesune o jeden stav do historie a stisknutím *Redo* nazpět. To je možné pouze v případě, že ukazatel stavu neukazuje na stav nejnovější. V případě, že uživatel provede libovolnou změnu poté, co se již vrátil do historie, jsou všechny novější stavy smazány, nahrazeny stavem novým a dochází k vynulování již zmíněného ukazatele.



Obrázek 4.5: Nabídka Edit.

4.2.3 Nabídka View

Hlavním významem nabídky *View* (Obrázek 4.6) je umožnit snazší orientaci mezi okny uživatelského rozhraní a obnovit je v případě jejich zavření. Nabídka čítá celkem šest položek označujících hlavní okna aplikace v přesném pořadí *Map View*, *Table View*, *Groups*, *Properties*, *Project Explorer* a *Minimization Results*. Kliknutím na kteroukoliv z těchto položek dojde k aktivaci či znovuotevření zvoleného okna a jeho uvedení do popředí. To má význam například po provedení změn rozložení oken.

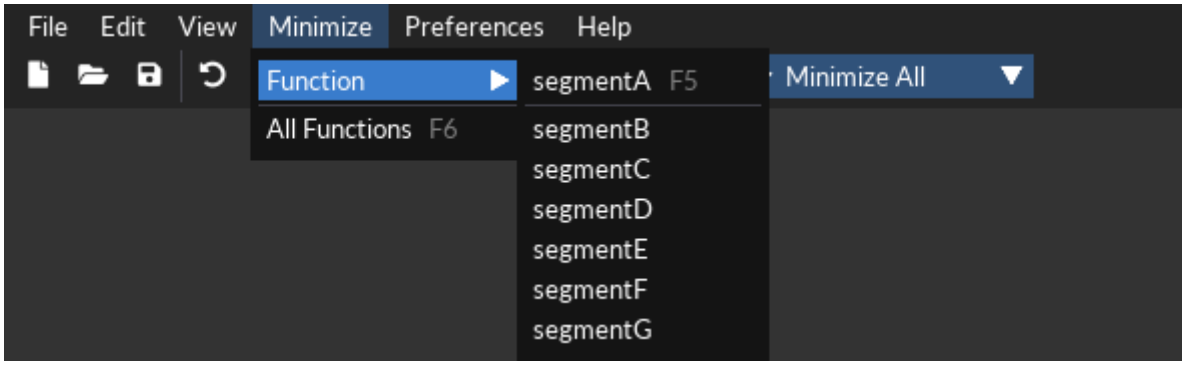


Obrázek 4.6: Nabídka View.

4.2.4 Nabídka Minimize

Obsah nabídky *Minimize*, jak již její název napovídá, slouží k provedení automatické minimalizace. Tu je možné aplikovat buď na jednu konkrétní funkci nebo na všechny funkce

zároveň pomocí položky *All Functions*. Konkrétní funkci lze zvolit ze seznamu nabízeného větvícím se prvkem *Function*. Na první místě je vždy uvedena funkce aktivní. Zbytek funkcí se nachází pod oddělovačem, pokud v projektu existuje více než jedna funkce, jak je možné vidět na obrázku (Obrázek 4.7). V případě existence minimalizačních smyček ve funkci určené k minimalizaci je uživatel upozorněn na jejich smazání, kterému může předejít.



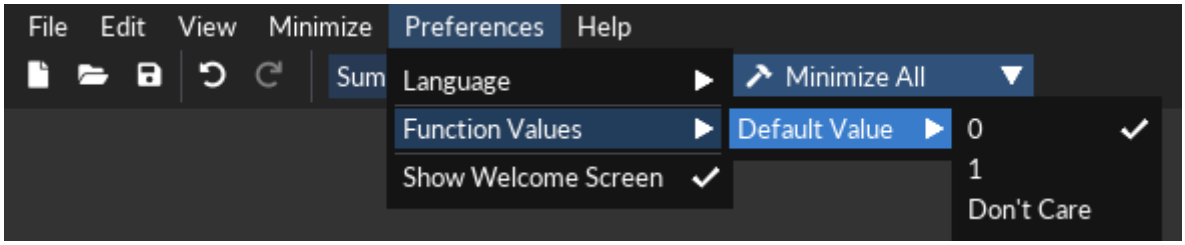
Obrázek 4.7: Nabídka Minimize.

4.2.5 Nabídka Preferences

Preferences slouží k provádění několika základních uživatelských nastavení. Prvním z nich je volba jazyka. Položka *Language* po aktivaci otevře podnabídku obsahující seznam všech dostupných překladů. Angličtina se nachází vždy na prvním místě, jelikož se jedná o výchozí jazyk aplikace, je definován přímo v kódu a je možné jej v případě potřeby kdykoliv obnovit. Zbytek jazyků je k dispozici pod oddělovačem. Každý z nich je reprezentován vlastním názvem, který je načten z odpovídajícího jazykového souboru ve složce *Languages*, která sousedí s hlavním spustitelným souborem programu. V případě, že daný soubor neobsahuje položku označující název jazyka, je k zobrazení použit název souboru. Při kliknutí na libovolný prvek seznamu dochází k okamžitému zpracování odpovídajícího souboru a nahrazení stávajících textů uživatelského rozhraní těmi novými. Chybějící řetězce jsou nahrazeny jejich anglickým ekvivalentem.

Další možností je nastavení výchozí funkční hodnoty prostřednictvím nabídky zobrazené po označení položky *Default Value* uvnitř *Function Values* (Obrázek 4.8). Tato funkční hodnota se vztahuje na všechny nově vzniklé funkce.

Poslední v řadě je volba *Show Welcome Screen*. Ta je zaškrtnutá v případě aktivity tohoto nastavení a umožňuje uživateli deaktivovat či případně znovu aktivovat uvítací obrazovku programu.



Obrázek 4.8: Nabídka Preferences.

4.2.6 Nabídka Help

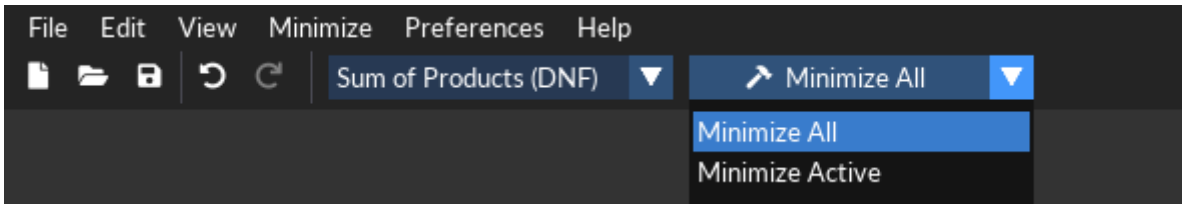
Poslední sekcí hlavní nabídky je sekce *Help*, která obsahuje jedinou položku s názvem *About Karnaugh Studio*. Po kliknutí na ni se zobrazí stejnojmenné okno (Obrázek 4.9), které tvoří přehled základních informací o programu a je strukturováno do dvou záložek. První z nich obsahuje pouze ty nejzákladnější informace. Ta druhá, nazvaná *Licenses*, skrývá seznam všech materiálů a knihoven použitých k vypracování projektu včetně jejich licencí, autorů a odkazů na webové stránky. Tento seznam je zobrazovaný prostřednictvím víceřádkového textového pole.

Obrázek 4.9: Okno About Karnaugh Studio.

4.2.7 Panel nástrojů

Panel nástrojů se nachází přímo pod hlavní nabídkou. Tvoří ho sada ikon a ostatních prvků, které představují rychlý přístup k vybraným akcím hlavní nabídky. V prvé řadě se jedná o trojici ikon pro správu projektu. Ty slouží k vytvoření nového, jeho načtení ze souboru či k uložení stávajícího. Od nich je oddělena dvojice editačních ikon pro vrácení a obnovení naposledy provedené akce. Následuje volba mezi součinovým a součtovým tvarem minimalizované funkce, která zároveň ovlivňuje zdroj minimalizačních smyček. Díky tomu je možné tvořit řešení oběma způsoby nezávisle na sobě. Posledním prvkem je tlačítko pro provedení automatické minimalizace. Tu lze provést buď pro všechny funkce zároveň nebo

jen pro tu aktivní. Mezi zmíněnými režimy je možné přepínat posledním tlačítkem ve tvaru šipky (Obrázek 4.10).



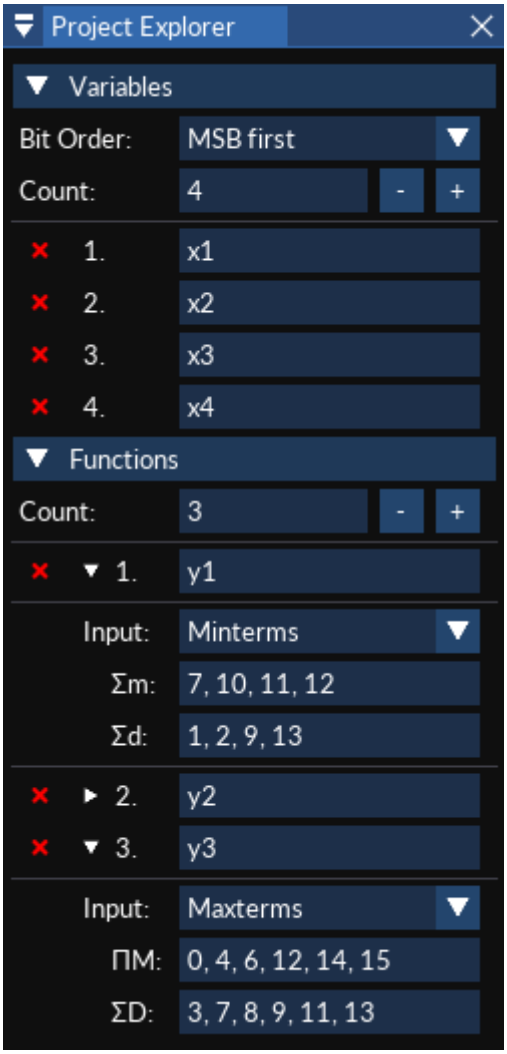
Obrázek 4.10: Výběr minimalizačního cíle v panelu nástrojů.

4.3 Okno Project Explorer

Jedním z nejpodstatnějších prvků uživatelského rozhraní je průzkumník projektu - okno *Project Explorer* (Obrázek 4.11). Ten představuje jeden z mála způsobů, jakými je možné nastavit nejzákladnější vlastnosti projektu, to znamená jeho proměnných a funkcí. Jedinou alternativou je možnost importovat obsah pravdivostní tabulky z libovolného souboru ve formátu *CSV*. Okno se dělí do dvou sekcí, které je možné pro zpřehlednění práce v něm individuálně skrýt.

Sekce *Variables*, jak již její název napovídá, slouží k úpravě vstupních proměnných. Nejprve se nabízí možnost změny pořadí bitů reprezentujících dílčí proměnné. Je možné volit mezi nejvýznamnějším a nejméně významným bitem na prvním místě, což má dopad na rozložení dat v Karnaughových mapách i na vizuální podobu pravdivostní tabulky. Další vlastností je počet proměnných. Ten je možné zadat přímo do textového pole nebo jej navýšit či snížit prostřednictvím tlačítek plus a mínus. Zvolený počet se vždy zarovná do rozsahu dvou až osmi proměnných, aby se předešlo neočekávanému a nesprávnému chování programu.

