Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológii

Študijný program: Počítačové a komunikačné systémy a siete

Univerzálny virtuálny verifikačný panel logických obvodov

(Tímový projekt - Tím č.4)

Bc. Dominik Macko

Bc. Mário Patoprstý

Bc. Martin Popelka

Bc. Miroslav Siebert

Bc. Martin Valko

Vedúci tímového projektu: Ing. Katarina Jelemenská, PhD.

Ročník: 1

Štúdium: Inžinierske

Máj 2010

Obsah

1. Úvod

V dnešnej dobe sa počítačová technika stala neoddeliteľnou súčasťou života každého z nás. Či už sa jedná o samotný počítač, vnorené počítačové systémy, digitálny prenos informácií, alebo o mobilné zariadenia všetky majú na svojej najnižšej hardvérovej úrovni práve logické členy a využívajú booleovskú logiku, bez ktorej by tieto zariadenia neboli funkčné. Práve z tohto dôvodu je potrebné sa zaoberať aj touto časťou zariadení, porozumieť im, vedieť ich používať a zdokonaľovať ich funkcionalitu.

V tejto práci postupne analyzujeme funkčnosť týchto obvodov, samotných logických členov a booleovskej logiky. V návrhu na základe týchto poznatkov navrhujeme univerzálny virtuálny verifikačný panel logických členov, ktorý slúži práve na simuláciu zapájania určených logických členov, overovanie ich funkčnosti a správania sa. Univerzálnosť panelu spočíva vo voľbe používateľa, ktorú úplnú funkčnú množinu členov chce použiť. V samotnej implementácií vytvárame konkrétny počítačový program určený pre osobné počítače, ktorý by spĺňal naše požiadavky a nahradil fyzické zapájanie týchto obvodov a ich testovanie. V časti testovania a overovania funkčnosti sa zameriame na mieru splnenia požiadaviek zadania, otestovanie krajných situácií a scenárov ako aj správanie sa samotných logických členov v najčastejších zapojeniach.

Uvedený program je vhodný najmä pre pedagogický proces výučby témy logických obvodov, kde musí každý študent počítačových systémov začať. Dostáva sa mu tým možnosť priamo si overiť správanie sa a vlastnosti logických členov v reálnych zapojeniach a nie sú pri tom potrebné konkrétne fyzické stavebnice. Cieľom tejto práce nie je plnohodnotne nahradiť priamy kontakt s uvedenými obvodmi, ale tento kontakt doplniť, zjednodušiť a lepšie sa pripraviť na reálne zapájanie členov, čím presne vieme čo a ako ideme merať, overovať a vieme aké výsledky máme dosiahnuť. Okrem samotného výučbového procesu je program vhodný aj na rýchle overovanie návrhov, zjednodušení, minimalizácií a samotných zapojení pri práci s uvedenou tematikou.

Samotný program je ale vytváraný na najvyššej programátorskej úrovni abstrakcie v objektovo orientovanom programovacom jazyku napriek tomu, že opisuje najnižšiu možnú programátorskú úroveň samotných logických členov. Zaujímavosťou tiež je, že osobný počítač na ktorom bude program spustený bude simulovať chod logických členov, ktoré budú práve tento program vykonávať. Tým je vidieť, že logické členy a zapojenia ktoré sme v dnešných počítačoch vytvorili okrem iného pomáhajú pri zdokonaľovaní a vývoji samých seba.

1.1 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.

1.2 Ciele projektu

Cieľom tohto tímového projektu je vytvoriť aplikáciu, ktorá bude v čo najvyššej možnej miere použiteľná a bude poskytovať viaceré funkcie. Chceme aby výstupom tejto práce nebol len program, ktorý bude spĺňať len presne zadané podmienky zadania, teda iba univerzálnym virtuálnym verifikačným panelom logických obvodov.

Mala by to byť pomôcka, pomocou ktorej si bude možné svoj vytvorený obvod skontrolovať prostredníctvom pravdivostnej tabuľky, ktorú bude možné v programe vygenerovať. Taktiež chceme, aby bolo možné zadať funkciu, na základe ktorej sa používateľovi vygeneruje logický obvod. Okrem toho bude možné vytvoriť obvod na základe zadania Karnaughovej mapy, ktorú bude tiež možné v programe aj vygenerovať z nakresleného obvodu.

