Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Počítačové a komunikačné systémy a siete

Tím č. 8

Univerzálny virtuálny verifikačný panel logických obvodov

Projektová dokumentácia

Bc. Michal Kudlačák
Bc. Stanislav Martinický
Bc. Juraj Orságh
Bc. Ján Pivarček
Bc. Juraj Sebín
Bc. Marek Sivák

Ročník: 1. Ing.
Predmet: Tímový projekt
Pedagogický vedúci tímu: Prof. Ing. M. Kolesár, CSc.
Ak. rok: 2009/2010

Obsah

1 Úvod

1.1 Účel a rozsah dokumentu

Dokument je výsledkom spoločnej práce šiestich študentov v predmete Tímový projekt v akademickom roku 2009/2010. Venuje sa najmä problematike logických obvodov a verifikačných programov pomocou, ktorých je možné modelovať rôzne logické obvody. Obsahuje teoretickú dokumentáciu, ktorá popisuje jednotlivé aspekty riešenia zadaného problému ako je analýza, špecifikácia a návrh riešenia verifikačného panelu. Tento dokument je určený hlavne študentom a pedagógom FIIT STU.

1.2 Prehľad dokumentu

Dokument je rozdelený do jednotlivých hlavných kapitol, ktoré zoskupujú danú časť riešenia problému. V prvej kapitole je uvedený účel a rozsah dokumentu, prehľad dokumentu, zadanie projektu, cieľ projektu a zoznam skratiek. Druhá kapitola analyzuje problematiku logických obvodov a poskytuje krátky prehľad existujúcich programov, ktoré slúžia na modelovanie logických obvodov. V tretej kapitole uvádzame špecifikáciu zadania a požiadaviek na výsledný produkt. V štvrtej kapitole je uvedený hrubý návrh riešenia, ktorý pozostáva z troch modulov. Piata kapitola opisuje vytvorený prototyp aplikácie, požiadavky na aplikáciu a dosiahnuté výsledky v tejto fáze riešenia. Posledná kapitola obsahuje zoznam použitej literatúry.

1.3 Zadanie projektu

Navrhnite a implementujte programový systém pre osobný počítač, pomocou ktorého možno zostaviť štruktúru a ručne overiť funkciu logického kombinačného obvodu s normálnou štruktúrou, ktorý má najviac štyri vstupy a štyri výstupy.

Programový systém má umožniť voľbu podľa možnosti čo najväčšieho počtu režimov činnosti na základe zadaných úplných súborov logických členov s konečným počtom vstupov.

Nastavovanie hodnôt vstupných premenných (vstupných vektorov) treba umožniť pomocou virtuálnych tlačidiel a hodnoty výstupných premenných (výstupných vektorov) majú byť signalizované virtuálnymi žiarovkami.

Programový systém treba navrhnúť tak, aby bol použiteľný v pedagogickom procese pre predmet Logické obvody.

1.4 Ciele projektu

Cieľom nášho projektu je vytvoriť programový systém spustiteľný na osobnom počítači, ktorý bude modelovať funkcionalitu logických obvodov. Dôraz je kladený na implementovanie úplného súboru členov, funkcionalitu a grafické používateľské rozhranie. Taktiež je dôležité, aby tento systém bol vyhovujúci na používanie v pedagogickom procese pre predmet Logické obvody. A preto je dôležité ho navrhnúť tak, aby bol jednoduchý, názorný a intuitívne ovládateľný. Aby výsledný produkt spĺňal všetky kritéria je najprv potrebné analyzovať teoretickú časť problematiky a existujúce softvérové riešenia používané na modelovanie logických obvodov. Analýza je potrebná pre následnú špecifikáciu zadania a požiadaviek systému, a taktiež na vytvorenie správneho návrhu riešenia.

1.5 Zoznam skratiek

- ČSN - Česká Technická Norma
- UVVP - Univerzálny virtuálny verifikačný panel
- XML - eXtensible Markup Language

1.6 Zoznam pojmov

- Drag & Drop – Technika, ktorá sa používa v grafickom používateľskom prostredí pri práci s objektmi. Pomocou myši je možné presúvať objekty jednoduchým stlačením tlačidla myši a jej súčasnom presúvaní do cieľového miesta na obrazovke. Po pustení

tlačidla sa objekt umiestni do cieľového bodu. Je to funkcia, ktorá umožňuje jednoduché ovládanie pomocou myši.

- SourceGrid - Knižnica, ktorá umožňuje jednoduché a efektívne implementovanie hocijakých dát vo forme tabuľky.

2 Analýza problémovej oblasti

Analýza problémovej oblasti je dôležitá na pochopenie samotnej problematiky a stanovanie cieľov a požiadaviek projektu. V tejto 4asti sa budeme venovať teoretickej časti problematiky a taktiež aj existujúcim softvérom, ktoré modelujú a simulujú logické obvody.

2.1 Teória

Na začiatok je veľmi dôležité uviesť teóriu, ktorá sa skrýva za celou problémovou oblasťou. Je to potrebné na lepšie pochopenia logiky, ktorá musí byť implementovaná vo výslednom produkte. Kapitola teória je spracovaná na základe literatúry [1][2][3][4].

2.1.1 Výroky

Za výrok považujeme každú oznamovaciu vetu o ktorej má zmysel hovoriť či je pravdivá alebo nie. Výroky sa označujú veľkými písmenami A, B, ..., Z. Výrok je základným stavebným prvkom výrokovej logiky, ktorú využívajú logické obvody.

Informácia o tom, ktorý z dvoch možných prípadov pri vyhodnocovaní výroku nastal, predstavuje logickú hodnotu výroku. Logická funkcia je formálnym popisom výroku a nositeľkou pravdivostnej hodnoty výroku. Ide o zobrazenie z množiny výrokov do množiny {0,1}. Logická funkcia priraďuje výroku pravdivostnú hodnotu.

Logickými premennými (logická 1, logická 0) sú popisované pravdivostné hodnoty jednoduchých výrokov. Jednoduché výroky je možné spájať pomocou logických spojok, tým vznikajú nové zložené výroky. Ich pravdivostná hodnota závisí iba od pravdivostných hodnôt výrokov, z ktorých sú zložené a samozrejme od zvolenej (logickej) spojky.

Hlavné zložené výroky sú konjunkciu, disjunkciu, implikáciu a ekvivalenciu. Konjunkcia sa v bežnej reči vyjadruje ako "A a súčasne B", označuje sa A Λ B. Disjunkcia sa vyjadruje ako "A alebo B" a označuje sa A V B. Implikácia, má tvar "ak A, potom B" a označuje sa A => B. Ekvivalencia má tvar "A práve vtedy, keď, B" a značí sa A <=> B. Je potrebné spomenúť

aj negáciu. Negácia sa vzťahuje iba na jeden výrok. Negácia výroku A je výrok A', popiera to, čo tvrdí pôvodný výrok. V hovorovej reči sa používa ako "neplatí A" alebo "nie je pravda, že A". Označuje sa aj ¬A, prípadne $\overline{A}$.

2.1.2 Boolovská algebra a jej modely operácií

Booleova algebra využíva symboliku analogickú ako v klasickej algebre. V booleovej algebre značíme premenné a, b, ..., z.

Unárne booleovské funkcie pracujú iba s jednou premennou. Najznámejšia unárna funkcia je logická negácia.

Binárne booleovské funkcie pracujú s dvomi premennými a, b. Najznámejšími binárnymi funkciami sú konjunkcia, disjunkcia, negácia konjunkcie, negácia disjunkcie a neekvivalencia.

V booleovej algebre platia nasledujúce zákony:

Zákon	Súčtový tvar	Súčinový tvar
Komutatívny zákon	a+b=b+a	a.b=b.a
Asociatívny zákon	a+(b+c)=(a+b)+c	a.(b.c)=(a.b).c
Distributívny zákon	a+(b.c)=(a+b).(a+c)	a.(b+c)=(a.b)+(a.c)
De Morganov zákon	$\overline{a+b}=\overline{a}.\overline{b}$	$\overline{a.b}=\overline{a}+\overline{b}$
Zákon absorbcie	a+a.b=a	a.(a+b)=a
Zákon absorbcie negácie	$a+\overline{a}.b=a+b$	$a.(\overline{a}+b)=a.b$
Zákon vylúčenia	$a+\overline{a}=1$	$a.\overline{a}=0$
Zákon neutrálnosti 0 a 1	a+0=a	a.1=a
Zákon agresívnosti 0 a 1	a+1=1	a.0=0
Zákon dvojitej negácie $\overline{A}=A$	¬¬a=a	¬¬a=a

Tabuľka č. 1: Boolovská algebra

Booleovské funkcie je možné realizovať (modelovať) pomocou rôznych technických prostriedkov. Fyzikálne systémy, ktoré realizujú tieto booleovské funkcie budeme nazývať logické členy. Pri realizácii pomocou elektrických prostriedkov sa najčastejšie používajú tieto dva spôsoby:

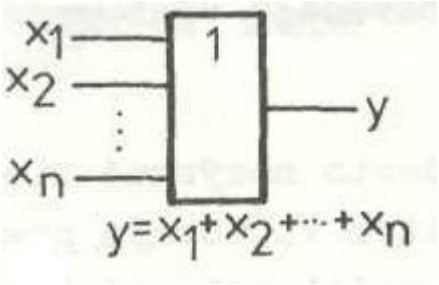
Využíva sa prítomnosť elektrického prúdu (napätia) na priradenie hodnoty 1 a jeho neprítomnosť na priradenie hodnoty 0. (V tomto prípade ide o tzv. pozitívnu logiku. Je možné uvažovať aj o opačnom priradení, vtedy hovoríme o negatívnej logike.

Využívajú sa technické zariadenia, ktoré sú schopné prevádzky pri dvoch (značne) odlišných hladinách elektrického napätia. Pre naše úvahy je výhodné jednu z nich označiť ako vysokú hladinu napätia (v praxi spravidla 5 V) a priradiť jej hodnotu 1 a druhú označiť ako nízku hladinu napätia (v praxi spravidla 0,5 V) a priradiť jej logickú hodnotu 0 (pozitívna logika).

2.1.3 Úplný súbor logických členov

Medzi základné logické členy patria :

- Logický člen alebo (OR) – realizuje operáciu logického súčtu, $F(x_1,x_2) = x_1 + x_2$

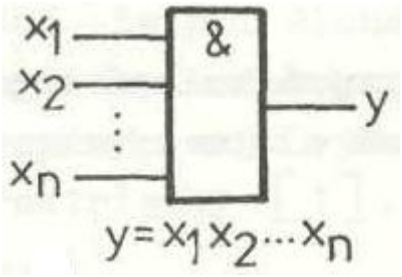


Obrázok č.1: Symbolická značka logického člena OR s viacerými vstupmi

X_1	X_2	$F(x_1,x_2)$
1	1	1
0	1	1
1	0	1
0	0	0

Tabuľka č.2: Pravdivostná tabuľka logického člena OR s 2 vstupmi

- Logický člen I (AND) – realizuje operáciu logického súčinu, $F(x_1,x_2) = x_1 \cdot x_2$

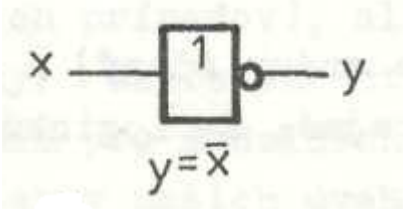


Obrázok č.2: Symbolická značka logického člena AND s viacerými vstupmi

X_1	X_2	$F(x_1,x_2)$
1	1	1
0	1	0
1	0	0
0	0	0

Tabuľka č.3: Pravdivostná tabuľka logického člena OR s 2 vstupmi

- Logický člen Invertor (NOT) – realizuje operáciu negácie, $F(x) = \bar{x}$

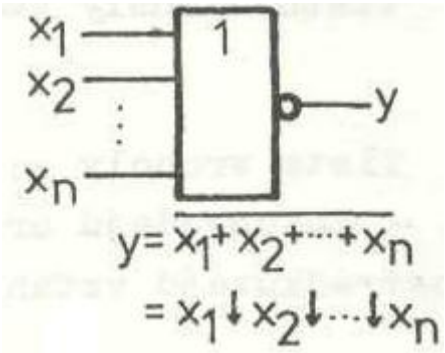


Obrázok č.3: Symbolická značka logického člena NOT

X	F(x)
1	0
0	1

Tabuľka č.4: Pravdivostná tabuľka logického člena NOT

- Piercov logický člen (NOR) – realizuje operáciu negácie logického súčtu, $F(x_1,x_2)=\overline{x_1 + x_2} = x_1 \downarrow x_2$

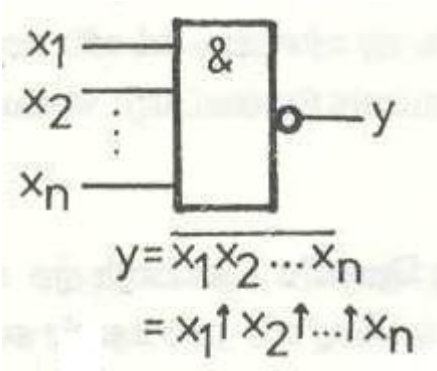


Obrázok č.4: Symbolická značka logického člena NOR s viacerými vstupmi

X_1	X_2	$F(x_1,x_2)$
1	1	0
0	1	0
1	0	0
0	0	1

Tabuľka č.5: Pravdivostná tabuľka logického člena NOR s 2 vstupmi

- Shefferov logický člen (NAND) – realizuje operáciu negácie logického súčinu, $F(x_1,x_2)= \overline{x_1.x_2} = x_1 \uparrow x_2$

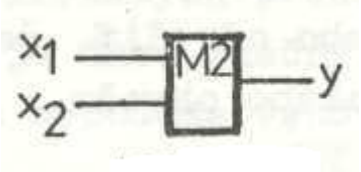


Obrázok č.5: Symbolická značka logického člena NAND s viacerými vstupmi

X_1	X_2	$F(x_1,x_2)$
1	1	0
0	1	1
1	0	1
0	0	1

Tabuľka č.6: Pravdivostná tabuľka logického člena NAND s 2 vstupmi

- Logický člen neekvivalencie (XOR) – realizuje operácie neekvivalencie, $F(x_1,x_2)= \overline{x_1}.x_2 + x_1.\overline{x_2} = x_1 \oplus x_2$

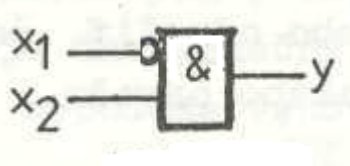


Obrázok č.6: Symbolická značka logického člena XOR

X_1	X_2	$F(x_1,x_2)$
1	1	0
0	1	1
1	0	1
0	0	0

Tabuľka č.7: Pravdivostná tabuľka logického člena XOR s 2 vstupmi

- Logický člen Zábrana – realizuje operáciu $F(x_1,x_2)=\overline{x_1}.x_2= x_1 \not\rightarrow x_2$