Pod oddělovačem se nachází seznam všech dostupných proměnných. Každá z nich je označená pořadovým číslem a reprezentována názvem, který je možné v odpovídajícím textovém poli měnit až do délky čtyřiadvaceti znaků. Vyjma změny názvu je možné proměnnou smazat kliknutím na křížek v levém rohu řádku. Červená barva křížku signalizuje možnost proměnnou smazat, šedá pak pravý opak, k čemuž dochází při dosažení minimálního počtu proměnných. Dále je možné proměnné jednoduše přesouvat tažením myši. Označitelnost řádku umožňuje mimo přesouvání i jeho zvýraznění v případě označení odpovídajícího prvku v jiném okně. To navyšuje přehlednost programu při práci s ním a umožňuje uživateli se v jeho obsahu snáze zorientovat.



Obrázek 4.11: Okno Project Explorer.

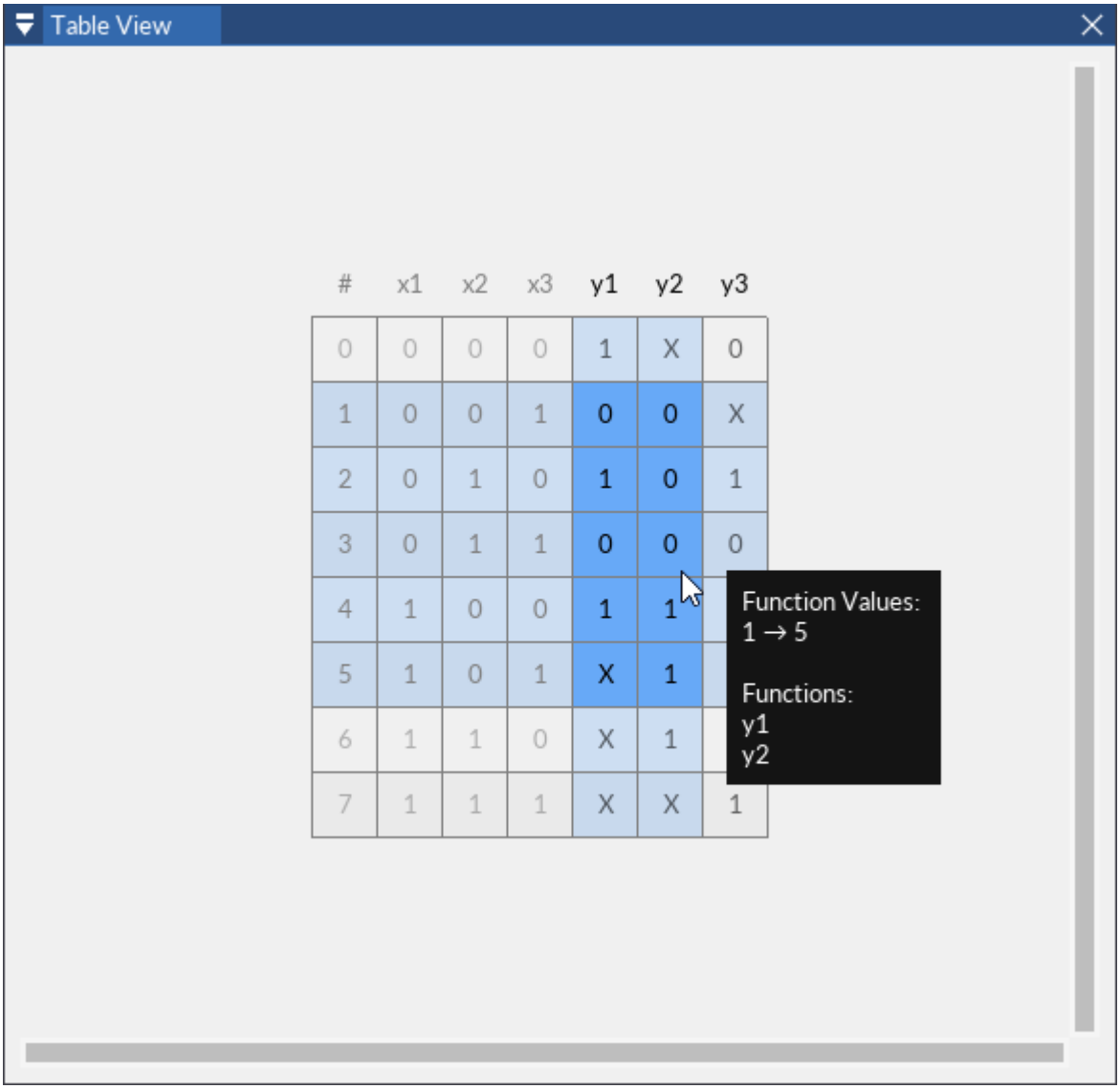
Druhá sekce se nazývá *Functions* a je strukturována obdobně jako sekce první. Jejím prostřednictvím je možné nastavit základní vlastnosti funkcí obsažených v projektu. Opět je možné nastavit počet prvků, tentokrát je však jeho hodnota zarovnána do rozmezí jedné až třiceti dvou funkcí, což ovšem není technickou limitací. Maximální počet funkcí je možné změnit na libovolnou hodnotu pouhým upravením konstanty ve zdrojovém kódu.

Každá funkce je realizována vlastním označitelným řádkem. V porovnání s proměnnými však funkce umožňují dodatečné rozkliknutí, kterým se zpřístupní možnost zadání funkčního vstupu. Uživatel má na výběr ze dvou textových variant. Buď je možné funkci zadat ve tvaru součtu mintermů, tedy indexů prvků obsahujících logickou jedničku, nebo jako součin indexů maxtermů - prvků obsahujících logickou nulu. V obou případech se nabízí i možnost doplnění sumy neurčitých stavů. Využití těchto vstupů pro zadání funkce není povinné, existuje možnost ručního vytvoření pravdivostní tabulky prostřednictvím okna *Table View*. Každá funkce může být zadána libovolným způsobem. Vstupní řetězce jsou

aktualizovány i při provádění změn pravdivostní tabulky. Je tedy možné je v případě potřeby z ručně vytvořené tabulky zpětně vyčíst.

4.4 Okno Table View

Význam okna *Table View* (Obrázek 4.12) spočívá ve vizuálním návrhu pravdivostní tabulky projektu. Představuje intuitivní způsob návrhu logických funkcí jako alternativu k jejich zadání v textové podobě nebo prostřednictvím importu. Na veškeré prováděné změny reaguje v reálném čase okno *Map View*, které na základě zadaných hodnot vyplňuje Karnaughovy mapy. Na případné smazání existujících minimalizačních smyček v důsledku provedených změn je uživatel upozorněn dialogovým oknem, jehož prostřednictvím může jejich smazání zamezit.

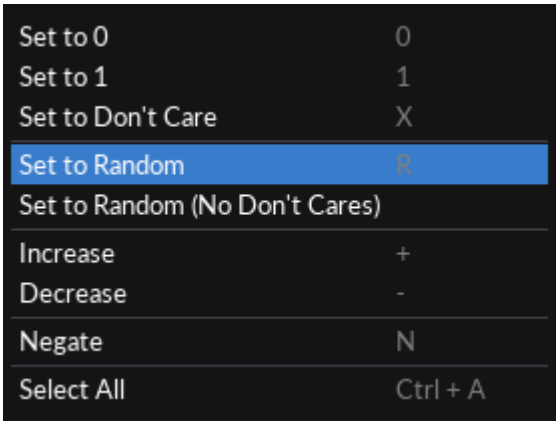


Obrázek 4.12: Okno Table View.

Pravdivostní tabulka se skládá ze tří hlavních částí, z nichž dvě mohou být skryty. Mezi ty patří číselné hodnoty, které jsou dány kombinací bitů proměnných na odpovídajícím řádku a proměnné samotné. Na prvním řádku tabulky se vždy nachází názvy prvků. V případě použití delšího názvu proměnné, než je šířka buňky, dojde k jeho otočení o devadesát stupňů v uživatelem zvoleném směru. První řádek je schopen se roztáhnout do výšky dle potřeby. Jedinými povinně zobrazovanými prvky jsou funkce, jelikož se právě kolem nich točí celá činnost okna.

Z důvodu maximalizace efektivity práce uživatele a jeho orientace ve tvořeném obsahu jsou některá okna uživatelského rozhraní vzájemně propojena. Okno *Table View* je jedním z nich. Najetím na libovolný prvek, ať už se jedná o proměnnou nebo funkci, dojde ke zvýraznění odpovídajícího prvku v okně *Project Explorer* a v případě výběru funkčních hodnot dojde ke zvýraznění zvolených buněk i v okně *Map View*. Mimo toho je při nájezdu myší na provedený výběr zobrazen popisek obsahující seznam označených funkcí a informace o zvoleném rozsahu funkčních hodnot. Význam má obzvláště při úpravě větších tabulek, které mohou dosahovat plochy čítající až 10 537 buněk. Tento popisek je pro snazší orientaci zobrazován i po celou dobu provádění výběru. Ten jde provést pouhým najetím na libovolnou buňku, označením více buněk najednou nebo kliknutím na název funkce, jejíž hodnoty mají být zvoleny.

Úpravy pravdivostní tabulky lze provádět při výběru jedné nebo více funkčních hodnot. Nabízí se řada způsobů, jakými se dá měnit hodnoty v rámci tohoto výběru. Tím nejrychlejším je navýšení funkční hodnoty o jedničku, čehož lze docílit kliknutím levého tlačítka myši. Posloupnost postupuje od logické nuly až po neurčitý stav, po kterém následuje vrácení se zpět do nuly. Dalším způsobem je využití kontextové nabídky, která čítá celkem devět položek. První tři slouží k nastavení na zvolenou hodnotu, ať už se jedná o logickou nulu, jedničku nebo neurčitý stav. Dvojice navazujících prvků umožňuje nastavení funkčních hodnot na zcela náhodnou hodnotu, ať už s využitím neurčitých stavů nebo bez nich. Dále se funkční hodnoty dají snížit nebo navýšit, kdy se provádí stejná činnost jako při kliknutí na levé tlačítko myši. Prvky je rovněž možné negovat, přičemž se neurčité stavy nemění. Jako poslední se nabízí možnost zvolení všech prvků pravdivostní tabulky. Většině položek je přiřazena klávesová zkratka, pomocí níž je možné odpovídající akci neprodleně vyvolat, aniž by bylo nutné otevřít kontextovou nabídku, jejíž podoba v rámci okna *Table View* je k vidění na obrázku (Obrázek 4.13).



Obrázek 4.13: Kontextová nabídka okna Table View.