Naším cieľom je vytvoriť program takých kvalít, aby sa mohol používať aj na cvičeniach niektorých predmetov, ako je napríklad logické obvody, a pod. Chceme aby sa používal nielen na vytvorenie nejakého obvodu, ale aby sa používal aj pre overovanie zadaní, napríklad pomocou pravdivostnej tabuľky alebo prostredníctvom Karnaughovej mapy.

1.3 Použité skratky

AND – logický súčin
XOR – výlučný logický súčet, exkluzívny súčet
NAND – negovaný logický súčin
OR – logický súčet

NOR – negovaný logický súčet
NOT – negácia
GUI – z ang. *Graphic User Interface,* t.j. grafické používateľské rozhranie

1.4 Zoznam obrázkov

4.3 Zoznam tabuliek

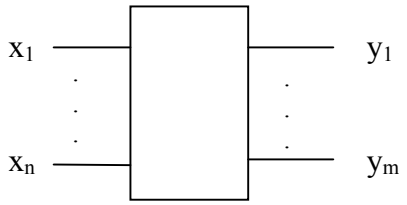
2. Analýza

Logický systém predstavuje číslicový systém, ktorého premenné nadobúdajú iba dve binárne hodnoty logická 1 a logická 0. Tieto hodnoty sa tiež nazývajú booleovské hodnoty a príslušné premenné takéhoto systému nazývame booleovské premenné.

Booleovská funkcia je každá funkcia f: $B^n \rightarrow B$, kde $n \in N^+$ a B = {0,1}. Booleovskú funkciu môžeme zapísať formou predpisu, pravdivostnou tabuľkou, alebo inými zápismi, ktoré momentálne nepotrebujeme poznať. Premenné takéhoto systému sa môžu vyskytovať v priamom tvare *x*, alebo negovanom tvare $\bar{x}$, pričom platí ak x=0 $\Leftrightarrow \bar{x}$=1 a $\bar{\bar{x}}$ = x. Úlohou mojej práce je modelovať práve takýto logický (booleovský) systém, ktorý bude realizovať booleovské funkcie logických členov AND a XOR.

2.1 Logické členy

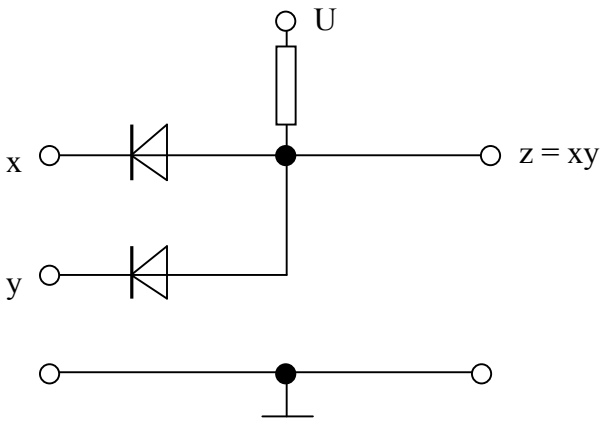
Booleovské funkcie možno modelovať pomocou rôznych technických prostriedkov, či už mechanických, hydraulických, pneumatických, alebo najčastejšie elektrických. Fyzikálne systémy, ktoré realizujú tieto základné booleovské funkcie nazývame logické členy alebo hradlá. Každý logický člen je charakterizovaný vzťahom medzi jeho vstupnými a výstupnými logickými premennými. Vstupné premenné sú nezávislé a výstupné premenné sú závislé premenné tej booleovskej funkcie, ktorú daný logický člen realizuje. Každej vstupnej premennej x je pri elektronických systémoch priradené napätie $U_x \in \{U_0, U_1\}$. U_1 nazývame jednotkovou úrovňou, U_0 nulovou úrovňou napätia (signálu). Najpoužívanejšími technológiami reprezentácie elektronických logických členov sú technológie TTL a CMOS. Logický člen s *n* vstupmi a *m* výstupmi je znázornený na obr. č. 1.