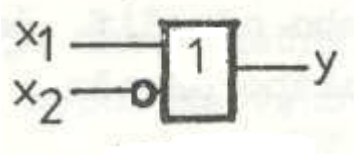


Obrázok č.7: Symbolická značka logického člena Zábrana

X_1	X_2	$F(x_1,x_2)$
1	1	0
0	1	1
1	0	0
0	0	0

Tabuľka č.8: Pravdivostná tabuľka logického člena Zábrana s 2 vstupmi

- Logický člen Implikácia – realizuje operáciu $F(x_1,x_2)=\overline{x_2} + x_1 = x_1 \rightarrow x_2$

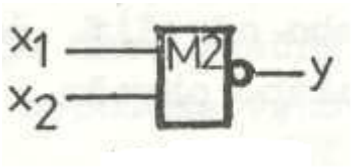


Obrázok č.8: Symbolická značka logického člena Implikácia

X_1	X_2	$F(x_1,x_2)$
1	1	1
0	1	0
1	0	1
0	0	1

Tabuľka č.9: Pravdivostná tabuľka logického člena Implikácia s 2 vstupmi

- Logický člen Ekvivalencia – realizuje operáciu $F(x_1,x_2)= \overline{x_1}.\overline{x_2} + x_1.x_2 = x_1 \equiv x_2$



Obrázok č.9: Symbolická značka logického člena Ekvivalencia

X_1	X_2	$F(x_1,x_2)$
1	1	1
0	1	0
1	0	1
0	0	1

Tabuľka č.10: Pravdivostná tabuľka logického člena Ekvivalencia s 2 vstupmi

Vezmime si niektorú množinu operácií $G=\{ g_1, g_2 \ldots g_k \}$, $k \geq 1$. Množinu G nazývame funkčou úplnou práve vtedy, ak pre každú Booleovu funkciu f existuje jej zodpovedajúci

výraz Z, t.j. f = |Z|, ktorý obsahuje iba operácie len z množiny G. Pre každý úplný súbor logických členov platí, že musí realizovať všetky operácie z niektorej funkčne úplnej množiny. Dá sa dokázať, že množina Booleových operácií je funčne úplná. Existujú viaceré množiny operácií, ktoré sú funkčne úplné. Ďalej sú uvedené všetky takzvané minimálne úplné systémy, zostavené z jednej, dvoch, alebo troch operácií, pre ktoré platí, že z nich nie je možné bez porušenia ich funkčnej úplnnosti odstrániť ani jednu operáciu.

- $\{\uparrow\}$
- $\{\downarrow\}$
- $\{\ .\ ,\ \bar{}\ \}$
- $\{\ +\ ,\ \bar{}\ \}$
- $\{\ \rightarrow\ ,\ \not\rightarrow\ \}$
- $\{\ \rightarrow\ ,\ \bar{}\ \}$
- $\{\ \rightarrow\ ,\ \oplus\ \}$
- $\{\ \not\rightarrow\ ,\ \equiv\ \}$
- $\{\ \not\rightarrow\ ,\ \bar{}\ \}$
- $\{\ .\ ,\ \oplus\ ,\ \equiv\ \}$
- $\{\ +\ ,\ \oplus\ ,\ \equiv\ \}$

Ak operácie môžu byť aj nulárne, t.j. 0 , I , potom existujú ešte nasledujúce minimálne úplné systémy:

- $\{\ \rightarrow\ ,\ 0\ \}$
- $\{\ \not\rightarrow\ ,\ I\ \}$
- $\{\ \oplus\ ,\ .\ ,\ I\ \}$
- $\{\ \equiv\ ,\ .\ ,\ 0\ \}$
- $\{\ \equiv\ ,\ +\ ,\ 0\ \}$
- $\{\ \oplus\ ,\ +\ ,\ I\ \}$

2.1.3.1 Logické obvody

Logické obvody sa skladajú z logických členov.

Logický člen je elementárny číslicový systém, ktorý realizuje niektorú booleovskú funkciu nad vstupnými premennými a jej výsledok poskytuje na svojom výstupe. Na označovanie logických členov sa používajú schematické značky. Najznámejšie sú schematické značky vychádzajú z normy ČSN a z americkej normy MIL-STD-806B.

Logické obvody delíme na obvody kombinačné a obvody sekvenčné Tieto dve skupiny obvodov sa opierajú o booleovu algebru. Obyčajne predpokladáme, že logický systém je definovaný na technickom zariadení a že jednotlivé premenné systému zodpovedajú určitým spojito sa meniacim fyzikálnym veličinám, najčastejšie veličinám elektromagnetického poľa. Logické hodnoty sa vyjadrujú pomocou dvoch disjunktných intervalov hodnôt príslušnej fyzikálnej veličiny, napr. elektrického napätia, prúdu alebo náboja, resp. magnetického toku a podobne. Ľubovoľnú n-ticu hodnôt vstupných premenných x1, ..., xn nazývame vstupným vektorom a ľubovoľnú m-ticu hodnôt výstupných premenných y1, ..., ym nazývame výstupným vektorom.

Pri kombinačných logických obvodoch výstupný vektor závisí od okamžitého vstupného vektora. Predchádzajúce kombinácie nemajú žiadny vplyv na výstupný vektor. Kombinačné logické obvody sú obvody, ktoré realizujú určitú logickú funkciu. Charakteristické je pre ne to, že výstup obvodu je jednoznačne daný jeho vstupom. Signál sa šíri postupne od vstupov k výstupu (neexistujú spätné väzby). Kombinačné obvody sú jednoduchšie ako sekvenčné obvody.

Pri sekvenčných logických obvodoch výstupný vektor závisí nielen od okamžitej kombinácie vstupov, ale taktiež od postupnosti vstupov v predchádzajúcom čase. Takto sa môže správať systém iba ak je schopný „pamätať si", čo sa odohralo na jeho vstupoch v minulosti a podľa toho reagovať na súčasný vstupný vektor.

Kombinačné aj sekvenčné logické obvody môžu pracovať buď asynchrónne alebo synchrónne. Pri asynchrónnych logických obvodoch vzniká zmena stavu a výstupného vektora okamžite po zmene stavu vstupov. Pri synchrónnych logických obvodoch vzniká zmena stavu a výstupného vektora až po prijatí synchronizačného (hodinového) signálu, kedy sa vyhodnocujú vstupné a stavové veličiny.

2.2 Softvér na podporu modelovania logických funkcií

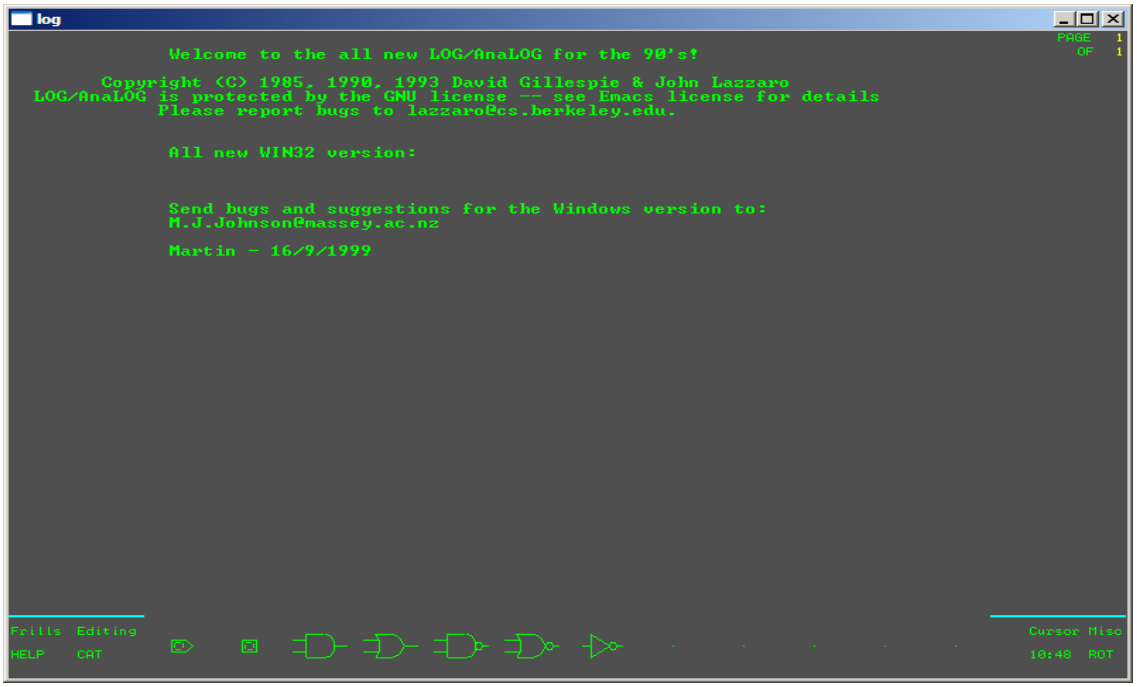
Existuje veľké množstvo komerčných a voľne dostupných softvérových riešení, ktoré sa zaoberajú problematikou modelovania logických funkcií a obvodov. V tejto kapitole opíšeme niektoré vybrané programy.

2.2.1 Log

V tejto aplikácii [4] sa dajú zostavovať jednoduché logické obvody s použitím základných logických členov ako AND, NAND, XOR, OR a iné. Táto aplikácia umožňuje, aby si používateľ sám navrhol schému a umiestnil logické členy ako uzná za vhodné. Logické členy sa dajú otáčať. LOG podporuje taktiež zvýraznenie vodičov v závislosti od logickej hodnoty, ktorá vodičom tečie. Nechýbajú mu ani základné editovanie funkcie ako mazanie, presúvanie a kopírovanie. V LOGu je možné svoju prácu uložiť alebo otvoriť už uloženú prácu.

Za najväčšiu výhodu považujem širokú škálu logických členov, ktoré LOG podporuje. LOG podporuje analógovú a taktiež digitálnu simuláciu.

Jediná vec voči ktorej by mohli byť výhrady, je azda trochu ťažkopádnejšie ovládanie a užívateľské rozhranie.



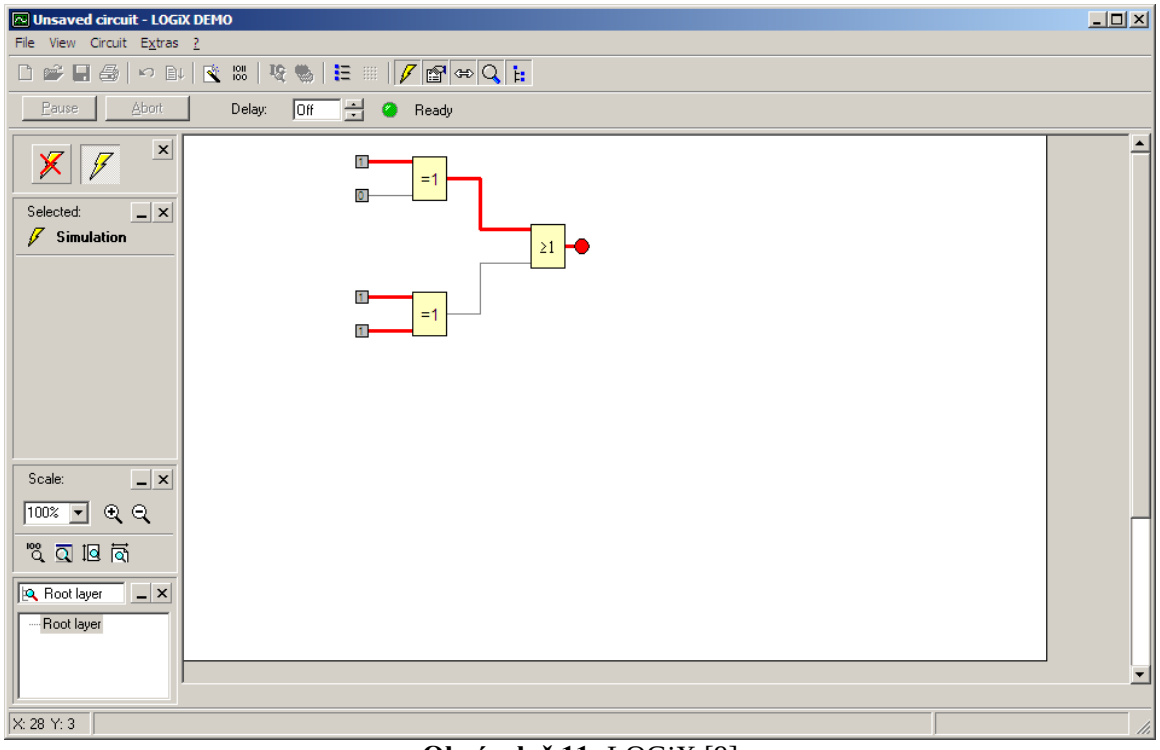
Obrázok č.10: LOG [4]

2.2.2 LOGiX

LOGiX [4] [9] je program pre tvorbu a simuláciu logických obvodov. Podobne ako LOG, LOGiX umožňuje umiestnenie logických členov na obrazovku a ich postupné spájanie vodičmi. O správnosti fungovania navrhnutého logického obvodu sa používateľ presvedčí pomocou simulácie, pri ktorej sú zvýraznené vodiče nesúce logickú 1.

LOGiX umožňuje generovanie logických obvodov na základe pravdivostnej tabuľky alebo booleovského výrazu. LOGiX umožňuje vytváranie vlastných elementov. Tieto elementy obsahujú obvod, ktorý požívateľ navrhne a potom k nemu pristupuje ako k logickému členu. LOGiX tiež ponúka základné editovanie vlastnosti. Je možné svoju prácu uložiť a neskôr opäť otvoriť. Navrhnutý logický obvod je možné vytlačiť na papier.

Nevýhodou tohto programu je, že ak používateľ chce s ním pracovať viac ako 30 minút, musí si zakúpiť licenciu. LOGiX dáva k dispozícii menej logických členov ako vyššie spomenutý LOG.