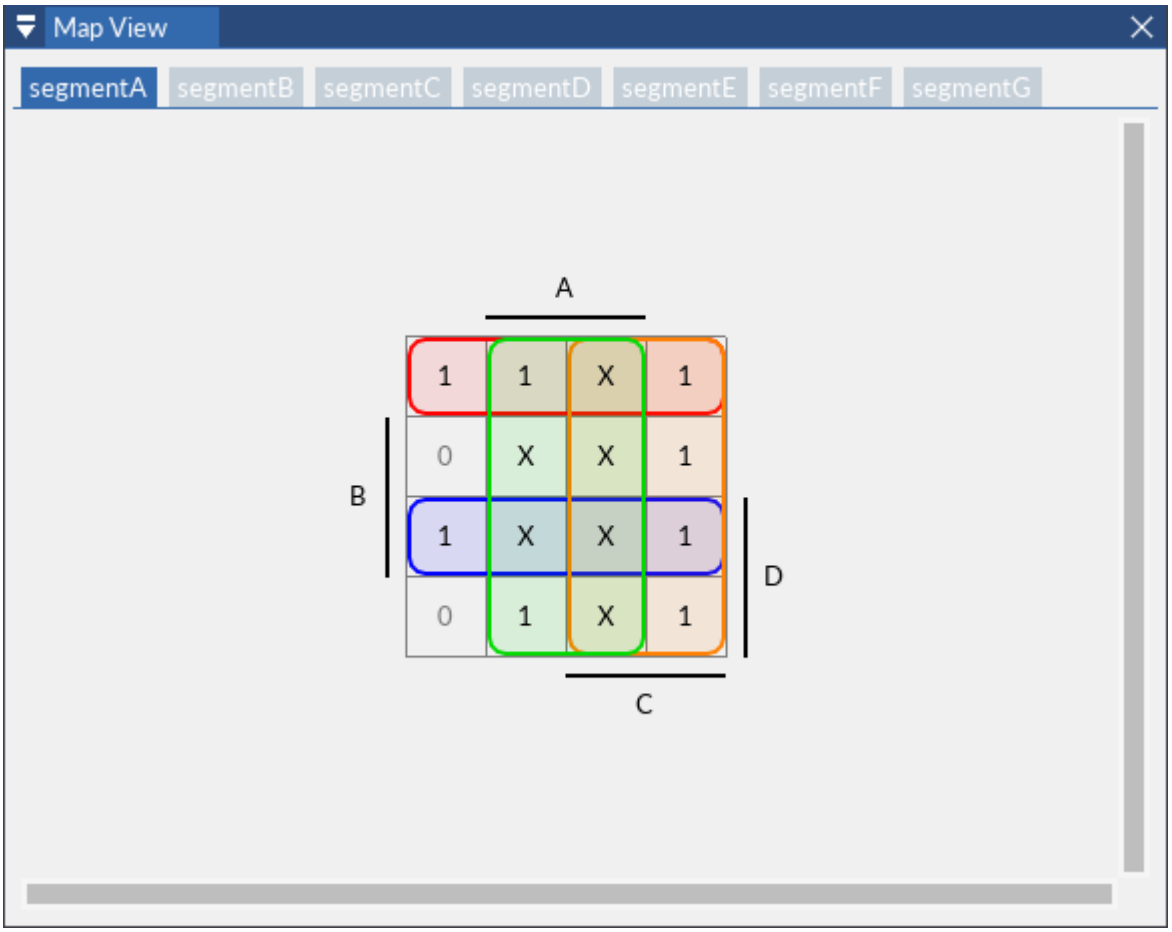
K pohybu napříč obsahem okna slouží buď běžně používané posuvníky, které jsou nepřetržitě zobrazovány v rozích okna, nebo intuitivní ovládání zprostředkovávané držením prostředního tlačítka myši a následným tažením v opačném směru, než je směr žádaného pohybu. Tento způsob umožňuje pohyb při provádění výběru. Vizuální podobu a rozložení pravdivostní tabulky je možné měnit prostřednictvím okna *Properties*. Veškeré dostupné úpravy jsou popsány v kapitole 4.8.

4.5 Okno Map View

Jediný způsob, jakým je možné provádět ruční návrh a úpravy minimalizačních smyček, představuje okno *Map View* (Obrázek 4.14). To se skládá z lišty záložek reprezentujících dílčí funkce projektu, pod níž se nachází samotný obsah v podobě Karnaughových map a odpovídajících minimalizačních smyček aktivní funkce. Ta odpovídá zvolené záložce okna *Map View* a ovlivňuje výsledek zobrazovaný prostřednictvím okna *Minimization Results* i několika dalších a je pro přehlednost zvýrazňována v okně *Project Explorer*.

Okno si klade za cíl rychlé a intuitivní provádění manuální minimalizace. Tvořit minimalizační smyčky je proto možné hned několika způsoby. Pomineme-li automatické vygenerování řešení, smyčku lze vytvořit dvojitým kliknutím na libovolné pole nebo označením více prvků zároveň. Vždy však musí platit, že alespoň jedna ze zahrnutých buněk nabývá logické jedničky (respektive nuly při tvorbě součinového tvaru funkce), zbytek již mohou tvořit neurčité stavy. Aplikace neumožňuje vytváření smyček, které obsahují neurčité stavy, neboť jejich existence nemá význam. V případě, že nelze vytvořit smyčku přesně odpovídající prováděnému výběru, vznikne smyčka s co nejideálnějším pokrytím prvků mapy uvnitř výběru. Podoba nové smyčky je přitom po celou dobu tažení viditelná. Po

dokončení výběru dochází k jejímu uložení na vrchol aktivního seznamu smyček a je jí automaticky přiřazena jedna z předem definované sady barev.



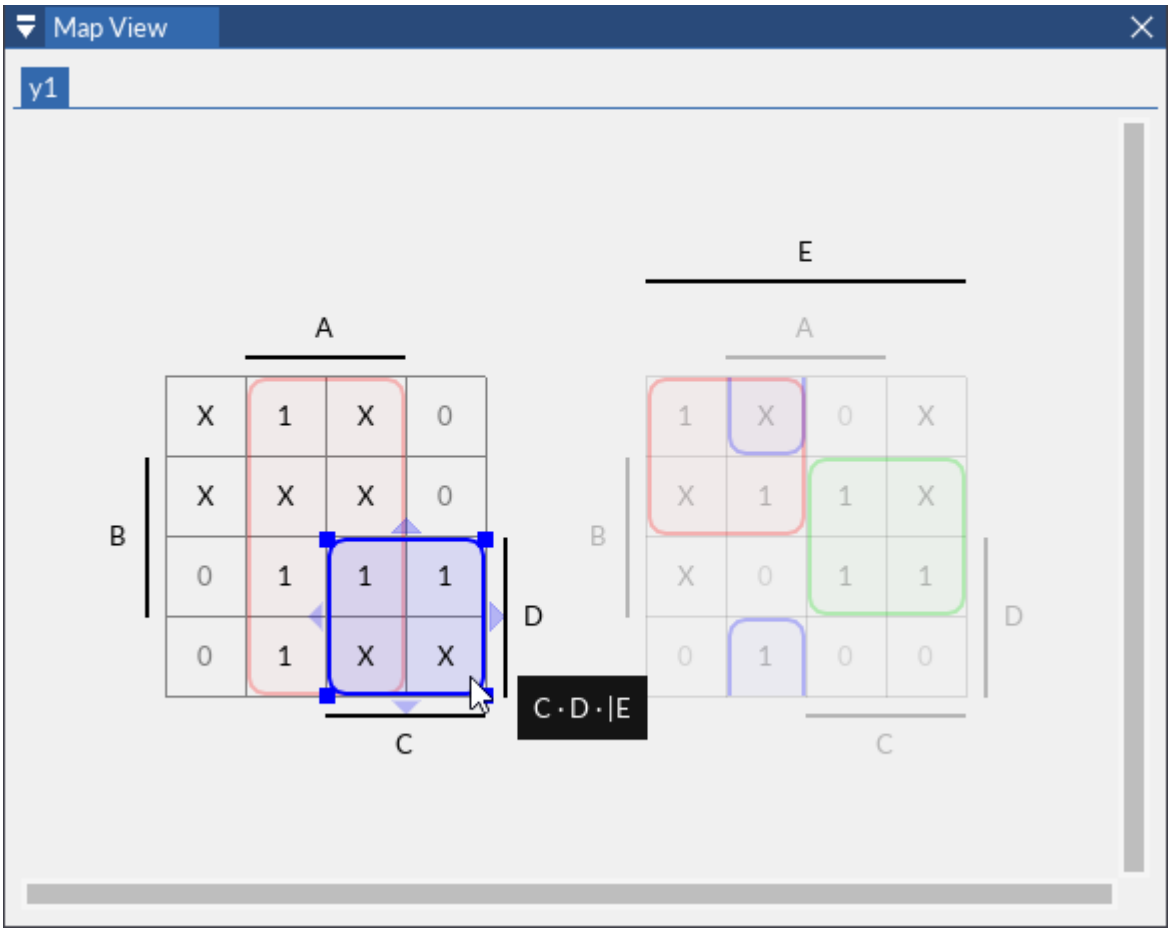
Obrázek 4.14: Okno Map View.

V rámci své činnosti okno nabízí i řadu pomocných prvků pro snazší orientaci v jeho obsahu. Uživatele se snaží nasměřovat k zahrnutelným prvkům jejich odlišným barevným zpracováním, které je možné libovolně měnit prostřednictvím okna *Properties*. Při samotném procesu vytváření nebo volbě smyčky dochází ke zprůhlednění všech ostatních smyček i map, aby se pozornost ubírala na právě volený prvek. Rovněž tak dochází ke zobrazení popisku, který obsahuje kompletní minterm smyčky, i jejímu zvýraznění v rámci okna *Groups*.

Stejně jako tvorba minimalizačních smyček je důležitá i jejich úprava, jejíž průběh je k vidění na obrázku (Obrázek 4.15). Při označení libovolné viditelné a neuzamčené smyčky se kolem ní objeví ovládací prvky, s jejichž pomocí je možné ji dále upravovat. Krom čtyř čtverců umístěných v rozích, které slouží čistě ke změně velikosti smyčky, jsou k dispozici i šipky uprostřed každé z jejích stran. Ty plní dvojí funkci, jednak slouží k automatickému rozšíření smyčky daným směrem a dále signalizují, že je tímto směrem

smyčku stále možné rozšířit. Alternativou je automatické rozšíření již existující smyčky do obou směrů zároveň dvojitým kliknutím na ni.

Minimalizační smyčky umožňují jejich intuitivní tažení přes hranu, které se pojí s možností vykreslovat smyčku zároveň na všech stranách mapy, na nichž se nachází. Ovládací prvky se v tomto případě objeví v místě posledního označení smyčky. Její označení je realizováno prostřednictvím bitových masek a díky tomu lze efektivně řešit označení přetékajících prvků.



Obrázek 4.15: Úprava minimalizační smyčky v okně Map View.

4.6 Okno Groups

Okno *Groups* tvoří přehled o všech existujících minimalizačních smyčkách právě upravované funkce a může se vyskytovat v několika různých režimech zobrazení. První z nich je zobrazován pouze v případě, že se ve funkci nenachází žádná minimalizační smyčka. Uživateli proto poskytuje základní pokyny k vytvoření první smyčky, načež se přepíná do běžného zobrazení. Tím se rozumí samotný přehled existujících minimalizačních smyček. Oba hlavní režimy jsou k vidění na obrázku (Obrázek 4.16).

Obrázek 4.16: Režimy zobrazení okna Groups.

V horní části okna je k dispozici aktuální počet smyček. V té dolní lze nalézt tlačítko pro smazání všech smyček najednou, což se při testování programu prokázalo být obzvlášť užitečné. Středová část okna je věnována samotným smyčkám a může nabývat dvou podob v závislosti na počtu proměnných a s tím souvisejícím počtu Karnaughových map. Při výskytu více než jedné mapy dochází k rozdělení okna do stejného počtu sekcí. Každá z nich je označena mintermem odpovídající mapy. Sekce je možné dle potřeby skrývat a pracovat jen s těmi, které jsou potřeba. Uvnitř každé z nich se nachází pouze ty minimalizační smyčky, které náleží dané mapě. V případě výskytu jediné mapy se jedná o všechny smyčky a ty již nejsou dále děleny.