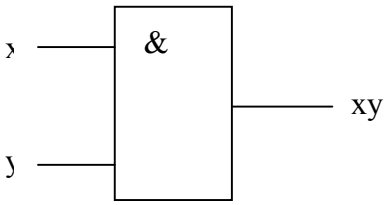
Obr. 1 Logický člen s n vstupmi a m výstupmi

2.1.1 Súčinový logický člen AND

Predstavuje logický člen s *n* vstupmi a 1 výstupom, potom hovoríme o *n*-vstupovom AND-e. Realizuje booleovskú funkciu f(x,y) = xy. Na obr. č. 2 je jeho diódová realizácia. Nech je $U > U_1$. Ak je na ktorúkoľvek diódu pripojené vstupné napätie U_0, dióda je otvorená, a ak zanedbáme jej vnútorný odpor v priamom smere, úroveň U_0 (zväčšená o úbytok na dióde, ktorý môžeme zanedbať) je aj na výstupe člena. Úroveň napätia U_1 sa objaví na výstupe člena práve vtedy, keď sú obidve diódy uzavreté, to nastane keď je na ne pripojené vstupné napätie U_1. Ak zanedbáme úbytky napätia na diódach, tak pre výstupné napätie platí $U_z = \min\{U_x,U_y\}$. Obvod teda skutočne realizuje logický súčin. Symbolická značka logického člena AND je na obr. 3.



Obr. 2 Vnútorná reprezentácia logického člena AND [3]



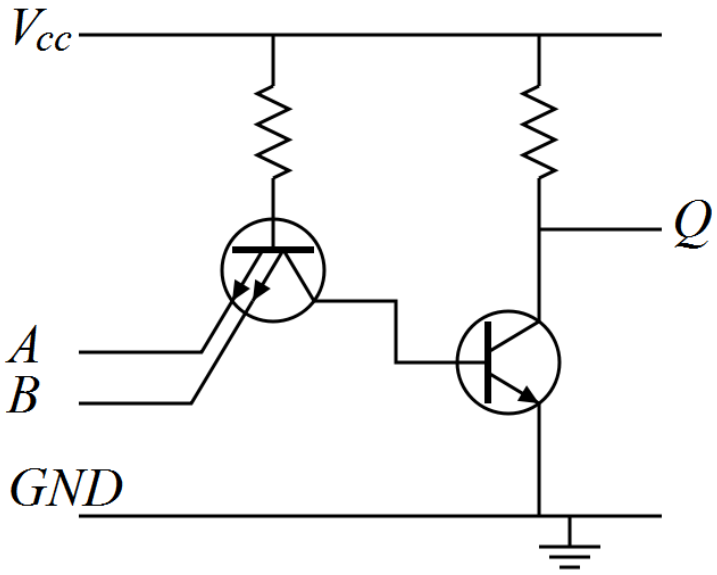
Obr. 3 Symbolická značka logického člena AND

x	y	f(x,y)
0	0	0
0	1	0
1	0	0
1	1	1

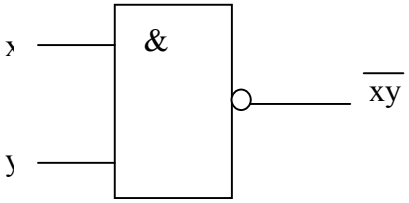
Tab. 1 Pravdivostná tabuľka logického člena AND

2.1.2 Súčinový logický člen NAND

Predstavuje logický člen s *n* vstupmi a 1 výstupom, potom hovoríme o *n*-vstupovom NAND-e. Realizuje booleovskú funkciu f(x,y) = $\overline{xy}$. Na obr. č. 4 je jeho realizácia technológiou TTL. Symbolická značka logického člena NAND je na obr. 5.