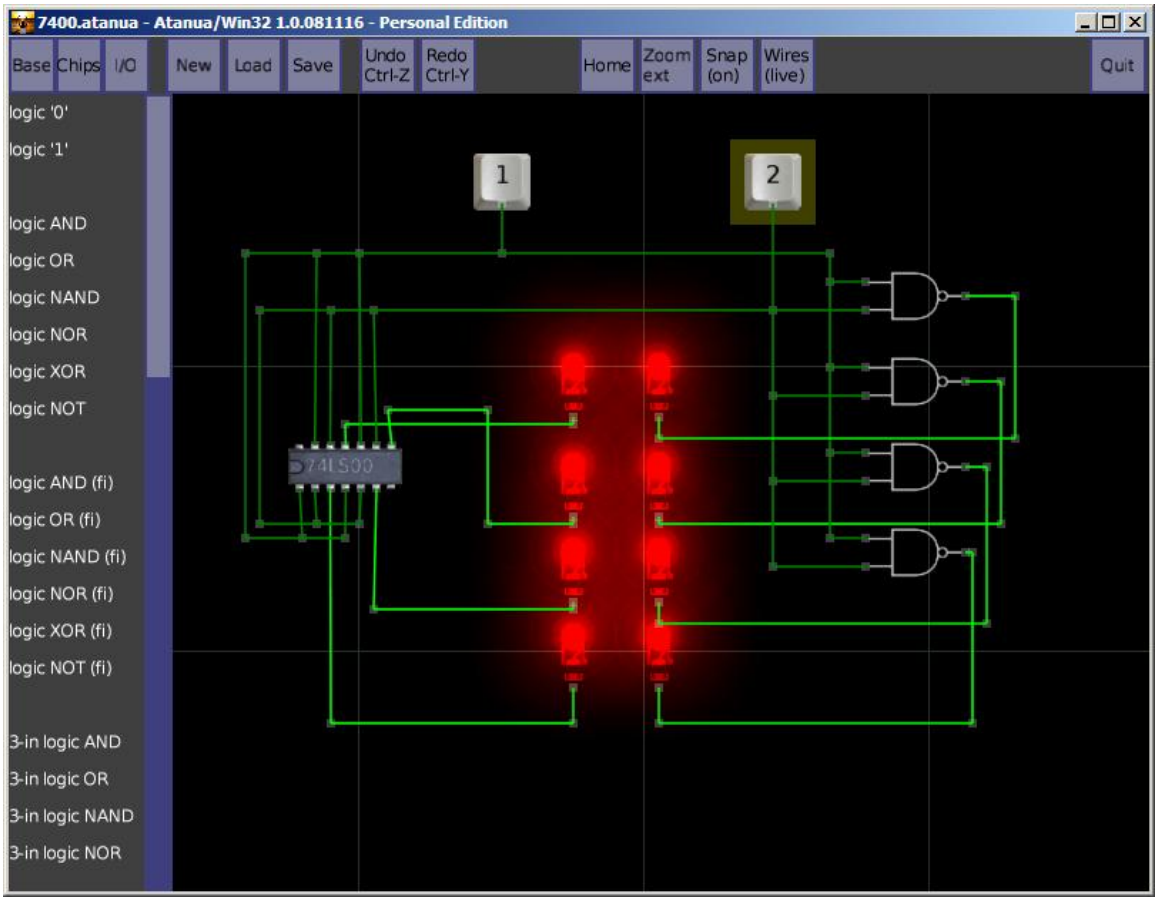
Obrázok č.11: LOGiX [9]

2.2.3 ATANUA

Atanua [4] [10] je logický simulátor navrhnutý na vyučovanie základov booleovskej logiky a elektroniky. Poskytuje príjemné používateľské rozhranie a jednoduché ovládanie. Umožňuje uložiť vytvorený obvod a opätovne ho otvoriť.

Používateľovi poskytuje základné logické obvody, ale taktiež integrované obvody (napr. IO 7400 obsahujúci NAND členy), s ktorými sa dá stretnúť v reálnom svete. Simulácia beží neustále a preto používateľ už pri návrhu vidí ako sa obvod správa. Atanua poskytuje množstvo vstupných a výstupných elementov. Z výstupných sú to hlavne rôznofarebné LED diódy a 7 segmentový displej. Ako vstupy sa dajú nadefinovať rôzne tlačidlá na klávesnici. Napríklad tlačidlo 1. Pri stlačení vygeneruje logickú 1 a pri pustení logickú 0.

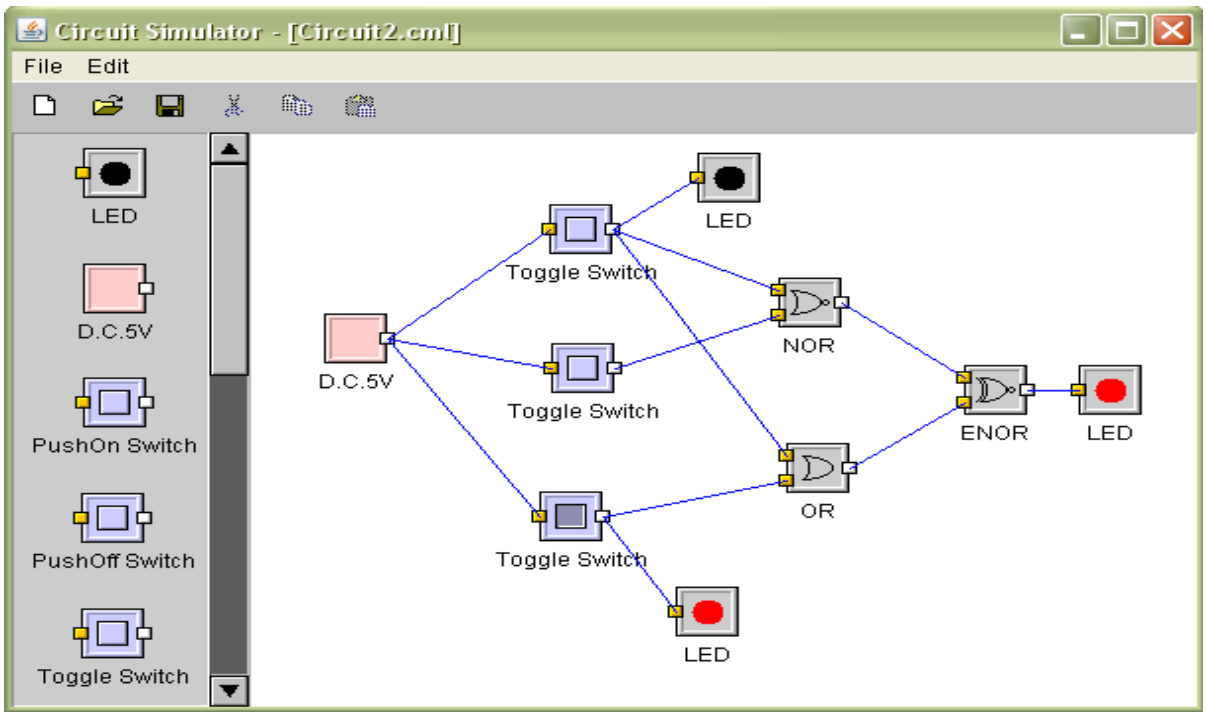
Zaujímavá je aj vlastnosť nazvaná Anti-cheating tool, ktorá je určená pre učiteľov na skontrolovanie domácej úlohy študentov. Atanua je k dispozícii pre platformy Windows, MAC OS a Linux. Pre nekomerčné využitie je program zadarmo.



Obrázok č.12: Atanua [10]

2.2.4 SIMCIR

Simcir 1.2.1 [5] [8] umožňuje používateľovi zapájať ponúknuté súčiastky a na ich vstup vysielať kombinácie signálov. Takto si používateľ môže overiť priechodnosť jednotlivých logických obvodov. Na ľavej strane okna aplikácie sa nachádza konečný súbor súčiastok, ktoré si používateľ môže ľubovoľné zapájať do logických štruktúr. V ponuke sa nachádza LED, zdroj napätia, tri druhy vypínačov a sedem druhov logických členov. Všetky logické členy s výnimkou invertora majú dva vstupné póly a jeden výstupný. Dané komponenty je možné pohybom myši preniesť na pracovnú plochu a tam ich ľubovoľné zapájať. Zmeny v zapojení alebo v signáloch sa prejavia hneď ako sa uskutočnia.



Obrázok č.13: Simcir 1.2.1 [8]

2.2.5 Porovnanie analyzovaného softvéru

V tejto časti dokumentu sa nachádza porovnanie výhod a nevýhod jednotlivých softvérových produktov, ktoré boli analyzované.

Názov produktu	Výhody	Nevýhody
Log	široký výber logických členov možnosť editovania modelu možnosť zvýraznenia logických hodnôt vo vodičoch	ťažkopádne ovládanie neprívetivé grafické rozhranie
LOGiX	možnosť vytvárania vlastných elemetov možnosť generovania logických obvodov na základe pravidvostnej tabuľky možnosť editovania modelu	menší výber logických členov
ATANUA	prehľadné grafické rozhranie množstvo vstupných a výstupných členov anti-cheating tool	
SIMCIR	prehľadné grafické rozhranie	malý počet logických členov

Tabuľka č. 11: Porovnanie dostupných softvérových riešení

3 Špecifikácia riešenia

Táto kapitola obsahuje analýzu a špecifikáciu zadania, z ktorej vychádzajú požiadavky na výsledný softvér. Táto časť je dôležitá na správne navrhnutie riešenia.

3.1 Analýza a špecifikácia zadania

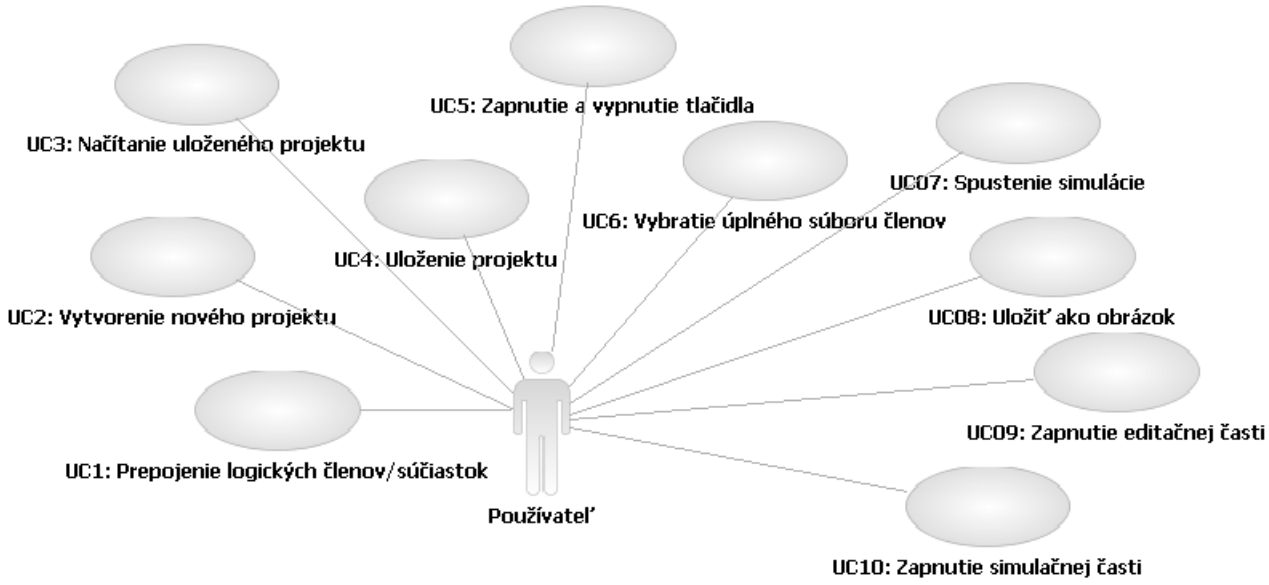
Navrhovaný programový systém by mal spĺňať zadanie v plnom rozsahu. Výsledný produkt má mať formu programu, ktorý je ľahko prevádzkovateľný na školských osobných počítačoch. Keďže tento programový systém bude používaný v pedagogickom procese, finálny produkt musí byť prívetivý pre používateľa, jednoducho a intuitívne ovládateľný a názorný. Ďalej musí umožniť modelovať a simulovať kombinačné logické obvody s normálnou štruktúrou, ktoré majú najviac štyri vstupy a štyri výstupy. Musí umožniť uloženie navrhnutého logického obvodu, následné otvorenie a upravenie. Logický obvod bude reprezentovaný graficky, jednotlivé členy budú napevno pridelené na plochu. Spojenie členov bude možné pomocou myši. Bude podporovaná veľká škála logických členov s rôznym počtom vstupov, ktoré budú uložené v knižnici logických členov do úplných súborov členov. Knižnica bude obsahovať všetky úplne súbory členov. Na zobrazenie vstupnej a výstupnej hodnoty budú použité žiarovky, ktoré budú farebne rozlišovať logickú nulu a jednotku. Jednotlivé časti modelovaného logického obvodu budú prepájané spojovníkmi, ktoré budú zobrazovať prechodovú hodnotu. Hodnota bude farebne rozlišovať logickú nula a jednotku. Výsledný produkt sa bude skladať z dvoch častí, editačnej a simulačnej. V editačnej časti bude možné navrhnúť a upraviť logický obvod. V simulačnej časti bude možné odsimulovať jeho správanie. Výsledný produkt má dávať správne výstupy. Pomocou simulácie bude možné overiť funkciu logického kombinačného obvodu.

Pri samotnom vývoji softvéru bude kladená pozornosť najmä na použiteľnosť, príjemné užívateľské prostredie a modulárnosť, aby sa v prípade potreby dal ľahko rozšíriť o ďalšie časti. Podľa nás by určite bola zaujímavá funkcia *Drag & Drop*, ktorá by poskytovala jednoduché a intuitívne ovládanie celej aplikácie.

3.2 Funkcionálne požiadavky na softvér

Uvažujeme o jednom používateľovi, ktorý bude mať prístup k nasledujúcim funkciám programu. Diagram prípadov použitia, ktorý zobrazuje funkcie nášho riešenia je na obrázku číslo 14. Bude poskytovaná nasledovná funkcionalita systému:

- prepojenie logických členov alebo súčiastok
- vytvorenie nového projektu
- načítanie uloženého projektu
- uloženie projektu
- zapnutie a vypnutie tlačidla
- vybratie úplného súboru členov
- spustenie simulácie
- uložiť ako obrázok
- zapnutie editačnej časti
- zapnutie simulačnej časti



Obrázok č.14: Prípady použitia.

3.2.1 Popis prípadov použitia

Táto časť obsahuje popis jednotlivých prípadov použitia, ktoré sú zobrazené na obrázku 14.

Prepojenie logických členov alebo súčiastok – Táto funkcia umožní priamo interakciu používateľa s logickými členmi. Bude možné prepojiť jednotlivé súčiastky medzi sebou a tak vytvoriť logickú schému.

Vytvorenie nového projektu – Jedna z možností na ovládacom panely (panel podporných funkcií) bude ponúkať možnosť vytvoriť nový projekt, ktorý bude obsahovať prázdnu pracovnú plochu.

Načítanie uloženého projektu – Jedna z možností na ovládacom panely (panel podporných funkcií) bude ponúkať možnosť otvoriť uložený projekt, ktorý bude obsahovať schému logického obvodu už predtým vytvorenú.

Uloženie projektu – Jedna z možností na ovládacom panely (panel podporných funkcií) bude ponúkať možnosť uložiť si vytvorenú schému logického zapojenia, a tak mať túto schému k dispozícii neskôr.