Smyčka je tvořena označitelným řádkem nesoucí název tvaru jejího mintermu uvnitř mapy. Každý řádek nabízí celkem čtyři ovládací prvky. Pomocí nich je možné zvolenou smyčku smazat, zamknout, zneviditelnit či změnit její barvu. Neviditelné smyčky je možné označit pouze prostřednictvím tohoto okna a ve výsledku minimalizace se vůbec neprojeví, jako by neexistovaly. Uzamčením nebo zneviditelněním smyček je možné vytvářet překrývající se smyčky, aniž by došlo k nechtěnému označení některé z těch stávajících. Jejich seznam je vypisován ve stejném pořadí, v jakém probíhá vykreslování smyček v okně

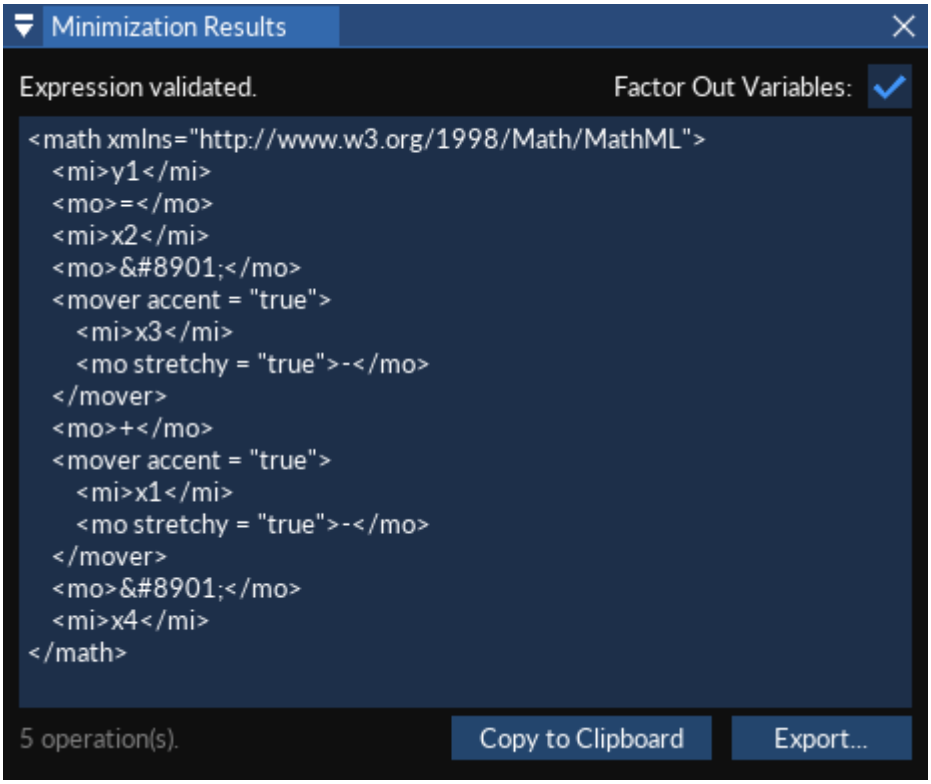
Map View. Znamená to tedy, že smyčka nejvíce nahoře je vykreslena jako první a ta nejníže až jako poslední, přesouváním řádků je možné toto pořadí měnit. To lze ovšem provést pouze v rámci sekce, pod níž spadají. Užitečnou vlastností okna je i zvýraznění označené smyčky v okně *Map View*, která se přenese do popředí a všechny ostatní se zprůhlední tak, aby bylo na první pohled patrné, o kterou smyčku se přesně jedná.

4.7 Okno Minimization Results

Prostřednictvím okna *Minimization Results* (Obrázek 4.17) je uživateli předkládána výsledná podoba minimalizované funkce ve zvoleném výstupním formátu. Ta je zobrazována použitím víceřádkového textového pole, které je nastaveno pouze pro čtení. V horní části okna je možné pozorovat stav validace vygenerovaného výrazu. Pokud je výraz neúplný nebo z jiného důvodu nesprávný, je na tento fakt uživatel upozorněn červeně zbarveným varovným textem. V opačném případě se na tomtéž místě nachází text potvrzující správnost výsledku.

Po boku validačního textu se nachází zaškrtávací pole umožňující měnit aktivitu vytýkání proměnných. Rovněž tak okno nabízí informaci o celkovém počtu operací včetně negace figurujících ve výsledném výrazu. To lze využít například k vyhodnocení efektivity řešení a případnému porovnání více řešení. Tohoto kritéria využívá i aplikace samotná při výběru nejlepší varianty funkce pro její export. Posledním prvkem okna je dvojice tlačítek nacházejících se v pravé dolním rohu. Patří mezi ně tlačítka pro rychlé zkopírování výsledku do schránky a otevření dialogového okna *Export Minimization Results* pro jejich export.

K aktualizaci výsledků dochází při každé provedené změně týkající se minimalizačních smyček nebo čehokoliv jiného, co s nimi souvisí. Z důvodu zamezení nežádoucího navýšení odezvy programu je celý proces aktualizace vykonáván vláknem na pozadí. To je důležité především kvůli validaci výsledků, která ve většině případů představuje neúměrnou výpočetní zátěž v porovnání se samotným generováním výsledků. Kód vlákna je připraven rychle a flexibilně reagovat na prováděné změny. Je schopný kdykoliv ukončit svou činnost a nahradit ji činností jinou.



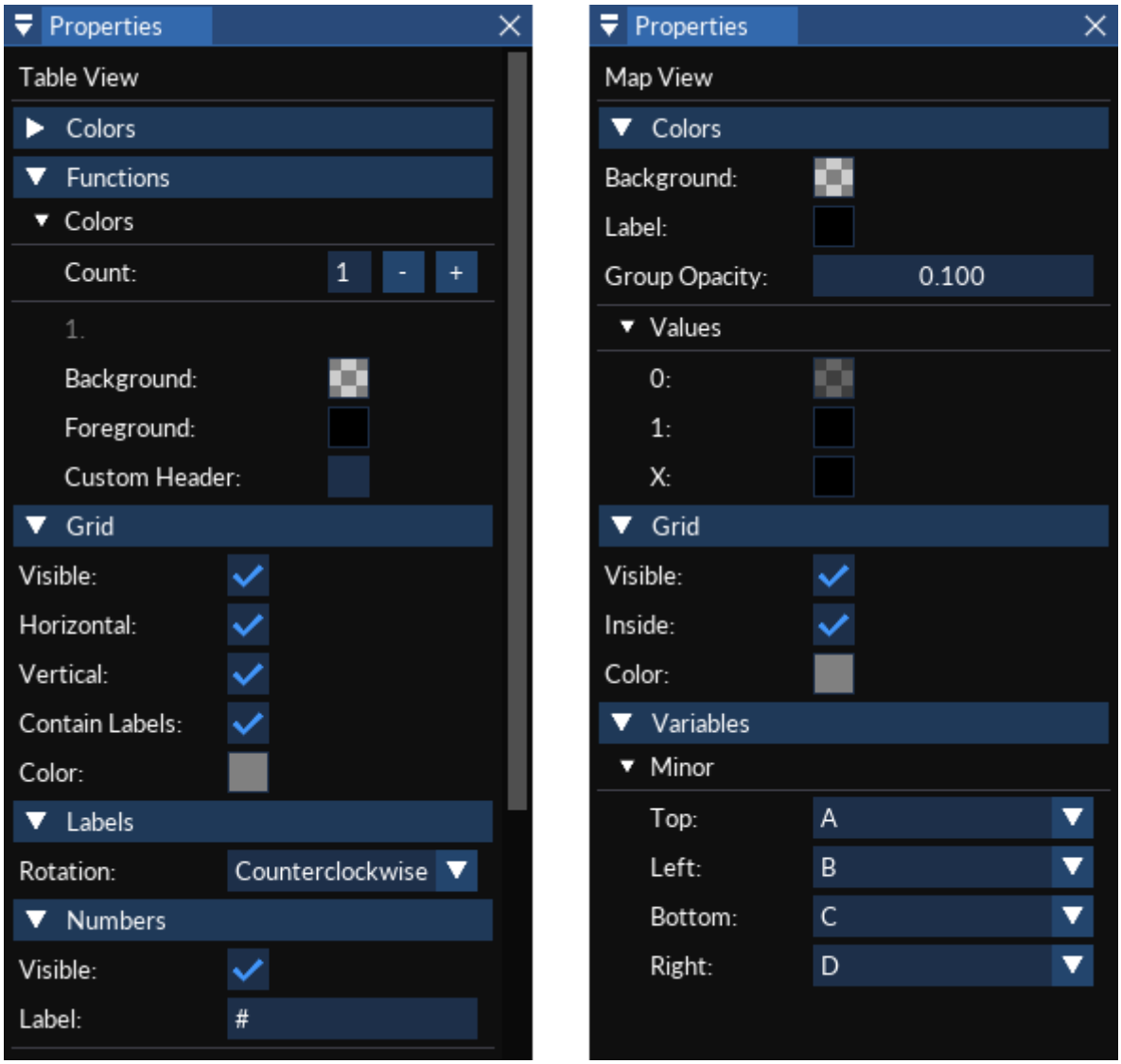
Obrázek 4.17: Okno Minimization Results.

4.8 Okno Properties

Okno *Properties* představuje způsob, jakým lze nastavovat nejrůznější vlastnosti prvků spadajících pod ostatní okna. Jeho obsah se mění v závislosti na posledním aktivním okně z trojice podporovaných - *Table View*, *Map View* a *Minimization Results*. Prováděné změny se neprodleně projevují na prvcích daného okna.

V případě vlastností okna *Table View* je nabízena řada úprav týkajících se struktury, rozložení i grafické podoby pravdivostní tabulky, jak lze vidět na levé straně obrázku (Obrázek 4.18). Dělí se do několika sekcí označujících dílčí prvky a vlastnosti tabulky. Konkrétně jde o sekce *Colors*, *Functions*, *Grid*, *Labels*, *Numbers* a *Variables*. Prostřednictvím té první je možné měnit barvu pozadí okna a barvy sudých a lichých řádků, které jsou vykreslovány nad barvami pozadí jednotlivých prvků. Významnější je však totožně nazvaná podsekce, která se nachází uvnitř většiny hlavních sekcí. Jednou z nich je *Functions*, která sestává pouze z této podsekce, pomocí které se dají nastavovat barvy jednotlivých funkcí. Jako první je k dispozici volba jejich počtu. Při hodnotě nižší, než je celkový počet funkcí, dochází k pravidelnému opakování barev. Dále se podsekce *Colors* dělí na odpovídající počet úseků, které představují dílčí skupiny barev. Každá skupina se z barvy pozadí a popředí a možnosti nastavit hlavičce prvku barvu vlastní. V případě

nevyužití tohoto nastavení se pro hlavičku použijí stejné barvy jako pro zbytku řádků. Uvnitř sekce *Grid* je k dispozici několik vlastností týkajících se vizuální podoby mřížky tabulky. Lze přepínat její horizontální, vertikální i celkové zobrazení, zahrnutí hlaviček prvků dovnitř ní nebo měnit její barvu. Sekce *Label* obsahuje jediný prvek, *Rotation*, pomocí něhož je možné nastavit směr, kterým budou otáčeny dlouhé názvy prvků v hlavičce. Poslední sekce *Numbers* a *Variables* zahrnují podsekci *Colors*, která má v jejich případě stejné chování jako v případě *Functions*. Dále obě sekce nabízí možnost přepínání jejich zobrazení v rámci tabulky, jelikož se nejedná o její povinné prvky. V případě *Numbers* je navíc možné měnit název zobrazovaný v tabulce.