Obr. 4 Vnútorná reprezentácia logického člena NAND technológiou TTL



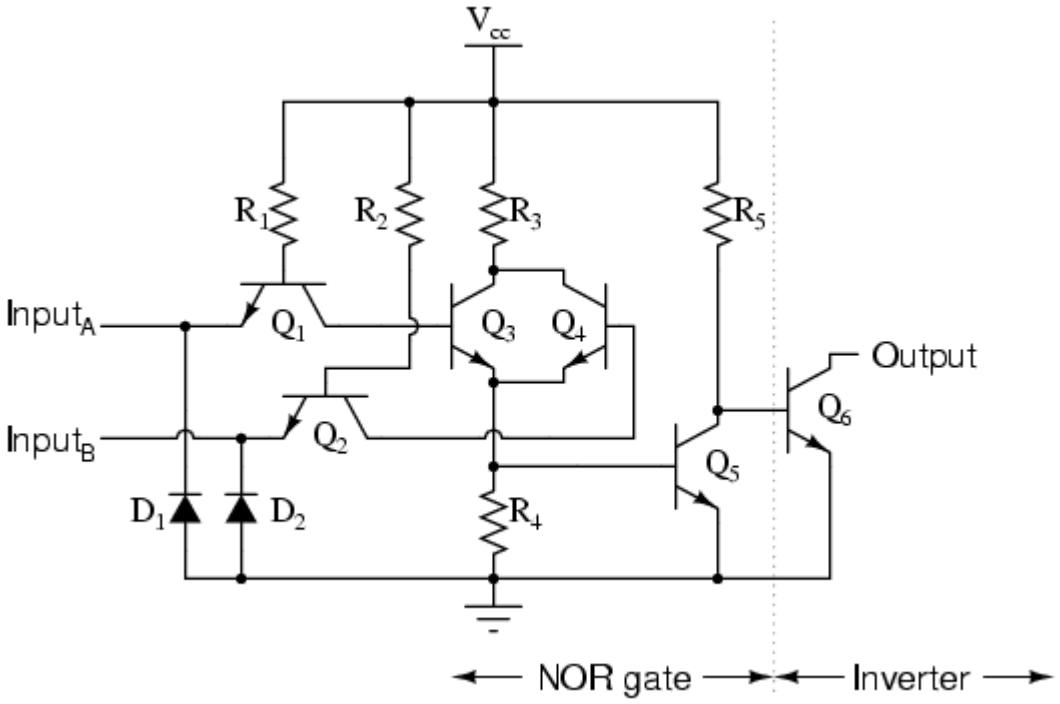
Obr. 5 Symbolická značka logického člena NAND

x	y	f(x,y)
0	0	1
0	1	1
1	0	1
1	1	0

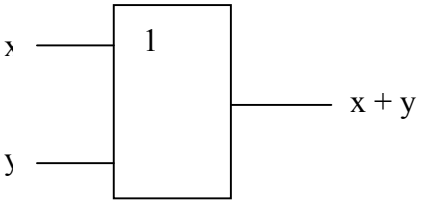
Tab. 2 Pravdivostná tabuľka logického člena NAND

2.1.3 Súčtový logický člen OR

Predstavuje logický člen s *n* vstupmi a 1 výstupom, potom hovoríme o *n*-vstupovom OR-e. Realizuje booleovskú funkciu f(x,y) = x + y. Na obr. č. 6 je jeho realizácia technológiou TTL. Je tvorený členom NOR, ktorého výstup je negovaný doplnením ďalšieho NPN tranzistora Q6. Symbolická značka logického člena OR je na obr. 7.



Obr. 6 Vnútorná reprezentácia logického člena OR technológiou TTL



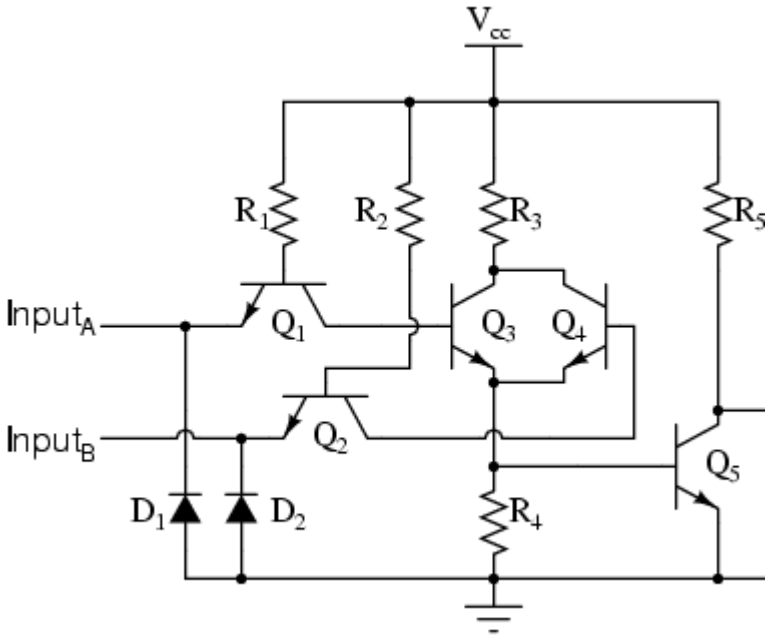
Obr. 7 Symbolická značka logického člena OR