Zapnutie a vypnutie tlačidla – Táto funkcia umožní meniť používateľovi logické hodnoty (log. 0 a log. 1) na vstupe. Stlačením tlačidla sa zmení vstupná hodnota.

Vybratie úplného súboru členov – Používateľ si bude môcť vybrať úplný súbor členov zo zoznamu. Každý súbor bude obsahovať obmedzený počet a typ preddefinovaných členov, s ktorými bude musieť používateľ pracovať.

Spustenie simulácie – Jedna z možností na ovládacom panely (panel podporných funkcií) bude ponúkať možnosť spustenia simulácie zapojeného logického obvodu. To umožní zobrazenie výsledných logických hodnôt na výstupe.

Uložiť ako obrázok – Jedna z možností na ovládacom panely (panel podporných funkcií) bude ponúkať možnosť uložiť zapojenú schému ako obrázok vo formáte .jpg, .gif alebo .bmp.

Zapnutie editačnej časti – Používateľ bude môcť prepínať medzi dvomi stavmi aplikácie. V tomto stave bude môcť editovať a zapájať logickú schému.

Zapnutie simulačnej časti – Používateľ bude môcť prepínať medzi dvomi stavmi aplikácie. V tomto stave bude môcť simulovať zapojenú logickú schému.

3.3 Nefunkcionálne požiadavky na softvér

Medzi ďalšie požiadavky, ktoré nemajú funkcionálny charakter by sme mohli zaradiť:

- Používateľsky prívetivé a prehľadné grafické rozhranie.
- Názorné a prehľadné zobrazenie výstupu a celého programu (farby a rozlíšenie).
- Farebné rozlíšenie spojov podľa vedenej logickej hodnoty.
- Jednoduché a intuitívne ovládanie.
- Prenositeľnosť a multiplatformový produkt.
- Nie je nutná inštalácia programu.
- Čo najmenšia náročnosť na počítačový systém (procesor a pamäť).
- Modulárnosť programu.

3.4 Požiadavky na grafické rozhranie

Je potrebné navrhnúť rozhranie, ktoré bude jednoduché, prehľadné a prívetivé. Preto je potrebné ho rozdeliť na rovnocenné časti. Grafické používateľské rozhranie sa bude skladať z nasledovných častí:

- pracovná plocha
- knižnica logických členov
- riadiaca lišta (panel podporných funkcií).

Pracovná plocha – Bude slúžiť na prepájanie a modelovanie logickej schémy.

Knižnica logických členov – Bude obsahovať úplný súbor logických členov, ktoré bude možné používať pri modelovaní schémy. Knižnica bude obsahovať úplné súbory členov.

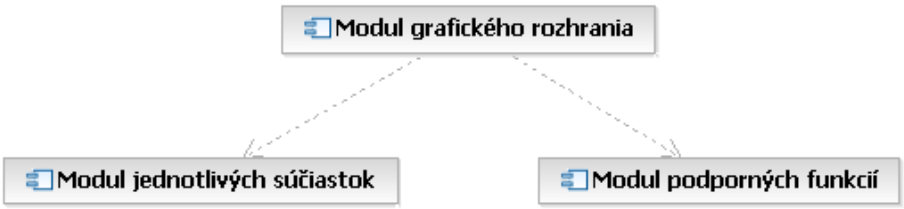
Riadiaca lišta – Bude ponúkať používateľovi jednoduché ovládanie aplikácie pomocou podporných funkcií.

4 Návrh riešenia

Pri návrhu výsledného produktu budeme vychádzať zo zadania a našej špecifikácie. Program budeme vyvíjať v jazyku C# (platforma Microsoft .Net Framework) v prostredí Microsoft Visual Studio. Keďže výsledný produkt bude použitý vo výučbe na školských počítačov, kde je nainštalovaný operačný systém Microsoft Windows, tak sme sa rozhodli pre toto prostredie a programovací jazyk. Taktiež by sme sa radi zdokonalili vo vývoji aplikácií v prostredí .Net. Pri samotnom návrhu riešenia problému sa bude vychádzať z vlastností objektovo orientovaného prístupu. Projekt rozložíme na niekoľko menších častí a ich riešenie si rozdelíme v rámci tímu. Budeme sa usilovať o čo najpresnejšie splnenie požiadaviek objednávateľa.

4.1 Hrubý návrh riešenia

Systém bude zložený z viacero podsystémov, ktoré sú medzi sebou navzájom závislé v určitých spoločných bodoch. Toto zaručí, že jednotlivé podsystémy, alebo moduly sa budú môcť vytvárať súčasne, pričom bude dopredu známe rozhranie pomocou ktorého si budú medzi sebou vymieňať dáta. Vysoko abstraktný model architektúry celého systému je znázornený na obrázku číslo 15.



Obrázok č.15: Moduly systému.

4.1.1 Modul logických súčiastok

Tento modul bude obsahovať abstraktnú reprezentáciu jednotlivých súčiastok logických obvodov. Výsledný produkt bude podporovať použitie hlavne nasledovných logických členov:

- AND – vynásobenie vstupných hodnôt.

- NAND – znegovanie vynásobených vstupných hodnôt.
- OR – sčítanie vstupných hodnôt.
- NOR – znegovanie sčítaných vstupných hodnôt.
- XOR – logická neekvivalencia.
- NOT – negovanie vstupnej hodnoty.

Ďalšie objekty, s ktorými bude možné pracovať sú:

- Žiarovka – indikácia 1 alebo 0. Bude sa používať na indikovanie vstupu a aj výstupu.
- Spoj – prechod medzi jednotlivými súčiastkami.
- A iné (pozn.: bude určené po dohode s vedúcim).

Členy budú zaradené do úplných súborov členov. Každý takýto súbor bude obsahovať obmedzený počet logických členov a len určitý typ členov. Jednotlivé členy budú napevno pridelené na plochu. Definície logických súčiastok respektíve členov budú udávať ich funkcionalitu. Všetky logické členy budú dediť od triedy *Gate*, ktorá bude vyzerať približne takto:

```
public class Gate
{
        int id = 0;
        int inputs;
        public boolean input1 = false; // logicka hodnota na prvom vstupe
        public boolean input2 = false; // logicka hodnota na druhom vstupe
        public boolean input3 = false; // logicka hodnota na tretom vstupe
        public boolean input4 = false; // logicka hodnota na stvrtom vstupe

        public boolean output = false; // logicka hodnota na vystupe

        public int input1ID = 0; // id prveho vstupu
        public int input2ID = 0; // id druheho vstupu
        public int input3ID = 0; // id tretieho vstupu
        public int input4ID = 0; // id stvrteho vstupu
```

```
    public int outputID = 0; // id vystupu
}
```

Každý člen bude mať maximálne štyri vstupy. Koľko ich naozaj bude, sa určí pri vytváraní objektu triedy reprezentujúcej logický člen nastavením premennej *inputs*. Bude tu aj možnosť upravovať počet vstupov aj vtedy, keď člen už bude vytvorený. Bude to prebiehať zmenou hodnoty premennej *inputs*.

Každý člen bude mať svoj identifikátor *id*. Bude to celé číslo väčšie ako nula. Keď výstup pripojíme na vstup nejakého člena, tak do premennej *outputID* sa zapíše *id* člena, do ktorého vstupu sme sa pripojili. Podobne to bude fungovať aj so vstupmi.

Každý člen bude mať metódu *processInputs*, ktorá podľa hodnôt na vstupe priradí hodnotu na výstupe. Táto metóda sa zavolá pre každý člen logiského obvodu vždy, keď sa udeje nejaká zmena. Príkladom zmeny môže byť odstránenie členu, prepojenie členov, zmena vstupnej premennej obvodu. Metóda *processInputs* pre trojvstupový člen AND vyzerá nasledovne:

```
public void processInputs()
{
    if (input1 && input3 && input3) output = true;
    else output = false;
}
```

4.1.2 Modul grafického používateľského rozhrania

Tento modul bude obsahovať reprezentáciu grafického rozhrania ako jediný prístupový bod používateľa k aplikácii. Modul bude obsahovať tri časti:

- pracovná plocha,
- knižnica logických členov,
- riadiaca lišta (panel podporných funkcií).

Pracovná plocha bude slúžiť na prepájanie a modelovanie logickej schémy. Knižnica logických členov bude obsahovať úplný súbor logických členov, ktoré bude možné používať

pri modelovaní schémy. Riadiaca lišta bude ponúkať používateľovi jednoduché ovládanie aplikácie. Jej súčasťou bude menu, ktoré bude umožňovať uloženie a načítanie schémy, otvorenie novej pracovnej plochy a iné podporné funkcie.

Jednotlivé členy budú napevno pridelené na plochu na základe výberu úplného logického súboru členov. Počet a typ členov bude obmedzený. Keď sa pridá nový logický člen na plochu, tak sa vytvorí objekt triedy, ktorá ho opisuje, ktorý sa pridá do poľa *gates*. Je to objekt triedy *ArrayList*, ktorá umožňuje robiť s užitočné operácie so svojimi členmi. Deklarácia a inicializácia *ArrayListu gates* vyzerá takto:

```
public static ArrayList<Gate> gates;
...
gates = new ArrayList();
```

Ak sa zmení hodnota výstupu nejakého členu, tak sa tak sa zavolajú metódy, ktoré ju prenesú na ostatné pripojené členy a okrem toho overia, či nie je potrebné zmeniť vizuálnu reprezentáciu členov (obrázky na pracovnej ploche).

4.1.3 Modul podporných funkcií

Modul podporných funkcií bude obsahovať ďalšiu funkcionalitu programu, ktorá priamo nesúvisí so zadaním, ako možnosť uloženia a načítania namodelovaného logického obvodu, ukladanie a načítavanie zmien v modely a podobne. Tieto funkcie bude možné ovládať pomocou riadiacej lišty, ktorá bude v hornej časti grafického používateľského rozhrania.

Základná funkcionalita programu bude ovládaná nasledovnými úkonmi na pracovnej ploche:

- **Prepojenie logických členov alebo súčiastok** – prepojenie logických členov a súčiastok na ploche.
- **Zapnutie a vypnutie tlačidla** – nastavovanie logickej hodnoty na vstupe.
- **Vybratie úplného súboru členov** – vybratie počtu a typu členov na modelovanie.
- **Zapnutie editačnej časti –** editovanie a zapájanie schémy.
- **Zapnutie simulačnej časti** – simulovanie zapojenej schémy.

Riadiaca lišta bude obsahovať nasledovné voľby:

- **Súbor**
  - **Nový** – vytvorenie nového projektu.
  - **Otvoriť** – načítanie uloženého projektu.
  - **Uložiť** – uloženie projektu.
  - **Uložiť ako obrázok** – schému na pracovnej ploche uloží ako jpg súbor.
  - **Ukončiť** – vypnutie programu.
- **Knižnica**
  - **Zobrazenie úplného súboru členov** – Zobrazí úplný súbor členov.
  - **Vybratie úplného súboru členov** – Vybratie úplného súboru členov.
- **Spustenie simulácie** – Umožní začatie simulovania toku logických hodnôt.
- **O programe**
  - **Autori** – Informácie o autoroch programu.
  - **Program** – Informácie o programe.
  - **Pomoc** – Krátky návod ako používať program.

Ďalšie funkcie môžu byť pridané po dohode s pedagogickým vedúcim.

5 Prototyp

Táto kapitola popisuje vytvorený prototyp aplikácie. Delí sa na dve hlavné časti. Požiadavky na aplikáciu opisuje funkcie a vlastnosti, ktoré sú zahrnuté v prototype. Časť dosiahnuté výsledky sa zaoberá výstupom z fázy tvorby prototypu. Opisuje možnosti aplikácie a všetky funkcie, ktoré boli v tejto fáze implementované.

5.1 Požiadavky na aplikáciu

Cieľom je navrhnúť a implementovať aplikáciu, ktorá by spĺňala požiadavky dohodnuté s vedúcim projektu. Požiadavky boli do značnej miery všeobecne formulované, preto sme často museli hľadať zhodu na spôsobe ich realizácie na spoločných stretnutiach celého tímu.

Aplikácia má umožniť zostaviť a ručne overiť štruktúru logického obvodu s normálnou štruktúrou. Používateľ mal k dispozícii konečný počet členov zo zvoleného minimálneho úplneho súboru logických členov. Na výber majú byť všetky možné minimálne úplné súbory logických členov. Spomenuté sú v kapitole 2.1.3. Medzi nimi má byť možné ľubovoľne prepínať, vždy je však možné pracovať len s jedným. Je potrebné ponúknuť taký počet jednotlivých členov, ktorý sa ukáže ako optimálny. Cieľom tohto je prinútiť používateľa pracovať len s tým, čo má k dispozícii. To môže znamenať, že počet členov a ich vstupov môže ovplyvniť výslednú štruktúru obvodu. Je na nás, či členy budú vo verifikačnom paneli umiestnené napevno alebo bude možné s nimi hýbať. Čo sa týka počtu vstupov a výstupov obvodu, požadované je, aby to bol tiež konečný počet, odporúčajú sa štyri. Pri každom člene má byť napísané, akú logickú hodnotu má na svojom výstupe.