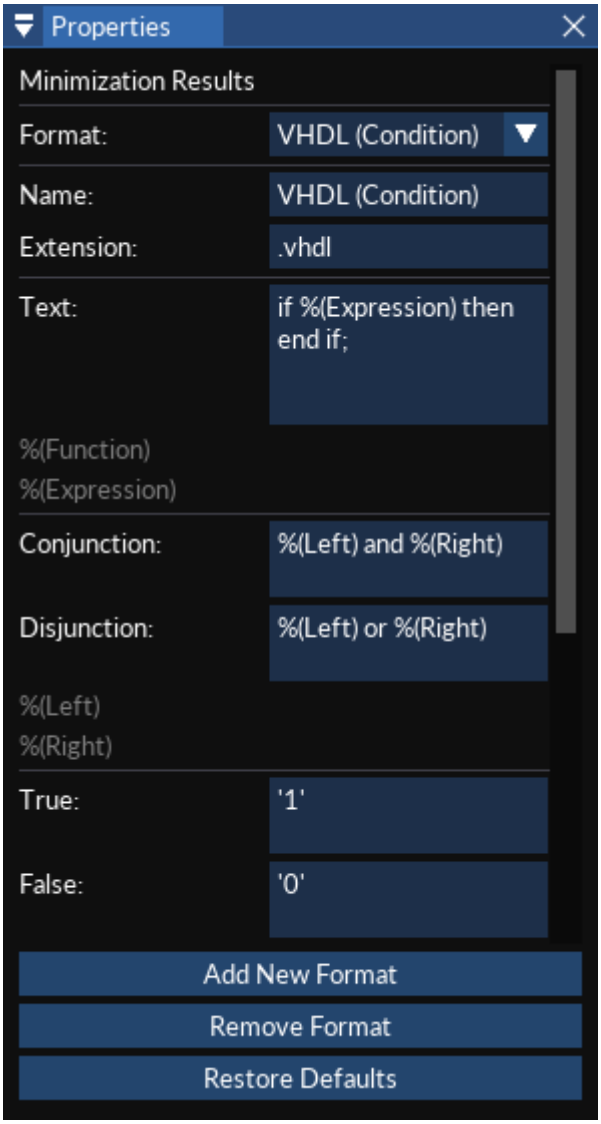


Obrázek 4.18: Okno Properties – vlastnosti oken Table View a Map View.

Vlastnosti okna *Map View* se od *Table View* liší především v množství nabízených možností, které je jen zlomkové, viz pravá strana obrázku (Obrázek 4.18). Nejpodstatnější

možností je nastavení pořadí proměnných, které jsou rozděleny do dvou sekcí - *Major* a *Minor*. Minoritními se rozumí proměnné nacházející se uvnitř každé Karnaughovy mapy, zatímco těmi majoritními vně proměnné. Na libovolné místo se dá dosadit libovolná proměnná. V závislosti na tom se upraví rozložení map, dojde k aktualizaci jejich hodnot a mohou být také smazány některé z existujících minimalizačních smyček. Na to je však uživatel upozorněn dialogovým oknem a může tomu předejít. Stejně jako v případě *Table View* je i zde možné upravovat vizuální podobu mřížky. Lze ji skrýt celou nebo jen její vnitřní část. Prostřednictvím sekce *Colors* lze krom barvy pozadí nastavit také barvy jednotlivých prvků zvlášť - logické jedničky, nuly i neurčitých stavů. Dále se nabízí změna průhlednosti vnitřní části minimalizačních smyček či barva popisků map.

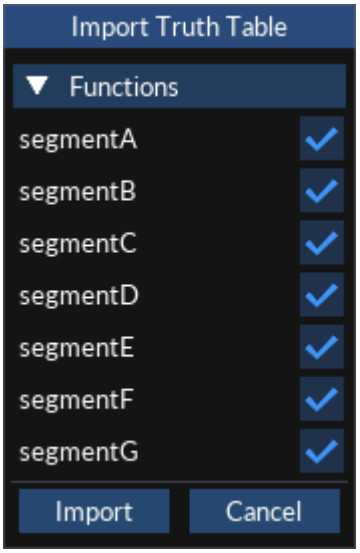
Nejvýznamnějším řežimem zobrazení jsou vlastnosti okna *Minimization Results*. V jejich případě je možné volit formát výstupního výrazu, přičemž ten poslední uživatelem zvolený je nastaven jako výchozí pro nové projekty. Dále lze upravit některý ze stávajících formátů, nebo vytvořit zcela nový. K tomu slouží trojice tlačítek nacházejících se ve spodní části okna. Stisknutím tlačítka *Add New Format* dojde k vytvoření nového formátu, který je možné pomocí zbytku prvků okna definovat. Tlačítko *Remove Format* slouží k odstranění libovolného uživatelem definovaného formátu a *Restore Defaults* umožňuje obnovit výchozí nastavení, což spočívá ve smazání všech formátů a následném obnovení těch výchozích. Hlavní část okna se skládá z řady textových vstupů, jejichž prostřednictvím lze upravit dílčí vlastnosti aktivního formátu. Krom nastavení základních prvků v podobě jeho názvu, výchozí koncovky souboru, textového tvaru konjunkce, disjunkce, rovnosti a dalších vlastností se dá ovlivnit míra, do jaké budou tvořené výrazy ohraničovány závorkami.



Obrázek 4.19: Okno Properties – vlastnosti okna Minimization Results.

4.9 Okno Import Truth Table

Jedním ze způsobů vyplnění pravdivostní tabulky je její import ze souboru ve formátu *CSV*. Ten je možné provést prostřednictvím dialogového okna *Import Truth Table* (Obrázek 4.20). Jeho otevření předchází volba souboru prostřednictvím systémového dialogu, k jehož zobrazení dojde po zvolení položek *Import* a *Truth Table* v hlavní nabídce programu. Jakmile uživatel soubor zvolí, dojde k pokusu o jeho načtení a následné zpracování. V případě špatně formátovaného nebo neexistujícího souboru dojde k upozornění uživatele na tento fakt prostřednictvím dodatečného dialogového okna. V opačném případě dochází k postupnému zpracování celého souboru znak po znaku.



Obrázek 4.20: Okno Import Truth Table.

Zpracování formátu *CSV* není nikterak složité, není proto nutné využívat externí knihovnu, která by spíše představovala zbytečnou zátěž. Největší výhodou vlastního řešení je možnost provádět i takové činnosti, které ve většině dostupných knihoven nejsou zcela obvyklé, například detekovat použitý oddělovač. Tím je značně navýšena míra kompatibility s externími nástroji, které formát *CSV* neinterpretují zcela správně a namísto čárky používají jako oddělovač například středník nebo tabulátor. Vlastní řešení umožňuje i průběžnou kontrolu validity dat a detekci přítomnosti základních prvků v podobě číselných hodnot a proměnných, které nutně nemusí být součástí zpracovávaného souboru.

Přítomnost číselných hodnot je detekována porovnáním hodnoty prvního sloupce právě načítaného řádku s jeho indexem, který musí odpovídat načtené hodnotě. Proměnné jsou detekovány porovnáním posloupnosti bitů, která musí odpovídat buď přímé nebo inverzní bitové reprezentaci indexu daného řádku. Pokud hodnoty bitů neodpovídají, budou proměnné považovány za neexistující a sloupce se namísto toho zpracují ve tvaru funkčních hodnot. Kód je v případě existence proměnných schopen dohledat i pořadí bitů. Povinnou součástí souboru jsou hlavičky obsahující názvy dílčích prvků a rovněž musí soubor obsahovat alespoň jednu funkci.

Po načtení se soubor očistí od přebytečných mezer a ostatních z hlediska zpracování bezvýznamných znaků a jeho obsah se rozdělí na jednotlivé řádky. Kód je mimo oddělovače položek schopen detekovat i oddělovač řádků, není tedy problém zpracovávat *CSV* soubory nezávisle na jejich původu. Zpracování souboru pak probíhá postupně řádek po řádku. Jako oddělovač prvků je označen první nalezený z trojice předdefinovaných oddělovačů, přičemž

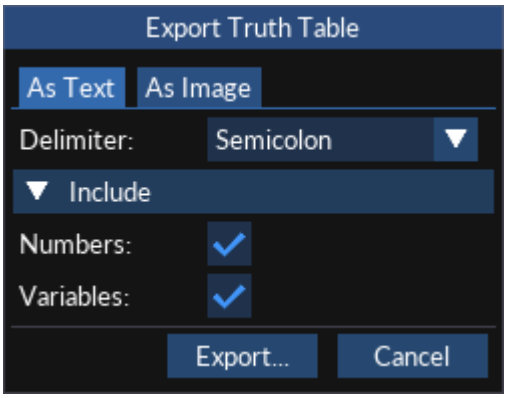
k jeho detekci nedochází uvnitř uvozovek. Cokoliv uvnitř uvozovek je považováno za čistý text a je tak možné mezi nimi použít libovolnou množinu znaků.

Všechna zpracovaná data jsou uložena do vnitřních struktur třídy *CSV*, která krom samotného zpracování souboru umožňuje i následný přístup ke zpracovaným datům. Na základě takto získaných dat je vyplněna pravdivostní tabulka a doplňující informace vztahující se k projektu. Až poté dojde k samotnému zobrazení okna *Import Truth Table*, které umožňuje dodatečně třídit funkce načtené ze souboru.

4.10 Okno Export Truth Table

Dialogové okno *Export Truth Table* slouží k provádění exportu pravdivostní tabulky a je možné jej otevřít pouze prostřednictvím hlavní nabídky programu. Pravdivostní tabulku lze exportovat do textového formátu *CSV* nebo do libovolného grafického formátu. Okno je rozdělené do dvou částí představujících dílčí typy exportu. Mezi nimi může uživatel přepínat pomocí záložek *As Image* a *As Text*.

Záložka *As Text* (Obrázek 4.21) umožňuje provést textový výstup a skrývá několik jeho základních nastavení. Nabízí se volba zahrnutí číselných hodnot a proměnných, které nutně nemusí být součástí výstupního souboru. Rovněž se dá změnit oddělovač, kterým budou oddělovány dílčí datové buňky, na jeden ze tří nejčastěji používaných – čárku, středník nebo tabulátor.

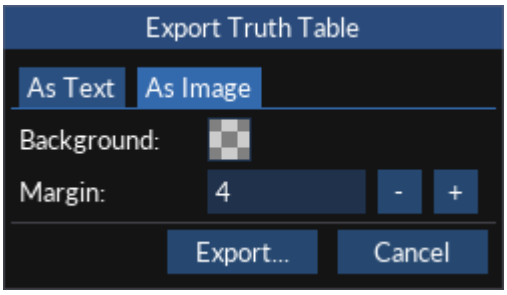


Obrázek 4.21: Okno Export Truth Table – záložka As Text.