x	y	f(x,y)
0	0	0
0	1	1
1	0	1
1	1	1

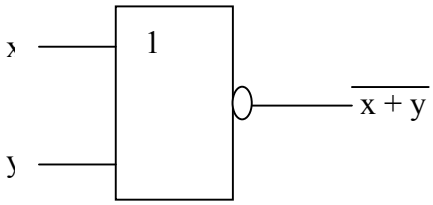
Tab. 3 Pravdivostná tabuľka logického člena OR

2.1.4 Súčtový logický člen NOR

Predstavuje logický člen s *n* vstupmi a 1 výstupom, potom hovoríme o *n*-vstupovom NOR-e. Realizuje booleovskú funkciu f(x,y) = $\overline{x + y}$. Na obr. č. 8 je jeho realizácia technológiou TTL. Schéma je obdobná ako u člena OR, len bez negácie výstupu. Symbolická značka logického člena NOR je na obr. 9.



Obr. 8 Vnútorná reprezentácia logického člena NOR technológiou TTL



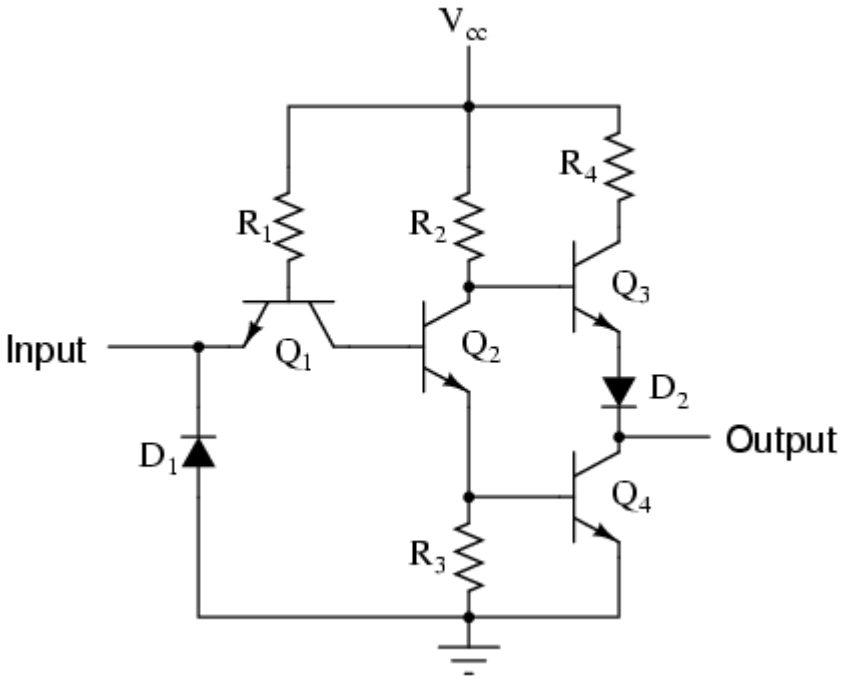
Obr. 9 Symbolická značka logického člena NOR

x	y	f(x,y)
0	0	1
0	1	0
1	0	0
1	1	0

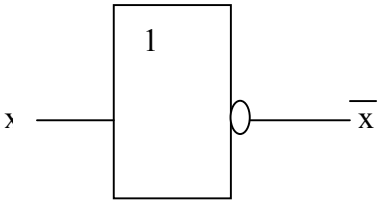
Tab. 4 Pravdivostná tabuľka logického člena NOR

2.1.5 Negovaný logický súčet NOT

Predstavuje logický člen s 1 vstupom a 1 výstupom. Realizuje booleovskú funkciu f(x) = $\overline{x}$. To je funkcia negácie, čiže invertuje vstupnú logickú hodnotu na opačnú výstupnú. Na obr. č. 10 je jeho realizácia technológiou TTL. Symbolická značka logického člena NOT je na obr. 11.