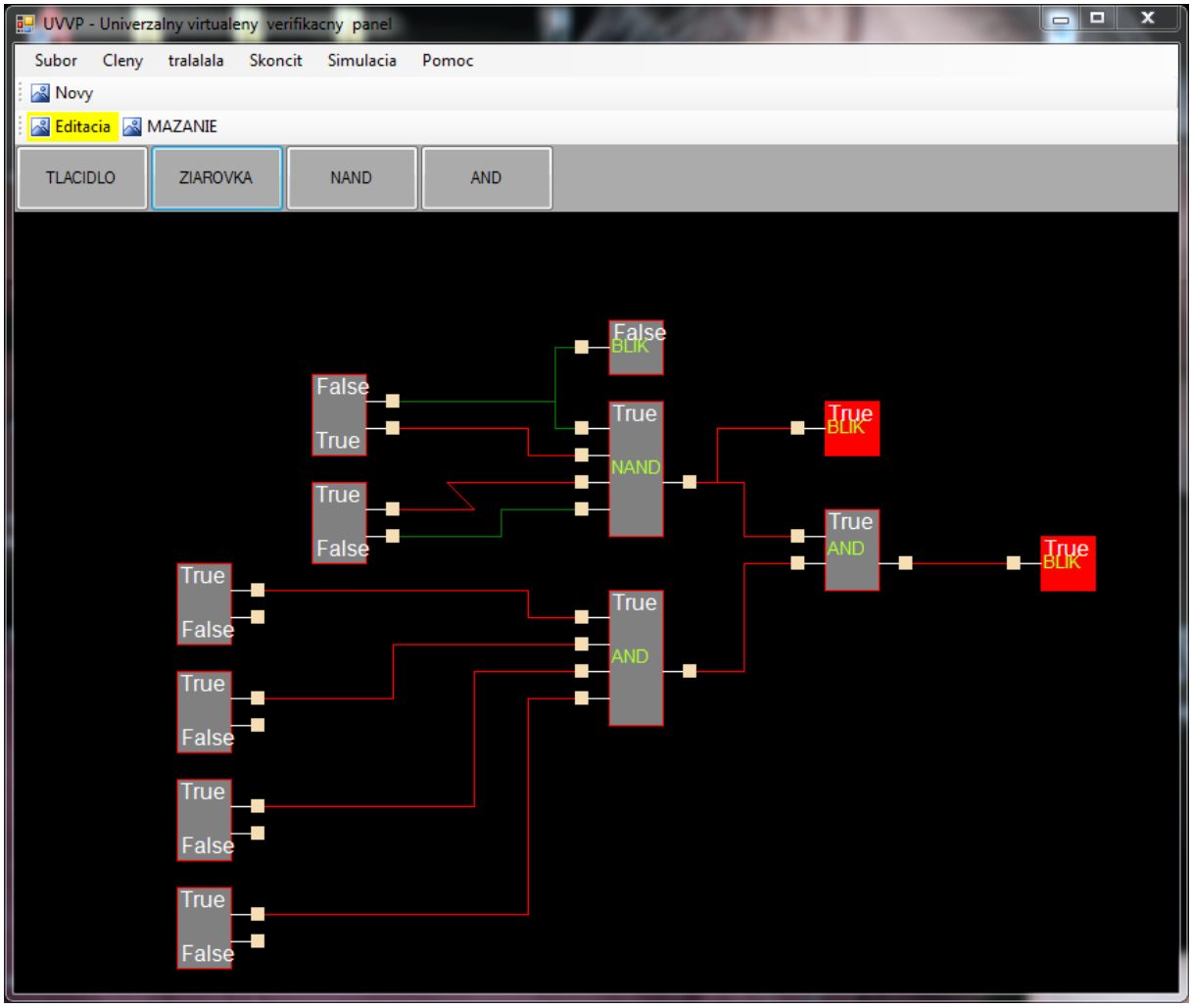
Ďalej je potrebné do štruktúry pridať žiarovky, ktoré by signalizovali výstupnú hodnotu zostaveného logického obvodu a tlačidlá, ktoré by do obvodu privádzali požadované logické hodnoty. Tlačidlo má mať dva výstupy, na jednom má byť logická 1, na druhom 0. Po stlačení sa majú oba výstupy prehodiť.

Bolo treba vymyslieť vhodný spôsob, ktorým by sa členy obvodu prepájali. Pre uľahčenie hľadania prípadných chýb na zostavenom logickom obvode malo byť umožnené sledovať

hodnoty aj na čiarach prepájajúcich členy. To sa dá dosiahnuť napríklad zmenou farieb, ktorými sa vykresľujú.

5.2 Dosiahnuté výsledky

Prototyp aplikácie vychádza z požiadaviek, ktoré boli naňho od začiatku kladené. Ukážka okna aplikácie je na obrázku 16.



Obrázok č.16: Ukážka prototypu.

Po spustení aplikácie sa zobrazí dialógové okno, ktoré vyzve používateľa, aby si vybral súbor logických členov. Na výber budú všetky možné. Údaje o nich sú uložené v xml súbore, z ktorého sa načítajú. Logický súbor, s ktorým pracujeme, je možné kedykoľvek zmeniť kliknutím na tlačidlo Nový. Vtedy sa do ponuky, z ktorej môžeme vyberať logické členy

nahrajú členy nového súboru a obsah pracovnej plochy, na ktorej zapájame logický obvod, sa vyčistí.

Prototyp zatiaľ dovoľuje pracovať s neobmedzeným počtom členov, vstupov a výstupov. Nie je to chyba, ale v ďalších fázach vývoja aplikácie sa všetko obmedzí na konečný počet. Budú k dispozícii členy úplného súboru, ktoré budú umiestnené priamo na paneli a budú pozostávať z konečného počtu vstupov.

Prototyp umožňuje prácu v dvoch režimoch: editácie a mazania. V režime editácie je možné pridávať, premiestňovať a prepájať členy a meniť hodnoty na tlačidlách. Režim mazania umožňuje vymazať ľubovoľnú časť logického obvodu, ktorú označíme kliknutím. Možné je teda vymazávať nielen logické členy, žiarovky, virtuálne tlačidlá, ale aj čiary.

Výber člena prebieha kliknutím na tlačidlo z ponuky. Pri niektorých členoch je možnosť zvoliť počet vstupov. Maximálne môžme nastaviť osem vstupov. Po potvrdení sa člen zobrazí v ľavom hornom okraji pracovnej plochy, odkiaľ je možné ho presunúť na ľubovoľné miesto plochy. Hodnota výstupu člena je napísaná v jeho hornej časti.

Okrem členov, ktoré tvoria logický súbor, sú v ponuke aj členy Tlačidlo a Žiarovka. Tlačidlo privádza do obvodu vstupné hodnoty. Ako bolo požadované, má dva výstupy, na každom je iná logická hodnota. Po kliknutí naňho sa vymenia. Žiarovka má len jeden vstup, ak je na ňu prevedená logická 1, rozsvieti sa načerveno, inak bude signalizovať logickú 0. Neskôr budú členy Tlačidlo a Žiarovka pevne umiestnené na paneli.

Členy sa prepájajú čiarami, ktoré sa kreslia režime editácie. Na prepojenie dvoch členov je potrebné kliknúť myšou na výstup jedného člena, preniesť ju na vstup druhého člena a pustiť. Po kliknutí sa pri kreslení čiary zobrazí biela mriežka, ktorá uľahčí kreslenie. Mriežka sa zobrazí aj pri presúvaní členov logického obvodu. Ak cez čiaru prechádza hodnota logická 0, je znázornená zelenou farbou. Ak 1, tak je červená.

Vzhľadom k tomu, že sa jedná o prototyp aplikácie, mnohé veci na nej budú prerobené, pridané alebo vymazané, za účelom zvýšenia jej efektívnosti alebo uľahčenia ovládania. Snahou bolo, aby bola aplikácia jednoduchá, názorná a intuitívne a ľahko ovládateľná. Je to

dôležité zvlášť preto, lebo má byť použiteľná v pedagogickom procese. Zostavený obvod je možné exportovať ako výstup vo forme obrázku.

6 Implementácia riešenia

Implementácia riešenia obsahuje doplnenie špecifikácie a riešenia, opis realizácie, návrh systému, architektúru systému, návrh algoritmov a iné dôležité časti. Táto kapitola obsahuje opis zmien a nedostatkov, ktoré boli v častiach špecifikácie a hrubý návrh. Taktiež obsahuje nové veci, ktoré pribudli pri realizácií produktu.

6.1 Doplnenie špecifikácie riešenia

Pôvodná špecifikácia vychádzala z predpokladu, že aplikácia bude plne prístupná používateľom a bude možné používať funkciou *Drag & Drop*. Avšak tento návrh bol zmenený. Hlavná myšlienka je vytvoriť hardvérový verifikačný panel, aký sa používal v minulosti. To znamená, že na stole bol položený veľký hardvérový modul, ktorý obsahoval obmedzený počet členov. Bolo možné len prepájať logické členy.

Po zapnutí aplikácie sa zobrazí okno, kde je možné si vybrať súbor členov. Používatelia resp. študenti si budú musieť vybrať daný súbor členov podľa zadania, ktoré dostanú. Jednotlivé súbory členov nie je možné editovať priamo v aplikácií. Takže študent bude môcť pracovať len s množinou členov, ktoré mu budú pridelené v rámci daného súboru členov. Každý súbor členov obsahuje presne definovaný počet jednotlivých členov, ktoré majú dva alebo štyri vstupy. Po vybratý nejakého súboru členov sa zobrazí pracovná plocha, na ktorej budú uložené jednotlivé logické členy. Používateľ bude len môcť prepojiť jednotlivé vstupy a výstupy podľa zadania, ktoré dostane. Keďže v prvej časti projektu už bola implementovaná funkcia *Drag & Drop* a tá práca nevyšla nazmar, bola ponechaná možnosť posúvania členov. Táto funkcia je vhodná aj v prípade, že používateľ potrebuje nejakým spôsobom popresúvať členy resp. nejaký člen zavadzia pri prepájaní členov čiarami. Po prepojení vstupov a výstupov je možné zapnúť simulácia. Každý projekt sa dá uložiť a následne otvoriť.

Bližšia špecifikácia jednotlivých možností a vlastností aplikácie je uvedená v kapitole 3, ktorá popisuje špecifikáciu riešenia, vlastnosti aplikácie, nefunkcionálne požiadavky a grafické rozhranie.

6.2 Doplnenie návrhu riešenia

Návrh riešenia priamo vychádza zo špecifikácie riešenia. Preto v tejto časti sú spomenuté len zmeny alebo doplnenie riešenia. Návrh riešenia je zložený z modulu grafického rozhrania, modulu logických súčiastok a modulu podporných funkcií.

6.2.1 Modul logických súčiastok

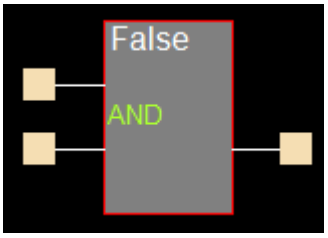
Tento modul bol pôvodne navrhovaný ako knižnica logických členov, keďže sa predpokladalo, že na pracovnú plochu budú jednotlivé logické členy pridávané postupne podľa potreby. Pôvodný návrh bol zmenený na súbory logických súčiastok. To znamená, že sú vytvorené súbory, ktoré obsahujú presne definovaný počet logických členov. Na pracovnú plochu sú pridávané jednotlivé súbory, ktoré študent nemôže editovať. Na modifikovanie alebo vytvorenie súborov bude vytvorená samostatná aplikácia, ktorú budú mať k dispozícií len pedagógovia. Táto aplikácia bude opísaná v kapitole Opis realizácie.

Súbory členov obsahujú kombinácie nasledovných logických členov - AND, NAND, OR, NOR, XOR, NOT, zábrana, implikácia, ekvivalencia, zdroj 0 a zdroj 1. Funkčná stránka jednotlivých členov je popísaná v kapitole Analýza problémovej oblasti.

6.2.2 Modul grafického používateľského rozhrania

V tomto module došlo k minimalizácií používateľských úkonov ako už je naznačené v kapitole Doplnenie špecifikácie. Používateľovi po vybraný súboru členov budú rozmiestené jednotlivé členy, ktorú bude stačiť správne prepojiť. Pôvodný návrh obsahujúci pracovnú plochu, knižnicu logických členov a riadiacu lištu ostáva viac menej nezmenný. Jedna z hlavných zmien nastala v časti knižnica logických členov t.j. knižnica neobsahuje logické členy, ale súbor logických členov.

Ďalšia zmena, ktorá bola vykonaná v tomto module je zmena grafickej reprezentácie logických členov. Pôvodný návrh bol realizovaný pomocou možností programovacieho jazyka, ktorý umožňuje jednoduchú grafickú reprezentáciu. Ako príklad uvádzame grafickú reprezentáciu logického člena AND.



Obrázok č.17: Pôvodný návrh.

Každý logický člen je na pracovnú plochu vykreslený ako obrázok a podľa hodnoty, ktorú má na výstupe je zafarbený na červeno (logická nula) alebo zeleno (logická jednotka). Taktiež aj jednotlivé čiary svojou farbou zobrazujú ako hodnota nimi prechádza. Biela resp. šedá farba značí nezapojený člen. To znamená, že daným členom neprechádza žiadna hodnota. Pre lepší spôsob hľadania chyby v zapojení bola doplnená aj žltá farba, ktorá znamená konflikt t.j. pripojenie dvoch alebo viacerých čiar na jeden vstup. Tento návrh je graficky prijateľnejší, efektívnejší a mení celý dojem pri práci. Ako príklad uvádzame grafickú reprezentáciu logického člena AND.

Obrázok č.18: Súčasný návrh.

Taktiež boli vykonané zmeny pri vykresľovaní čiar. Mierka mriežky pracovnej plochy bola zmenšená, aby poskytovala efektívnejšiu prácu s členmi a čiarami (spojmi). Vďaka menšej mriežke je možné vedľa seba uložiť viacej spojov. Prepájanie členov má dva režimy. Prvý režim je voľné kreslenie, kedy si používateľ môže určiť ako bude ťahať čiary. Druhý režim je kontrolované kreslenie. To znamená, že čiary sú kreslené len horizontálne alebo vertikálne tak, aby bol zachovaný pravý uhol. Tento režim sa zapína stlačením a držaním klávesy *shift*. Keď je stlačený *shift*, program vykresľované čiary zarovnáva na mriežku. Tento doplnok funkcionality pomáha používateľom a umožňuje efektívnejšiu prácu s programom.

Ďalšia zaujímavá možnosť ovládania spočíva v rýchlom pohybe v rámci plátna. Pri držaní pravého tlačidla myši je možné sa rýchlo presúvať v rámci plátna pomocou pohybovania

myšou do strán. Táto možnosť prináša jednoduchšiu a efektívnejšiu prácu s programom pri zapájaní logických členov.

6.2.3 Modul podporných funkcií

Základná funkcionalita programu zostáva nezmenený. Avšak nastali zmeny na riadiacej lište, ktorá bola upravená tak, aby obsahovala len potrebné prvky. Zbytočné položky boli odstránené. Taktiež bola odstránená položka knižnica. Táto zmena bola nevyhnutná, keďže došlo k zmene pridávania a manipulácií prvkov. Ako už bolo spomenuté, tak prvky sú pridávané pomocou samostatného okna, ktorá sa zobrazí na začiatku pri spustení programu.

Po úprave riadiaca lišta obsahuje nasledovné voľby:

- **Súbor**
  - **Nový** – vytvorenie nového projektu.
  - **Otvoriť** – načítanie uloženého projektu.
  - **Uložiť** – uloženie projektu.
  - **Uložiť ako obrázok** – schému na pracovnej ploche uloží ako jpg súbor.
  - **Ukončiť** – vypnutie programu.
- **Spustenie simulácie** – Umožní začatie simulovania toku logických hodnôt.
- **Pomoc**
  - **O programe** – Krátke informácie o programe a ako ho používať.

Druhá úroveň riadiacej lišty je nasledovná:

- **Nový** – vytvorenie nového projektu.
- **Editácia** – editovanie schémy zobrazenej na pracovnej ploche.
- **Mazanie** – vymazanie čiar nakreslených na pracovnej ploche.
- **Plátno** – zmena veľkosti plátna (pracovnej plochy).