Struktura souboru v textovém formátu *CSV* je triviální, k jeho vytvoření stačí využít pouhé konkatenace řetězců, pomocí níž se dílčí části výsledku spojí v jeden celek. Nejprve je nutné vyplnit hlavičku souboru nacházející se na jeho prvním řádku. Díky její přítomnosti je při zpracování souboru možné rozpoznat názvy všech prvků v tabulce zahrnutých. Postupně se názvy jednotlivých prvků zapíší v uvozovkách za sebe a oddělí se zvoleným

oddělovačem. Hlavní část tabulky tvoří číselná data, která jsou načítána přímo z pravdivostní tabulky projektu. Ta jsou postupně převáděna do textové podoby a vkládána do výstupního řetězce. Posledním krokem je uložení nashromážděných dat do souboru, který si uživatel zvolil prostřednictvím systémového dialogu po stisknutí tlačítka *Export*.

Druhá záložka okna nese název *As Image* (Obrázek 4.22) a slouží, jak již její název napovídá, k exportování grafické podoby pravdivostní tabulky do obrázkového souboru. V tomto případě je možné nastavit barvu pozadí výstupního obrázku a šířku jeho neviditelného okraje. Pokud je zvolena průhledná barva pozadí a zvolený grafický formát průhlednost nepodporuje, je využita její neprůhledná varianta, aby nedošlo k neočekávaným výsledkům. Výchozí barvou pozadí je barva nastavená prostřednictvím okna *Properties*, které je zdrojem veškerých nastavení ovlivňující vizuální podobu exportované tabulky. Šířka neviditelného okraje se dá nastavit skrze položku *Margin*, a to v rozmezí nula až sto pixelů. Okraj se aplikuje na všechny strany výstupního obrázku, jelikož není vždy žádoucí mít přesně ořezaný výstup.



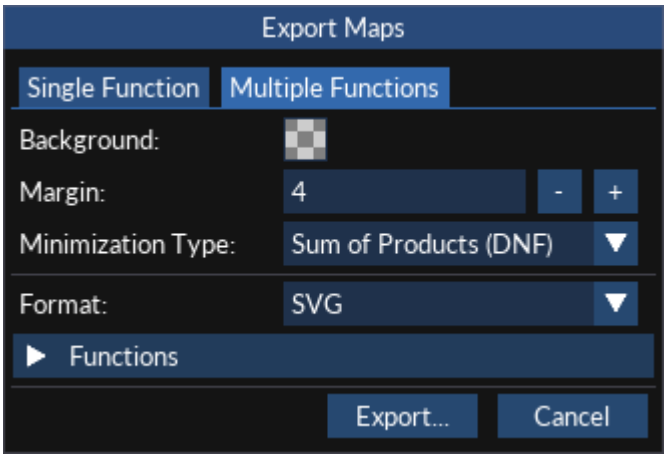
Obrázek 4.22: Okno Export Truth Table – záložka As Image.

Kliknutím na tlačítko *Export* dojde k otevření systémového dialogu pro výběr umístění exportovaného souboru. Jeho prostřednictvím lze zvolit požadovaný grafický formát, na jehož základě se má export provést. Uživatel může volit mezi nejrozšířenějšími rastrovými formáty *PNG*, *JPEG*, *TIFF* a *BMP*, jejichž tvorba probíhá využitím knihovny *Magick++*, a vektorovým formátem *SVG*.

Export jako takový probíhá volbou vhodného vykreslovacího jádra v závislosti na zvoleném výstupním formátu. To je použito při volání hlavní vykreslovací funkce okna *Table View* namísto využití jádra výchozího, které slouží pouze k vykreslování uvnitř uživatelského rozhraní. Výsledek je nakonec uložen do uživatelem zvoleného souboru.

4.11 Okno Export Maps

Smyslem dialogového okna *Export Maps* je jeho schopnost exportovat vizuální podobu Karnaughových map navržených prostřednictvím okna *Map View*. Tvoří jej záložky *Single Function* a *Multiple Functions*, které slouží k exportu map jedné nebo více funkcí naráz. Výslednou podobu okna lze vidět na obrázku (Obrázek 4.23).

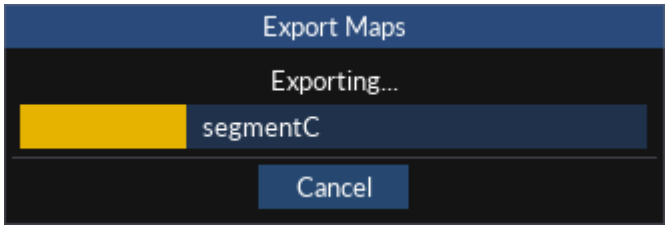


Obrázek 4.23: Okno Export Maps.

V případě obou záložek je možné měnit barvu pozadí i tloušťku neviditelného okraje, stejně jako je tomu v případě exportu zprostředkovávaného oknem *Export Truth Table*. Následuje volba typu minimalizace ovlivňující sadu minimalizačních smyček, které se objeví na výstupu. Posledním společným nastavením je možnost zvolit funkci, respektive funkce, které budou exportovány. V případě exportu více funkcí je také možné zvolit souborový formát, na jehož základě má být výstup generován. Při exportu jedné funkce je tento formát volitelný až prostřednictvím systémového okna pro uložení souboru, k jehož otevření dojde po stisknutí tlačítka *Export*. Toto tlačítko při exportu více než jedné funkce zobrazí okno pro výběr cílové složky, jelikož je nutné výstup každé z funkcí uložit do vlastního souboru, jehož název sestává z názvu odpovídající funkce. Skrze něj již není možné vybírat cílový souborový formát.

O průběhu exportu je uživatel informován prostřednictvím dialogového okna zobrazujícího aktuální průběh exportu a název právě exportovaného souboru, jak lze vidět na obrázku (Obrázek 4.24). Doba exportu může činit jen pár sekund nebo i několik minut. Vše se odvíjí od zvoleného souborového formátu a počtu exportovaných funkcí, Karnaughových map i smyček v nich se nacházejících. Probíhající export je možné v případě potřeby kdykoliv ukončit tlačítkem *Cancel*.

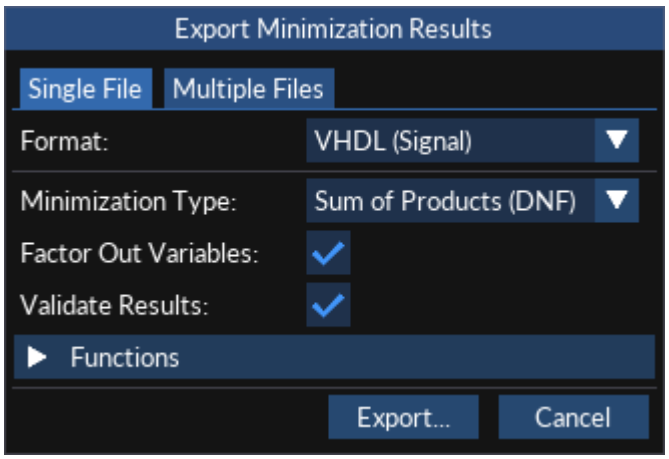
Samotný export probíhá stejným způsobem jako export grafické podoby pravdivostní tabulky. Dochází k volání hlavní vykreslovací funkce okna *Map View* použitím vykreslovacího jádra zvoleného na základě volby výstupního formátu. Export do vektorového formátu *SVG* má tu výhodu, že se jedná o pouhý textový soubor, o jehož vykreslení se starají až externí aplikace. Jeho vytvoření je tudíž velmi rychlé, obzvlášť v porovnání s generováním rastrových obrázků.



Obrázek 4.24: Okno Export Maps – průběh exportu.

4.12 Okno Export Minimization Results

K vygenerování a následnému uložení výsledků minimalizace do jednoho či více souborů slouží okno *Export Minimization Results*, které je k vidění na obrázku (Obrázek 4.25). Uživatel má možnost výběru libovolného výstupního formátu, na jehož základě budou výsledky generovány. Následuje volba typu minimalizace umožňující výběr zdroje minimalizačních smyček, z nichž se výsledky sestaví. Krom obou běžně nabízených typů je k dispozici i položka *Best*, která označuje automaticky prováděnou volbu nejlepšího z výsledků v rámci právě exportované funkce. Tím nejlepším se rozumí úplný výraz čítající nejméně operací z obou nabízených výsledků.



Obrázek 4.25: Okno Export Minimization Results.

Dalším prvkem v nabídce je možnost deaktivace dodatečného vytýkání proměnných a validace výsledků. Ta představuje časově náročný proces, díky kterému je ovšem možné

uživatele upozornit na případnou nesprávnost výsledků. K tomu dochází pomocí dodatečného dialogového okna, v němž jsou zobrazeny názvy funkcí, jejichž zminimalizovaných tvarů se nesprávnost či neúplnost týká. Jako poslední věc lze zvolit funkce, které budou do výstupu zahrnuty. Výběr však musí čítat alespoň jednu funkci.

Při exportu výsledných funkčních tvarů do více souborů může uživatel nastavit jejich libovolnou koncovku včetně případného doplnění názvu, který sestává z názvů dané funkce. Každá z nich je exportována do vlastního souboru. Při exportu do jednoho souboru dochází k uložení výsledků oddělených volným řádkem pod sebou. Stejně jako při exportu Karnaughových map je i v tomto případě uživatel informován o jeho průběhu prostřednictvím dodatečného okna a je možné celý proces kdykoliv ukončit.

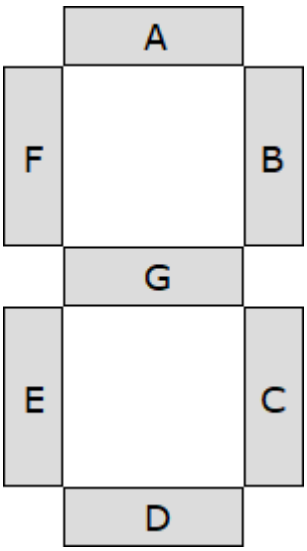
Proces exportu každé ze zvolených funkcí začíná vygenerováním výrazu v požadovaném formátu na základě minimalizačních smyček zvoleného typu minimalizace. V případě povolené validace výsledků pokračuje generováním validačního řetězce, na jehož základě proběhne samotná validace. Při volbě typu *Best* dochází k opakování tohoto procesu pro oba typy minimalizace. Na základě získaného počtu operací v obou výrazech pak dochází k volbě toho vhodnějšího k exportu. Posledním krokem je samotné uložení výrazů do jednoho či více souborů.

5 OVĚŘENÍ FUNKČNOSTI PROGRAMU A JEHO VÝSTUPŮ

Vzhledem k důkladnému testování aplikace v průběhu celého jejího vývoje byl pro ověření její funkčnosti a výstupů zvolen jeden konkrétní případ. Ten je rozvedený do detailů v kapitole 5.1 a pokrývá samotné použití aplikace i ověření jejích schopností grafického a textového exportu.

5.1 Sedmi-segmentový displej

Pod pojmem sedmi-segmentový displej se skrývá elektronické zobrazovací zařízení, skládající se ze sedmi či osmi individuálních segmentů (Obrázek 5.1). Ty bývají nejčastěji realizovány elektroluminiscenčními diodami. Největší význam těchto displejů spočívá ve schopnosti zobrazovat čísla, díky čemuž našli praktické využití v řadě produktů, například v digitálních hodinách, budících nebo palubních počítačích uvnitř automobilů. Zobrazovaný znak je závislý na kombinaci stavů dílčích segmentů. Každý z nich buď svítí, nebo nesvítí. [17]



Obrázek 5.1: Rozložení sedmi-segmentového displeje.