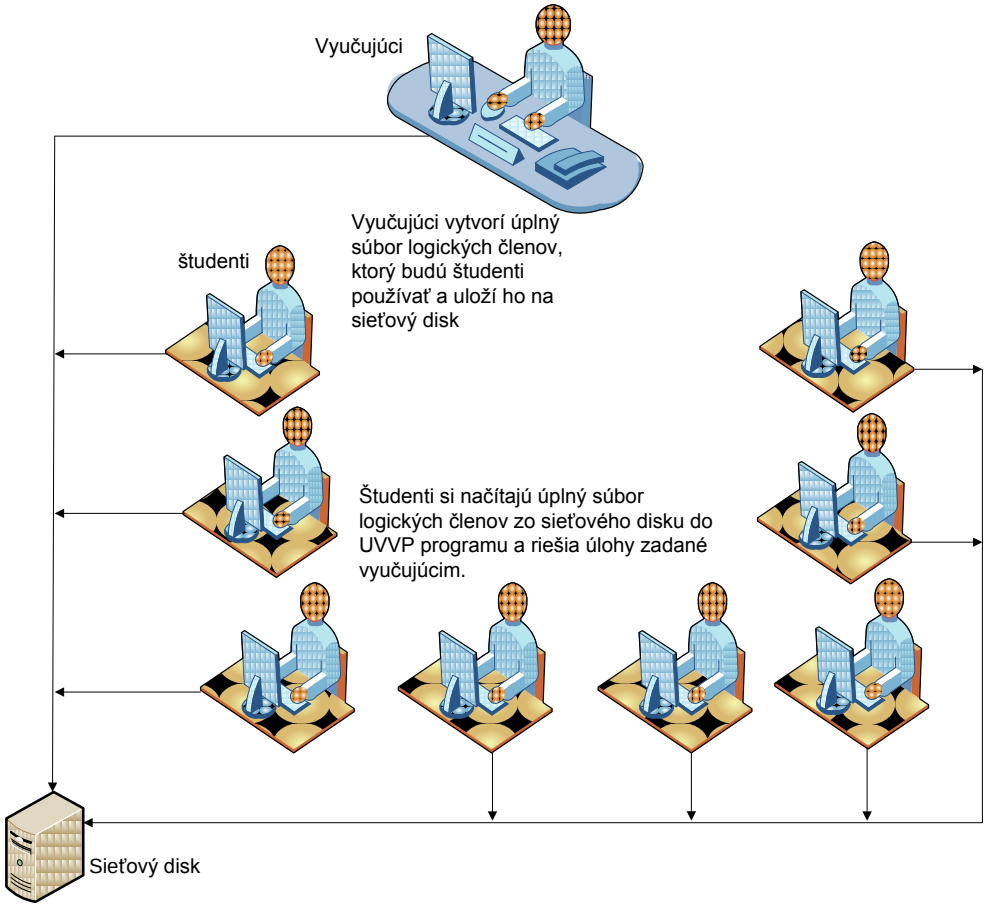
6.3 Návrh systému

Táto kapitole obsahuje architektúru systému, fyzický model údajov riešenia a použité algoritmy.

6.3.1 Architektúra systému

Práca s UVVP je jednoduchá. Správca počítačovej učebne nainštaluje na počítače UVVP produkt. Následne na počítačoch v UVVP produkte nastaví cestu k súboru s úplnými súbormi logických členov (či už počas inštalácie alebo po). Na tento sieťový disk umiestni súbor s úplnými súbormi logických členov.

Počas hodiny si študenti otvoria program UVVP. Program načíta zo sieťového disku množinu úplných súborov logických členov. Študent vyberie ktorú množinu chce použiť podľa pokynov vyučujúceho a následne rieši zadané úlohy v UVVP programe. Pokiaľ vyučujúci potrebuje zmeniť niektorú úplnú množinu logických členov alebo ju upraviť, spustí špeciálny program, ktorý edituje súbor obsahujúci úplné súbory logických členov (súbor na sieťovom disku). Vykonané zmeny uloží a všetci študenti majú k dispozícii upravenú množinu úplných súborov logických členov, s ktorou môžu pracovať na cvičeniach. Architektúra systému je na nasledujúcom obrázku.



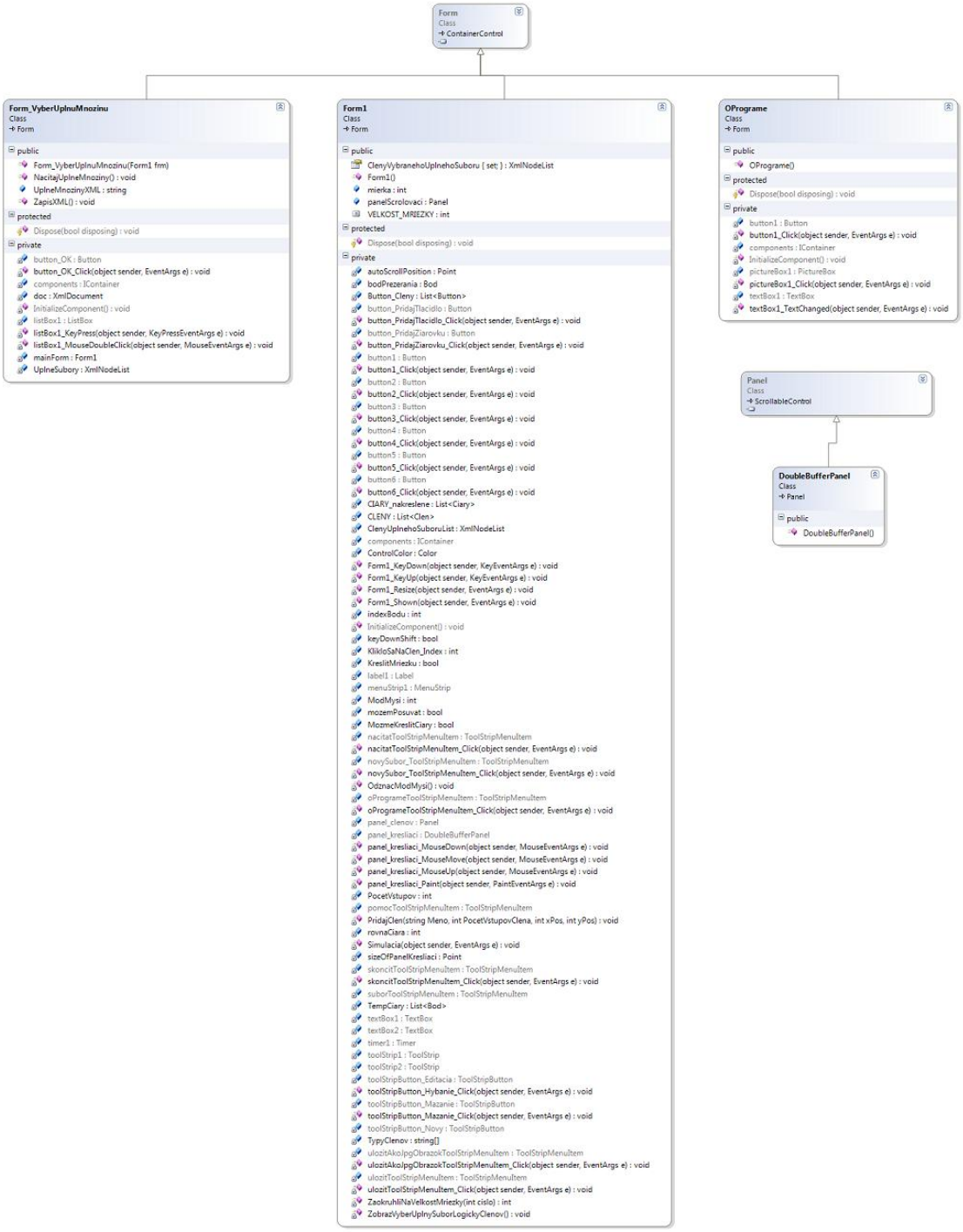
Obrázok č.19: Architektúra systému.

6.3.2 Fyzický model údajov systému

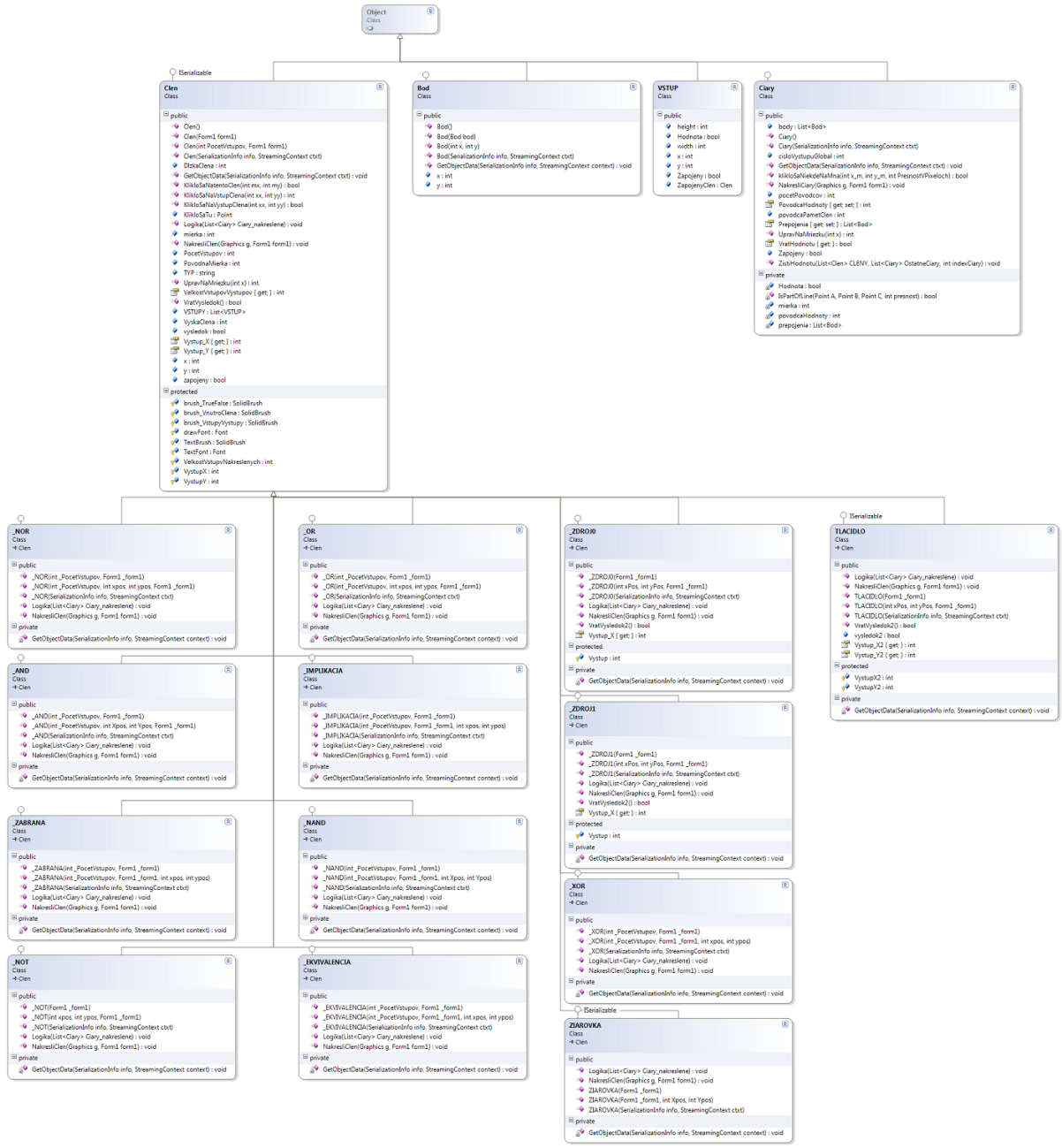
Na nasledujúcich dvoch obrázkoch sa nachádza model tried programu UVVP. Na prvom obrázku sa nachádzajú triedy - Form_VyberUplnuMnozinu, Form1 a OPrograme odvodené od triedy Form.

- **Form_VyberUplnuMnozinu** - je trieda, ktorá je zodpovedná za výber úplnej množiny členov.
- **Form1** - táto trieda je hlavná, zabezpečuje samotný beh program UVVP.
- **OPrograme** - táto trieda zabezpečuje zobrazenie krátkej informácie o programe.

Na druhom obrázku sa nachádzajú triedy Clen odvodenej od triedy Object a jednotlivé triedy členov odvodené od tejto triedy. Ďalej sa tam nachádzajú triedy Bod, Vstupy a Ciary odvodené od triedy Object.



Obrázok č.20: Model tried.



Obrázok č.21: Model tried.

6.3.3 Návrh algoritmov spracovania

Pri spracovaní jednotlivých úloh, sme museli navrhnúť niekoľko algoritmov na ich spracovanie. Jedným z hlavných, bol algoritmus na rozpoznávanie hodnôt nakreslených čiar. Keďže sú čiary vykresľované po mriežke, čiara pozostáva z pretínajúcich bodov mriežky. Okrem toho každá čiara obsahuje body, ktoré určujú miesta prepojenia s ostatnými čiarami respektíve členmi. Spájanie objektov čiarami sme vyriešili jednoducho. Či je nejaký

člen pripojený na čiaru zisťujeme jednoducho. Zisťujeme či sa niektorý bod, z bodov prepojenia, prekrýva s výstupom niektorého objektu. Ak áno, čiare priradíme logickú hodnotu výstupu tohto objektu. Pokiaľ sa tento bod prekrýva s nejakou inou čiarou, tieto čiary sú prepojené. Jednotlivé čiary majú pôvodcu hodnoty, čo je čiara ktorá určuje ich hodnotu. Ak je čiara napojená na výstup tak jej pôvodca hodnoty je -1, čo znamená, že jej pôvodca hodnoty je ona sama. Ak sa má určiť pôvodca hodnoty nejakej čiary, tak sa najskôr zistí, aké čiary s ňou susedia, a ďalej sa zistia ich pôvodcovia hodnôt, potom sa zistia pôvodcovia hodnôt pôvodcov hodnôt a tak ďalej, až kým sa nedôjde na čiary, ktoré si samy určujú hodnotu, alebo nie sú zapojené. Hodnota čiary sa určí na základe hodnoty čiary s indexom pôvodca hodnoty. Ak je týchto pôvodcov viacej (v prípade, že čiara má napríklad prepojenie z viacerých vstupov) , tak sa čiara označí žltou farbou (odporúča sa na zmazanie, ale je možné zmazať aj inú čiaru, následkom čoho sa hodnoty jednotlivých hrán prepočítajú) a odpojí sa. Ak čiara nie je napojená tak sa označí bielou farbou. Ak má logickú hodnotu 1 tak zelenou, ak nula tak červenou.

Ďalším dôležitým algoritmom bola simulácia obvodu. Rozhodli sme sa, že obvod sa bude simulovať stále – to znamená, že zmeny v obvode sa prejavia hneď po uskutočnených zmenách v návrhu. Preto simulácia beží v samostatnom vlákne. Simulácia obvodu sa teda, vykoná vždy za pár milisekúnd. Výhoda tohto riešenia je tá, že je teoreticky možné zapojiť aj obvod s nestálou hodnotou. Simulácia je riešená tým spôsobom, že sa prejde v cykle každý jeden objekt a nastavia sa mu príslušné výstupy. Toto riešenie sme videli ako najjednoduchšie a najefektívnejšie.

Riešili sme taktiež aj pohyb objektov po schéme. Keďže sme zvolili mriežku rozhodli sme sa, že objekty sa budú môcť pohybovať len po mriežke. Toto sme zabezpečili tým, že keď sa myšou chytí objekt, zistí sa poloha myši a tá sa upraví na najbližší násobok mriežky v smere x a taktiež aj v smere y. Následne sa objektu nastavia súradnice na násobok mriežky a ten sa na týchto súradniciach vykreslí.

Ďalej sme riešili vnútornú reprezentáciu objektov a ich uloženie a načítanie buď z predtým uloženého súboru alebo zo súboru, v ktorom sú uložené úplné súbory logických členov.

6.4 Ohraničenia

Aplikácia je určená na vzdelávacie účely. Primárne je určená pre študentov na vypracovanie zadaní v predmete Logické obvody. Tak ako je spomenuté v špecifikácií a návrhu riešenia, aplikácia má simulovať hardvérový verifikačný panel logických obvodov. Tomuto zadaniu je aj prispôsobené vypracovanie.

6.5 Priority riešenia

Hlavnou prioritou celého riešenia je vytvoriť použiteľný program, ktorý bude možné použiť na vzdelávacie účely. Čiastkové priority sú:

- vytvorenie funkcionálnej stránky programu,
- implementovanie algoritmov spracovania,
- vytvorenie grafického rozhrania,
- interpretácia grafického rozhrania,
- vytvorenie úplného súboru členov,
- doplnenie podporných funkcií,
- vytvorenie dokumentácie riešenia.

6.6 Výber implementačného jazyka a prostredia

Ako už bolo spomenuté v návrhu riešenia program je vyvíjaný v jazyku C# na platforme Microsoft .Net Framework v prostredí Microsoft Visual Studio. Keďže výsledný produkt bude použitý vo výučbe na školských počítačov, kde je nainštalovaný operačný systém Microsoft Windows, tak sme sa rozhodli pre toto prostredie a programovací jazyk.

6.7 Opis realizácie

Táto kapitola obsahuje opis realizácie prvkov produktu a doplnenia oproti pôvodnému návrhu a taktiež opis implementácie jednotlivých modulov.

6.7.1 Implementácia jednotlivých modulov

Jednotlivé moduly boli implementované v programe UVVP. Modul logických súčiastok bol implementovaný pomocou súborov logických súčiastok, ktoré môžu byť vytvára a modifikované pomocou XML editora, ktorý je opísaný v nasledujúcej podkapitole. Modul grafického používateľského rozhrania je implementovaný pomocou programu UVVP a modul podporných funkcií takisto.

6.7.2 Doplnenie oproti návrhu

Celkový návrh riešenia bol rozšírený o externú aplikácie XML editor, ktorá má za úlohu vytvorenie súboru *.xml*. Tento súbor obsahuje súbor logických členov ako už bolo spomínané v kapitole Doplnenie návrhu riešenia. Táto aplikácia je určená pre pedagógov, ktorí budú môcť modifikovať jednotlivé súbory členov a tak ich pripraviť pre študentov. Taktiež je možné vytvoriť nový súbor členov a zahrnúť ho už k existujúcim.

Náš XML editor je implementovaný pre také isté prostredie ako aplikácia na verifikáciu logických členov. To znamená, že pracuje pod operačným systémov Microsoft Windows. Editor je realizovaný za pomoci otvoreného projektu, ktorý sa nazýva SourceGrid [13]. SourceGrid je .NET platforma pre Windows, ktorá umožňuje vytvorenie alebo zmenu dát vo forme tabuliek. Projekt je kompatibilný v prostredí Microsoft Framework .NET 1.1 a realizovaný pomocou programovacieho jazyka C#. Na fungovanie využíva knižnicu SourceLibrary.dll vo verzii 1.2.0.0. SourceGrid umožňuje jednoduché a efektívne implementovanie hocijakých dát vo forme tabuľky. Tento fakt perfektne slúži pre potreby nášho projektu pri vytváraní a modifikovaný súboru *.xml*, ktorý obsahuje súbory logických členov. Za pomoci knižnice SourceGrid sme vytvorili jednoduchý, ale funkčne veľmi efektívny XML editor.

Ďalším doplnením je možnosť pridať neobmedzení počet vstupov a výstupov pre logické členy. Základné nastavenie je, že logické členy majú dva alebo štyri vstupy a jeden výstup. Avšak programátorsky bolo správanie doplnené a rozšírené. Toto doplnenie je urobené v rámci bonusového riešenia, ktoré predstavuje rozšírenie zadania. V prípade potreby rozšíriť možnosti aplikácie, stačí urobiť jednoduchšie úkony na integrovanie viacerých vstupov a výstupov.

Tretie doplnenie oproti návrhu je zamerané na pracovnú plochu resp. plátno. Veľkosť mriežky, ktorá je integrovaná do plátna bola zmenšená. Tým sme dosiahli väčší priestor na kreslenie čiar. Aby si mohol používateľ zmeniť veľkosť obrazovky, pridali sme možnosť meniť veľkosť plátna. Toto umožní používateľovi efektívnejšiu prácu so schémou.

7 Overenie výsledku

Implementované riešenie bolo overené pod operačným systémom Windows. Pri testovaní sa vyskytli problémy so stabilitou, či už pri samotnej aplikácii alebo XML editore. Po hlbšej analýze boli príčiny problémov odhalené a odstránené.

Všetky navrhnuté funkcie boli už počas implementácie overené, aby sme si mohli byť istí, že za každých okolností sa vykonáva presne to, čo požadujeme. Najväčší dôraz pri testovaní sme kládli na modul grafického používateľského rozhrania. Cieľom bolo zabezpečiť, aby sa za žiadnych okolností nevyskytli neočakávané výsledky. To bolo možné len pri dôkladnej simulácii práce užívateľa s programom a kontrolovaním ladiacich výpisov. Na základe nich sme predovšetkým sledovali, či na logické súčiastky sú zapojené tie súčiastky, ktoré požaduje používateľ a privedené také logické hodnoty, aké sa očakávajú. Ďalšou dôležitou úlohou pre testovanie bolo zabezpečiť, aby sa zmeny na verifikačnom paneli prejavili okamžite a boli správne graficky reprezentované.

Po overení výsledku môžeme konštatovať, že aplikácia pod operačným systémov Windows pracuje stabilne a všetky chyby, ktoré sa vyskytli pri implementácii boli odstránené a aplikácia spĺňa všetky požiadavky, ktoré na ňu boli kladené.

8 Záznamy o používaní systému

Systém bol používaný počas celého procesu vývoja. Jednotlivý členovia tímu pravidelne testovali jeho funkčnosť a to, či si zachováva požadované správanie aj v nepredpokladaných situáciách. Pri tomto testovaní sa prišlo na niekoľko závažných problémov, ktoré sa delegovali členom tímu zodpovedných za implementáciu. Spätne môžeme zhodnotiť, že sa nám podarilo odstrániť všetky závažné chyby, ktoré by mohli spôsobiť nepoužiteľnosť výsledného produktu alebo by boli nežiaduce z pohľadu používateľa.

Išlo predovšetkým o chyby, ktoré boli spôsobené neštandardným napájaním čiar. V jednom prípade sa signál nešíril zo zapojenej vetvy, ak bola vetva v slučke. V druhom prípade, ak sa zapojil signál už do zapojenej vetvy a mal opačnú hodnotu, ako vetva do ktorej sa zapájal, rozšírila sa doň táto hodnota. Oba problémy boli vyriešené implementovaním zložitejšieho algoritmu prehľadávania grafu, ktorý prehľadáva graf od konca a prechádza všetky zapojenia až k zdroju. Tam sa pozrie akú má zdroj hodnotu a prefarbí celú časť vetvy, ktorú prechádzal na zodpovedajúcu farbu. V druhom prípade sme implementovali funkciu, ktorá zobrazí čiaru spájajúcu časť jedného obvodu s druhým, pričom tieto časti majú opačné hodnoty. Na grafickú reprezentáciu sme použili žltú farbu. Týmto program signalizuje užívateľovi zlé zapojenie obvodu.

Niektoré menšie problémy sa vyskytli aj pri pridávaní grafických reprezentácií jednotlivých členov úplného súboru. Problémy so zlým posunutých zapojení členov sa však rýchlo vyriešili ich grafickou úpravou.

Posledný väčší problém, ktorý sme riešili bol s nedostatkom miesta pre kreslenie čiar. Jedna z ponúkaných možností bola zväčšenie granularity kresliaceho plátna. Toto riešenie však zo sebou prinášalo ďalšie problémy, ako napríklad ten, že pri opätovnom zmenšení granularity sa čiary robené vo väčšej granularite nezachovávali, ale prispôsobovali sa meniacej sa mriežke. Potom vznikali situácie, že sa čiary, ktoré boli blízko seba spájali a vytvárali sa nezmyselné prepojenia. Tento spôsob riešenia sme zavrhli a namiesto toho sme implementovali variabilnú veľkosť plátna a možnosť pohybu jednotlivých členov. Takto sa zabezpečilo to, že pre kreslenie prepojení, bude mať používateľ vždy dosť miesta.

9 Záver

Predmety Tímový projekt I,II boli pre mnohých z nás prvou skúsenosťou s tvorením zložitejšieho softvérového riešenia vo väčšom tíme. Myslíme si, že sme sa úlohy zhostili dobre a vďaka výbornej spolupráci sme vytvorili kvalitný produkt, ktorý má reálnu šancu nájsť si svoje miesto v pedagogickom procese.

9.1 Čo sme nestihli

Vzhľadom na pôvodnú špecifikáciu po konzultácii s našim vedúcim projektu sme upustili od zobrazovania logických funkcií v podobe karnaughových máp. Taktiež náš systém nebude zobrazovať konečné automaty typu mealy, moore, ich vzájomnú konverziu ani ich redukciu. Ďalej sme vypustili možnosť prevodu výrazov na ich minimálnu skupinovú disjunktívnu normálnu formu a ich zobrazenie do karnaughových máp aj automatickú štruktúrnu syntézu synchrónnych respektíve asynchrónnych sekvenčných obvodov. Riešili sme teda iba pôvodné zadanie, tak ako bolo navrhnuté, nie rozšírenú verziu, ktorú sme uvideli v ponuke.

9.2 Čo sme sa naučili

Pri vypracovávaní zadania pre predmet Tímový Projekt sme si osvojili základy práce v tíme. Naučili sme sa spolupracovať na projekte väčšieho rozsahu. Na pravidelných tímových stretnutiach sme zlepšili svoje komunikačné schopnosti a schopnosť dohodnúť sa na konkrétnych riešeniach. Pre niektorých z nás, bola nová skúsenosť aj práca s jazykom C# a rôznymi programami na úpravu obrázkov. Taktiež sme sa zdokonalili vo vedomostiach, ktoré sme už predtým používali pri vypracovávaní niektorých projektov, ako napríklad špecifikácia a návrh systému. Najviac však bolo badateľné zlepšenie v oblasti vzájomnej spolupráce a komunikácii v rámci tímu. Tieto zložky boli nevyhnutné na úspešné dokončenie nášho projektu Virtuálny verifikačný panel logických obvodov.

10 Použitá literatúra

[1] Doc. Ing. N. Frištacký, CSc.; Doc. Ing. J. Kolenička, CSc.; Doc. Ing. M. Kolesár, CSc.: *Logické systémy - Kombinačné obvody.* júl 1986. Bratislava. Editačné stredisko SVŠT. [cit. 2009-10-24].

[2] Doc. RNDr. Jana Galanová, PhD.; RNDr. Peter Kaprálik, PhD.; Mgr. Marcel Polakovič, PhD.: *Logické systémy.* [cit. 2009-10-31].

[3] Frištacký N., Kolesár M., Kolenička J., Hlavatý J: *Logické systémy,* 1986. Alfa – Vydavateľstvo technickej a ekonomickej literatúry Bratislava. [cit. 2009-10-24].

[4] Pivarček Ján: Virtuálny verifikačný panel s členmi XOR a OR, Bakalárska práca, FIIT STU 2009.

[5] Sivák Marek: Hradlá a hradlové štruktúry, podpora výuky – aplikácia pod OS Windows, Bakalárska práca, FIIT STU 2009.

[6] Palaj Tomáš: Virtuálny verifikačný panel s členmi AND-NOR a NAND, Bakalárska práca, FIIT STU 2009.

[7] Pilarčík Viliam: Virtuálny verifikačný panel s členmi NOR, Bakalárska práca, FIIT STU 2009.

[8] Arase, Kazuhiko: Simcir the circuit simulator. [Online]. 6. júla 2000. [cit. 2009-11-07]. Dostupný z WWW: http://www.d-project.com/simcir/index.html.

[9] CommTec: LOGiX - Simulation of Logic Circuits. [Online]. 04. júl 2005. [cit. 2009-11-07]. Dostupný z WWW: http://www.simtel.net/product.php%5Bid%5D90288%5Bcid%5D86%5BSiteID%5Dsimtel.net.

[10] Komppa: Atanua Logic Simulator. [Online]. [cit. 2009-11-07]. Dostupný z WWW: http://sol.gfxile.net/atanua/.

[11] Knaian, Ara. Digital Simulator. [Online]. Massachusetts Institute of Technology, 1995. [cit. 2009-11-07]. Dostupný z WWW: http://www.mit.edu/people/ara/ds.html.

[12] Tetzl, Andreas: Logic Simulator. [Online]. 22. novembra 2008. [cit. 2009-11-07]. Dostupný z WWW: http://www.tetzl.de/java_logic_simulator.html.