Tento příklad si klade za cíl návrh a následnou minimalizaci sedmi funkcí, reprezentujících dílčí segmenty simulovaného sedmi-segmentového displeje, použitelných ke zobrazení číselných hodnot nula až devět. K tomu je zapotřebí využít čtyři vstupní proměnné, jejichž prostřednictvím se zobrazovaná hodnota mění. Na číselných hodnotách deset až patnáct je možné demonstrovat využití neurčitých stavů, což v praxi znamená, že jejich zobrazení není definované.

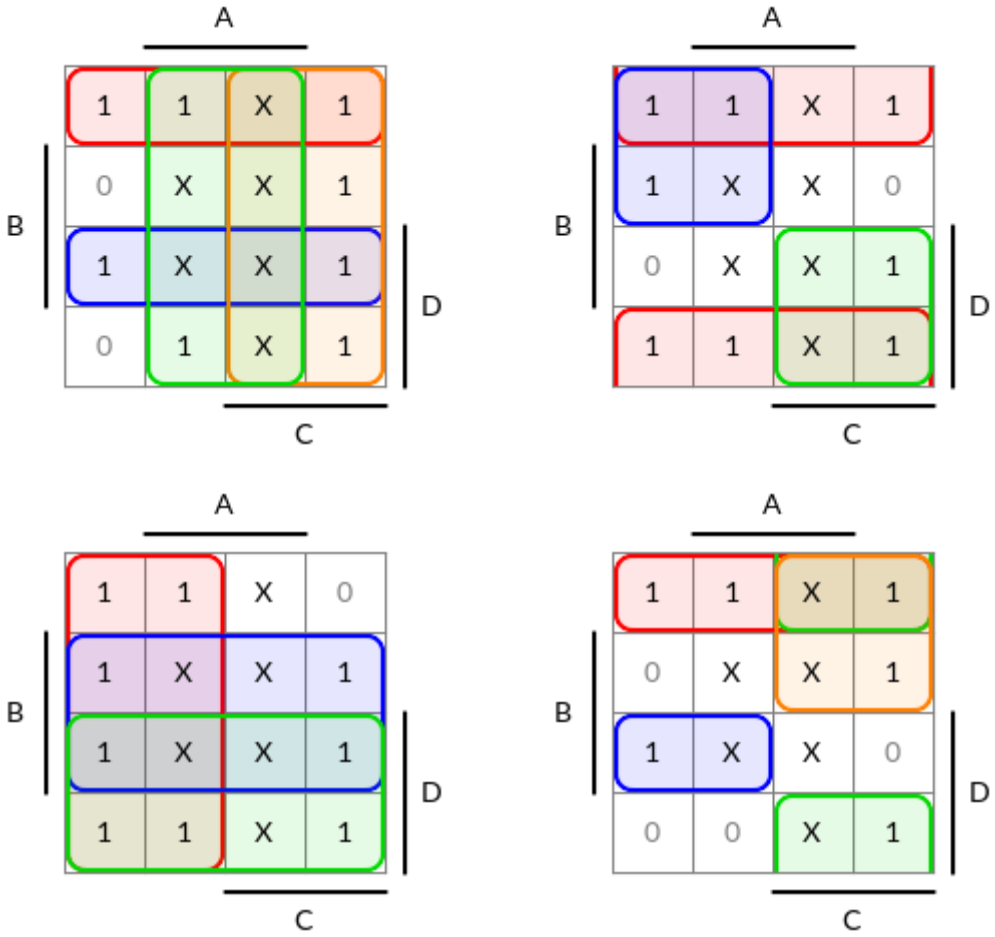
Pro ověření funkčnosti aplikace a jejího použití v reálném světe byl pro spuštění vygenerovaného zdrojového kódu zvolen simulovaný programovatelný automat ve vývojovém prostředí *Mosaic* od české společnosti *TECO*. Jako formát výsledků minimalizace poslouží programovací jazyk *Structured Text*, v němž se tyto systémy běžně programují. Exportované výstupy programu tedy lze beze změny zkopírovat a přímo vložit do předem připraveného zdrojového kódu k simulaci zobrazení sedmi-segmentového displeje.

Nejprve je nutné navrhnout logickou funkci pro každý z dílčích segmentů. Při výskytu logické jedničky daný segment svítí, v opačném případě nikoliv. Po spuštění aplikace *Karnaugh Studio* se vytvoří nový projekt a nastaví se v něm čtyři vstupní proměnné a celkem sedm funkcí, jímž se přiřadí názvy odpovídající označení segmentů. Pravdivostní tabulka se vyplní dle potřeby zobrazení číselných hodnot nula až devět, zbytek prvků je označen jako neurčitý stav. K návrhu dojde prostřednictvím okna *Table View*. Navrženou pravdivostní tabulku je možné vidět v podobě obrázku exportovaného programem (Obrázek 5.2).

#	A	B	C	D	segmentA	segmentB	segmentC	segmentD	segmentE	segmentF	segmentG
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Obrázek 5.2: Pravdivostní tabulka sedmi-segmentového displeje.

Nejpodstatnějším krokem je provedení samotné minimalizace, aby bylo možné získat výstupní zdrojový kód. Nejprve se nastaví formát výsledků minimalizace na formát *Structured Text (Statement)*. Následně dojde jediným kliknutím na tlačítko *Minimize All* v panelu nástrojů k vygenerování řešení všech funkcí naráz. Pro otestování grafického exportu Karnaughových map došlo k exportu map prvních čtyř funkcí *segmentA* až *segmentD*, jejichž podobu je možné vidět na obrázku (Obrázek 5.3).



Obrázek 5.3: Řešení Karnaughových map funkcí segmentA až segmentD.

Výsledky minimalizace se nyní exportují do souboru, což je možné provést stisknutím tlačítka *Export* v pravém dolním rohu okna *Minimization Results*, které otevře podpůrné dialogové okno. Pomocí něj je proveden export výsledků do jednoho souboru. Vytýkání proměnných je povoleno spolu s ověřením výsledků. V případě vygenerování nesprávného řešení minimalizace by tak aplikace vypsala chybové hlášení se seznamem nesprávně vyřešených funkcí, což by poukazovalo na chybné chování programu. K tomu ovšem v tomto příkladu nedošlo a export proběhl bez jediné chyby. Mělo by tedy být možné vygenerovaný zdrojový kód přímo použít.

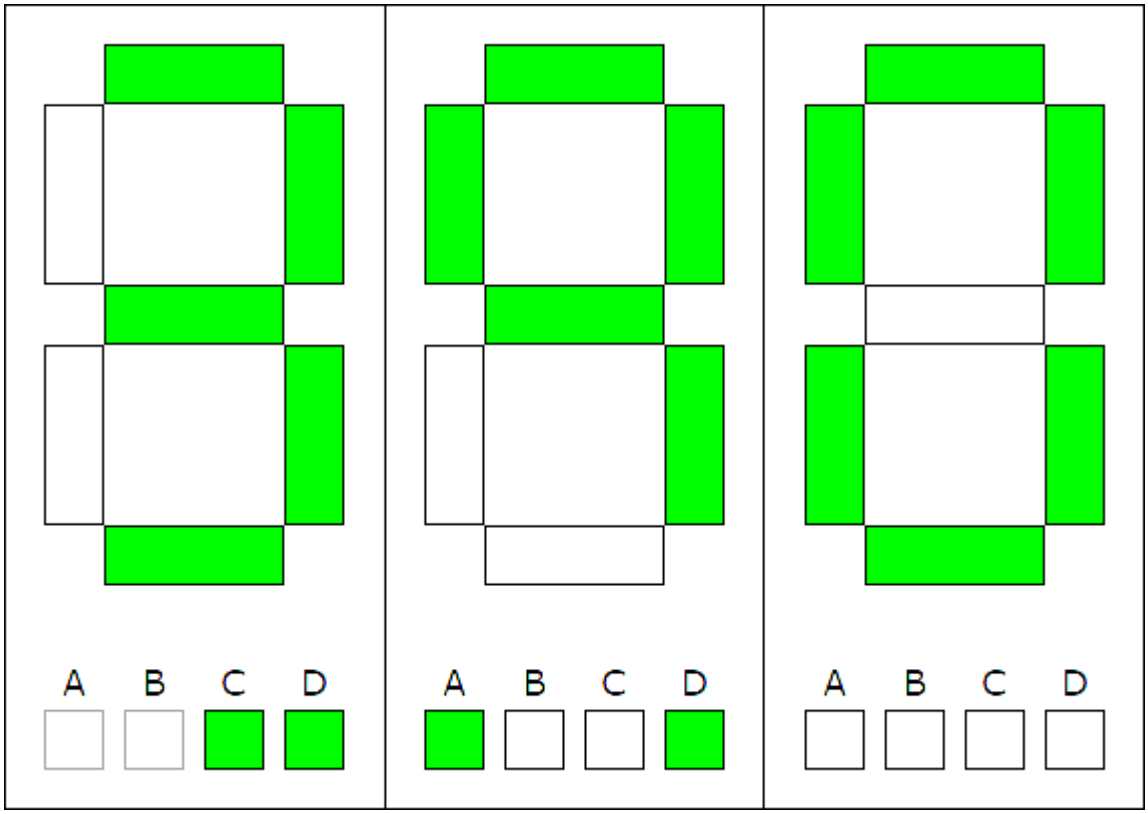
Uvnitř prostředí *Mosaic* je využitím programovacího jazyka *Structured Text* a vestavěného nástroje *Web Maker* navrženo jednoduché rozhraní pro zobrazení sedmi-segmentového displeje. Součástí tohoto rozhraní je sada globálních proměnných, jejichž názvy odpovídají názvům vyexportovaných logických funkcí. S jejich stavy bylo v prostředí *Web Maker* propojeno celkem sedm zelených obdélníků, které jsou zobrazovány pouze v případě, že daná proměnná nabývá hodnoty logické jedničky. Nyní už jen zbývá vložit do

hlavní části zdrojového kódu kód vyexportovaný aplikací, čímž vznikne výsledek na obrázku (Obrázek 5.4).

```
var_global

    A : bool := false;
    B : bool := false;
    C : bool := false;
    D : bool := false;

    segmentA : bool;
    segmentB : bool;
    segmentC : bool;
    segmentD : bool;
    segmentE : bool;
    segmentF : bool;
    segmentG : bool;

end_var

program main

    segmentA := A or C or (B and D) or (not B and not D);
    segmentB := (C and D) or (not C and not D) or not B;
    segmentC := D or B or not C;
    segmentD := (not B and (C or not D)) or (C and not D)
                or (B and not C and D);
    segmentE := not D and (C or not B);
    segmentF := (B and (not C or not D)) or A or (not C and not D);
    segmentG := (B and (not C or not D)) or (not B and C) or A;

end_program
```

Obrázek 5.4: Kompletní zdrojový kód příkladu v jazyce Structured Text.

Za předpokladu, že byl každý dosud provedený krok správný, by se měla v prostředí simulace zobrazit číselná hodnota odpovídající kombinaci bitů vstupních proměnných. K tomu skutečně došlo, jak je možné se přesvědčit z trojice vybraných případů spojených do jediného obrázku (Obrázek 5.5).