[13] Icardi D.: SourceGrid - Open Source C# Grid Control. [Online]. marec 2004. [cit. 2010-03-28]. Dostupný z WWW: http://www.codeproject.com/KB/grid/csharpgridcontrol.aspx

11 Príloha

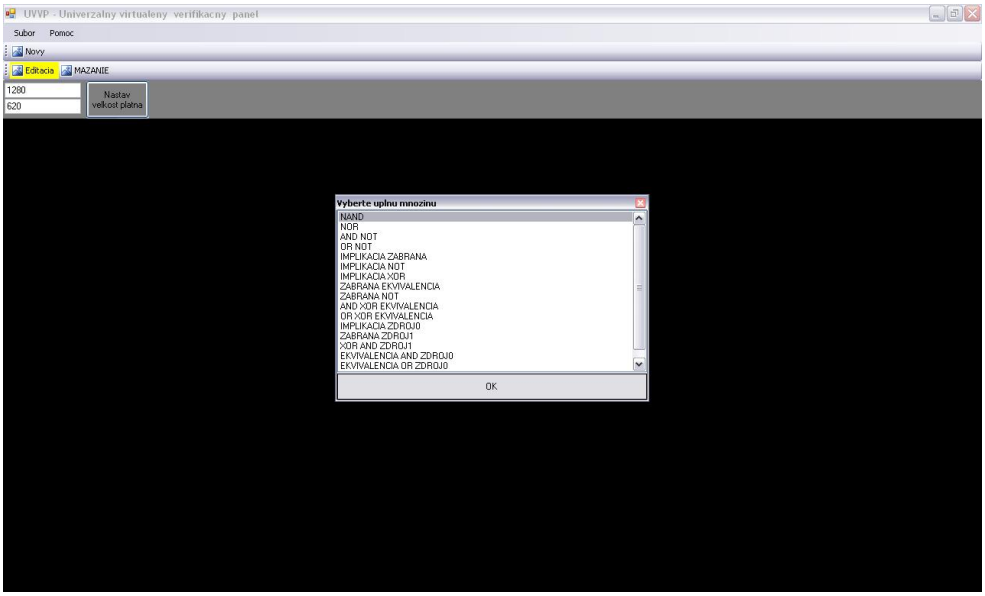
V tejto časti dokumentu sa nachádzajú prílohy ako napríklad - používateľská príručka, systémová príručka a podobne.

11.1 Príloha A: Používateľská príručka

Používateľská príručka obsahuje jednoduchý návod ako využívať aplikácie Univerzálny virtuálny verifikačný panel (uvvp.exe) a XML editor (UVVP_XML_EDITOR. exe).

11.1.1 Univerzálny virtuálny verifikačný panel

Po spustení aplikácie sa zobrazí úvodné okno, ktoré umožňuje výber jednej úplnej množiny súborov logických členov.

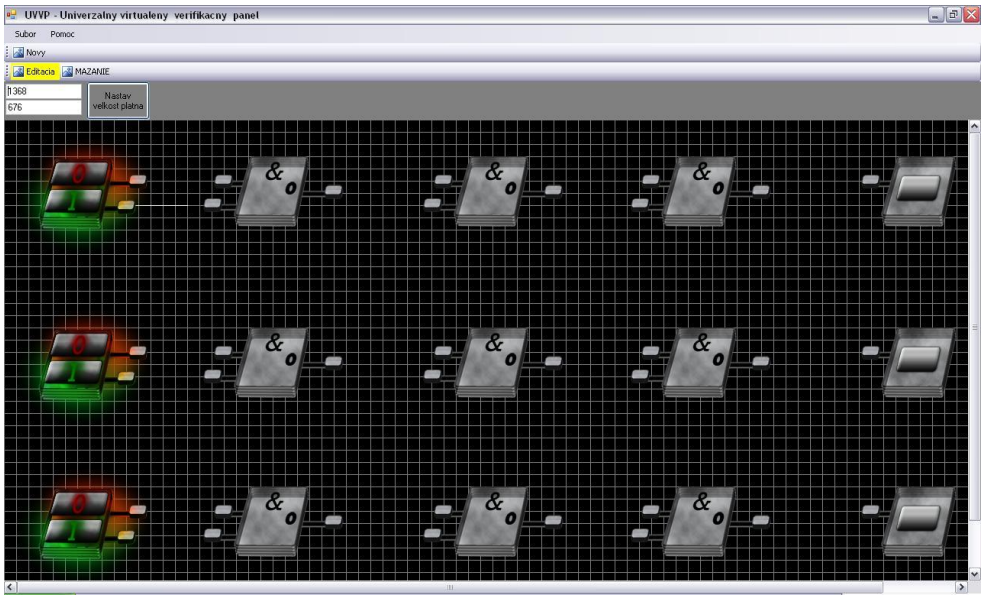


Obrázok č.22: Úvodná obrazovka.

Po vybratí jednej množiny sa načíta verifikačný panel, ktorý obsahuje vstupné tlačidlá, výstupné žiarovky a logické členy patriace do danej množiny.

Obrázok č.23: Načítaná úplná množina.

Po načítaní množiny môže užívateľ začať používať aplikáciu. Spojovacie čiary medzi jednotlivými súčiastkami sa kreslia pomocou ľavého tlačidla myši, kedy pri kreslení je potrebné držať toto tlačidlo stlačené. Čiary sa kreslia po virtuálnej mriežke, ktoré sa zobrazí počas kreslenia. Druhý režim kreslenia je kontrolované kreslenie. To znamená, že čiary sú kreslené len horizontálne alebo vertikálne tak, aby bol zachovaný pravý uhol. Tento režim sa zapína stlačením a držaním klávesy *shift*. Keď je stlačený *shift*, program vykresľované čiary zarovnáva na mriežku. Pri kreslení je potrebné byť v režime editácia. Na obrázku vyznačené žltou.

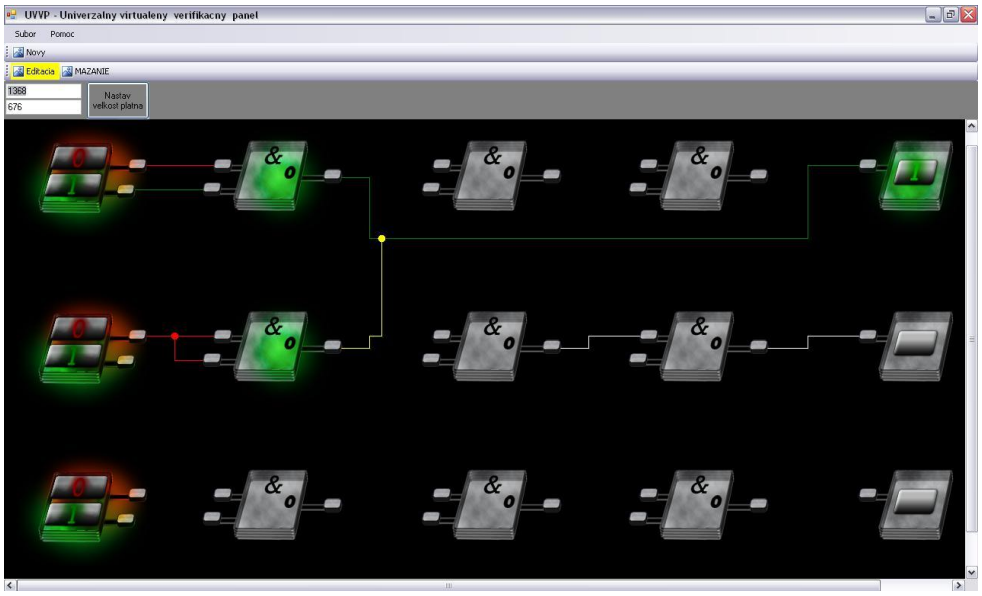


Obrázok č.24: Virtuálna mriežka.

Kreslené čiary môžu mať 4 druhy farieb:

- **červená:** na čiare sa nachádza logická hodnota 1.
- **zelená:** na čiare sa nachádza logická hodnota 0.
- **biela:** na čiare nie je žiadna logická hodnota.
- **žltá:** nastal konflikt, čiary nie sú správne prepojené.

Nasledujúci obrázok ukáže všetky 4 typy čiar a tiež simuláciu zapojenia. Simulácia prebieha neustále, hodnoty sa menia na základe zmeny zapojenia, resp. hodnôt vstupných signálov.



Obrázok č.25: Ukážka simulácie.

Ak užívateľ omylom nakreslí čiaru na nesprávne miesto, je možné ju odstrániť. Užívateľ sa najskôr prepne do režimu mazanie. V tomto režime sa mu zmení kurzor myši. Čiara sa odstráni spôsobom, že na ňu užívateľ klikne ľavým tlačidlom myši.

Logické členy nie sú na ploche uložené napevno. Užívateľ si ich môže ľubovoľne presúvať po ploche. Pre tento úkon treba vybrať člen ľavým tlačidlom myši a pri stálom držaní ho ťahať po ploche na želané miesto.

Pri obrazovkách s menším rozlíšením sa nemusia všetky členy zmestiť na plochu. Preto je potrebné plochu posúvať. Je to možné dosiahnuť klasicky pomocou bočných posuvných líšt

alebo myšou. Myšou sa to realizuje stlačením pravého tlačidla v ktoromkoľvek mieste na ploche, jeho držaní a ťahom myši do požadovaného smeru.

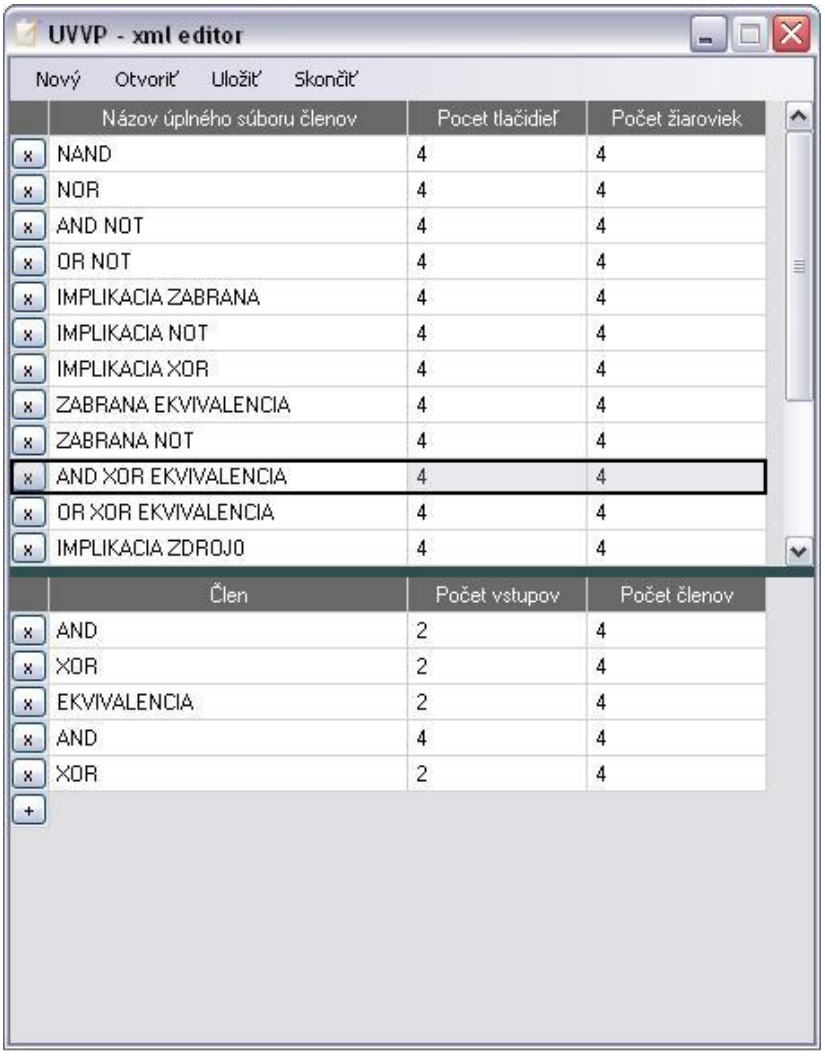
Ak je veľkosť kresliaceho plátna malá, je možné ho zväčšiť pomocou nastavenia, ktoré sa nachádza v hlavnom okne na kresliacou plochou. Po načítaní úplnej množiny je veľkosť plátna na najmenšej hodnote. Túto hodnotu nie je možné zmenšiť.

Aplikácia obsahuje menu, ktoré obsahuje položky:

- **Nový:** načítanie novej úplnej množiny prvokov.
- **Uložiť ako jpg obrázok:** uloženie nakreslenej schémy ako jpg obrázok.
- **Uložiť:** uloženie nakreslenej schémy do súboru, ktorý umožní jej znovu načítanie.
- **Načítať:** načítanie predtým uloženej schémy.
- **Skončiť:** ukončenie programu.

11.1.2 XML editor

Používané úplne množiny prvkov sa nachádzajú v externom xml súbore. Súbor xml obsahuje názov množiny, počet vstupov, výstupov a súčiastok. Ak by nami navrhnuté obsahy množín nevyhovovali, je možné ich ľahko zmeniť pomocou daného XML editora. Na nasledujúcom obrázku je zobrazené okno aplikácie.



Obrázok č.26: XML Editor.

V hornej časti aplikácie sa nachádza zoznam jednotlivých úplných množín spolu s počtom tlačidiel (vstupov) a žiaroviek (výstupov). V spodnej časti sa pre konkrétnu množinu zobrazujú typy členov, ktoré obsahuje. Pre jednotlivý súbory sa definujú počty vstupov a tiež počet členov. Počtom členov sa myslí počet, ktorý sa zobrazí na ploche panela v jednom

stĺpci pod sebou. Každý člen predstavuje jeden stĺpec na ploche panela. Pridávanie ďalších členov sa realizuje pomocou tlačidla plus a odstraňovanie pomocou tlačidla x.

Menu aplikácie obsahuje položky:

- **Nový:** vytvorí nový XML súbor.
- **Otvoriť:** otvorí už existujúci súbor, ktorý môže užívateľ pohodlne editovať.
- **Uložiť:** umožní uložiť vykonané zmeny.
- **Skončiť:** skončí program.

11.2 Príloha B: Systémová príručka

Systémová príručka obsahuje informácie o systémových požiadavkách, čo sa týka podporovaných operačných systémov a tiež hardvérových požiadaviek. Ďalej obsahuje stručné informácie týkajúce sa inštalácie a spustenia aplikácie.

11.2.1 Systémové požiadavky

Minimálne systémové požiadavky, ktoré sú potrebné na bezproblémové fungovanie aplikácie:

- Podporované operačné systémy: Windows 98, Windows 2000, Windows Server 2003 Windows XP, Windows Vista, Windows Server 2008, Windows 7.
- Procesor: Minimum 400MHz Pentium, resp. jeho ekvivalenty. Odporúčané 1GHz Pentium, resp. jeho ekvivalenty.
- Operačná pamäť: Minimum 96MB, odporúčané 256MB.
- Miesto na pevnom disku: 280 MB pre verzie x86, 610 MB pre verzie x64.

11.2.2 Inštalačná príručka

Na úspešné spustenie aplikácií je potrebné spĺňať systémové požiadavky, čo sa týka operačného systému a tiež hardvéru. Program bol vyvinutý na platforme Microsoft .Net Framework 2.0. Preto pred spustením programu je potrebné nainštalovať balík knižníc. Ak sa balík na danom počítači nenachádza, je možné ho voľne stiahnuť zo stránok Microsoftu. Odkaz na stránku, kde sa balík nachádza:
http://www.microsoft.com/downloads/details.aspx?FamilyID=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=en

Po nainštalovaní balíka môžeme pristúpiť spusteniu aplikácií. Aplikácie sú realizovaná ako skompilovaný spustiteľný .exe súbor. Preto nie je potrebná žiadna inštalácia.