Obrázek 5.5: Simulované zobrazení sedmi-segmentového displeje.

Tímto příkladem bylo dokázáno reálné využití aplikace přímo v praxi. S tím souvisí fakt, že umožňuje nejen velmi snadné provádění návrhu a minimalizace logických funkcí, ale také generování výstupů v podobě zdrojového kódu v průmyslově používaném programovacím jazyce, který bylo možné bez dalších úprav použít. Byla rovněž ověřena funkčnost všech aplikací nabízených forem exportu. Výsledky lze ve všech případech považovat za velmi uspokojivé. Zdrojový kód příkladu by bylo možné jednoduše propojit s reálným sedmi-segmentovým displejem za předpokladu fyzického přístupu k němu.

ZÁVĚR

Výsledkem mé bakalářské práce je desktopová aplikace nazvaná *Karnaugh Studio*, která spojuje funkčnost grafického editoru s obecným nástrojem pro návrh a minimalizaci logických funkcí. Podařilo se mi splnit každý z bodů zadání a dosáhnout všech stanovených cílů.

Vypracování projektu bylo časově náročné, a to především kvůli četnému výskytu komplikací spojených s mou neznalostí použité knihovny *Dear ImGui*, se kterou jsem se v průběhu vývoje aplikace seznamoval, dále nedostatkem její dokumentace a omezenou nabídkou prvků v porovnání s ostatními knihovnami pro tvorbu grafických uživatelských rozhraní. Na její jádro jsem byl nucen nadstavit několik dodatečných vlastností, které jsou v případě její konkurence samozřejmostí. Její volby však i přes zmíněné nedostatky nelituji a věřím, že se jedná o velmi schopný nástroj. Využitím všech zvolených technologií jsem navíc získal cenné zkušenosti.

Na praktickém příkladu návrhu a realizace zobrazení sedmi-segmentového displeje jsem ověřil funkčnost vyvinuté aplikace a schopnost jejího začlenění do vývojového procesu. Rovněž jsem ověřil veškeré její schopnosti grafického i textového exportu, kterých jsem využil i při vypracování úvodních částí práce.

Naskýtá se hned několik možností, jak aplikaci dále rozvinout. Do budoucna vidím značný potenciál ve schopnosti zpracování vstupního výrazu v libovolném textovém formátu, vykonávání automatické minimalizace v reálném čase při provádění úprav pravdivostní tabulky, i ve zobrazení výsledku minimalizace ve formě logického obvodu s využitím hradel a možností jeho grafického exportu. Tyto vize by umožnily širší začlenění aplikace do praxe.

SEZNAM POUŽITÉ LITERATURY

[1] HOLDSWORTH, Brian a Clive WOODS. *Digital Logic Design*. 4th ed. Boston: Newnes, 2002. ISBN 978-075-0645-829.

[2] KUPHALDT, Tony R. *Vol. IV - Digital - Electronics Textbook* [online]. EETech Media [cit. 2019-04-28]. Dostupné z: https://www.allaboutcircuits.com/textbook/digital/

[3] HORKÝ, Jan et al. *ELUC* [online]. [2015] [cit. 2019-04-29]. Dostupné z: https://eluc.kr-olomoucky.cz/verejne/ucebnice/22/lekce

[4] PRATA, Stephen. *Mistrovství v C++*. 4. aktualiz. vyd. Brno: Computer Press, 2013. Bestseller (Computer Press). ISBN 978-802-5138-281.

[5] *The C++ Resources Network* [online]. c2000-2019 [cit. 2019-04-29]. Dostupné z: http://www.cplusplus.com/

[6] *JSON* [online]. [cit. 2019-04-29]. Dostupné z: https://www.json.org/

[7] Extensible Markup Language (XML) 1.0 (Fifth Edition). *World Wide Web Consortium (W3C)* [online]. W3C, c2019, 26 November 2008 [cit. 2019-04-28]. Dostupné z: https://www.w3.org/TR/xml/

[8] RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files. *IETF Tools* [online]. IETF Tools Team, October 2005 [cit. 2019-04-26]. Dostupné z: https://tools.ietf.org/html/rfc4180

[9] *SVG: Scalable Vector Graphics | MDN* [online]. Mozilla, c2005-2019 [cit. 2019-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/SVG

[10] *Qt | Cross-platform software development for embedded & desktop* [online]. The Qt Company, c2019 [cit. 2019-04-29]. Dostupné z: https://www.qt.io/

[11] BRAVENEC, Petr. Knihovna Qt. *Hobrasoft* [online]. 4. 4. 2016 [cit. 2019-04-29]. Dostupné z: https://www.hobrasoft.cz/cs/blog/bravenec/qt

[12] *WxWidgets: Cross-Platform GUI Library* [online]. wxWidgets, c2019 [cit. 2019-04-29]. Dostupné z: https://www.wxwidgets.org/

[13] CORNUT, Omar. Dear ImGui: Bloat-free Immediate Mode Graphical User Interface for C++ with minimal dependencies. *GitHub* [online]. GitHub, c2019 [cit. 2019-04-26]. Dostupné z: https://github.com/ocornut/imgui

[14] NORNEBY, Johannes. Immediate Mode Model/View/Controller. *Johno.se - software & music* [online]. [2013] [cit. 2019-04-29]. Dostupné z: http://www.johno.se/book/imgui.html

[15] *OpenCV* [online]. OpenCV team, c2019 [cit. 2019-04-29]. Dostupné z: https://opencv.org/

[16] *ImageMagick - Convert, Edit, or Compose Bitmap Images* [online]. ImageMagick Studio, c1999-2019 [cit. 2019-04-29]. Dostupné z: https://imagemagick.org/

[17] What is a 7 Segment Display?. *Learning about Electronics* [online]. c2018 [cit. 2019-04-26]. Dostupné z: http://www.learningaboutelectronics.com/Articles/7-segment-display

[18] HORKÝ, Jan et al. Minimalizace logických funkcí. *ELUC* [online]. [2015] [cit. 2019-04-30]. Dostupné z: https://eluc.kr-olomoucky.cz/verejne/lekce/933

[19] ZILLER, Eike. Qt Creator 4.3.0 released. In: *Qt | Cross-platform software development for embedded & desktop* [online]. The Qt Company, c2019, May 24th, 2017 [cit. 2019-04-30]. Dostupné z: https://blog.qt.io/blog/2017/05/24/qt-creator-4-3-0-released/

[20] Audacity 2.2.0 in default Light theme running on Windows 10. In: *Audacity ® | Free, open source, cross-platform audio software for multi-track recording and editing.* [online]. Audacity, c2019 [cit. 2019-04-30]. Dostupné z: https://www.audacityteam.org/about/screenshots/

[21] Dear ImGui: Custom engine. In: *GitHub* [online]. GitHub, c2019 [cit. 2019-04-30]. Dostupné z: https://github.com/ocornut/imgui

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

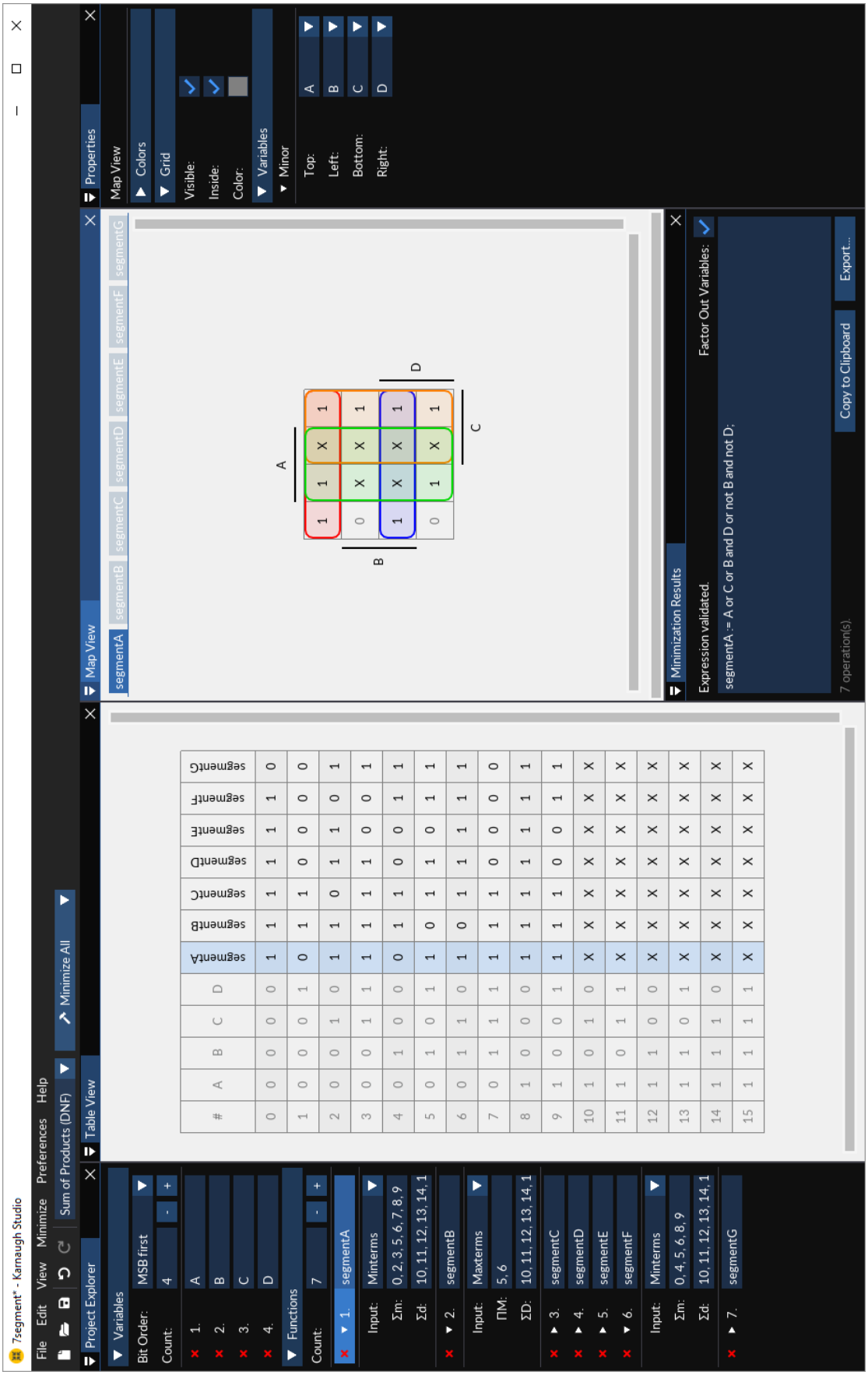
STL	Standardní knihovna šablon jazyka C++
CSV	Comma-separated values (čárkou oddělené hodnoty)
XML	Extensible markup language (rozšiřitelný značkovací jazyk)
GUI	Graphical user interface (grafické uživatelské rozhraní)
Bit	Binary digit (dvojková číslice)

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: UŽIVATELSKÉ ROZHRANÍ APLIKACE



PŘÍLOHA P II: STRUKTURA OBSAHU CD

Struktura obsahu přiloženého na CD je následující:

- Soubor *fulltext.pdf* obsahuje text práce ve formátu *PDF/A.*
- Adresář *Karnaugh Studio* obsahuje aplikaci ve spustitelné podobě.
- Adresář *Source Code* obsahuje zdrojový kód aplikace včetně všech souborů potřebných k jejímu sestavení.