Obr. 10 Vnútorná reprezentácia logického člena NOT technológiou TTL



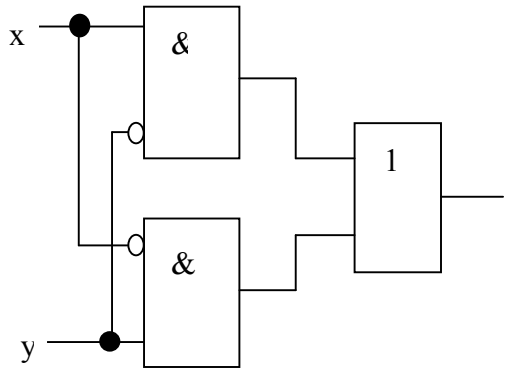
Obr. 11 Symbolická značka logického člena NOT

x	f(x)
0	1
1	0

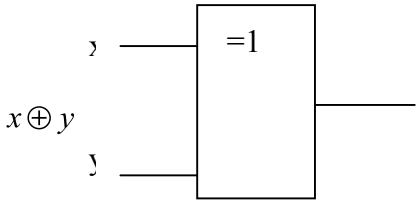
Tab. 5 Pravdivostná tabuľka logického člena NOT

2.1.6 Člen neekvivalencie XOR

Predstavuje logický člen s 2 vstupmi a 1 výstupom. Patrí k obvodom so strednou integráciou. Inak nazývaný EXCLUSIVE – OR, alebo aj suma modulo 2. Realizuje funkciu y = $x_1 \oplus x_2 = \overline{x_1}x_2 + x_1\overline{x_2}$. V jednom integrovanom obvode 7486 sú štyri logické členy tohto typu. Najväčšie oneskorenie jedného člena je na úrovni dvojnásobku oneskorenia logických členov s malou integráciou. Na rozdiel od logických členov NAND, NOR a AND-NOR, operácia XOR netvorí funkčne úplný systém. Bez použitia ďalších logických členov ju možno použiť iba v niektorých výnimočných prípadoch. Na obr. č. 12 je zobrazená vnútorná reprezentácia členu. Symbolická značka logického člena XOR je na obr. 13.



Obr. 12 Vnútorná reprezentácia logického člena neekvivalencie [3]



Obr. 13 Symbolická značka logického člena XOR

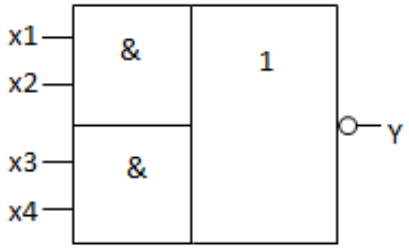
x	y	f(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 6 Pravdivostná tabuľka logického člena XOR

2.1.7 Logický člen AND-NOR

Logický člen AND-NOR je kombinácia jedného logického člena NOR a dvoch logických členov AND. Je realizovaný ako dvoj stupňový kombinačný obvod, kde sú výstupy dvoch členov AND pripojené na vstupy člena NOR. Na obrázku č. 14 môžeme vidieť symbolickú značku logického člena AND-NOR.

Realizuje booleovskú funkciu: $y = \overline{x_1.x_2 + x_3.x_4}$



Obr. 14 Symbolická značka logického člena AND-NOR

x_1	x_2	x_3	x_4	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Tab. 7 Pravdivostná tabuľka hodnôt pre štvorvstupový AND-NOR

2.2 Funkčne úplná množina

Množinu G nazývame funkčne úplnou práve vtedy, ak pre každú B-funkciu f existuje jej zodpovedajúci výraz, ktorý obsahuje iba operácie z množiny G. Ináč povedané z operácií obsiahnutých vo funkčne úplnej množine vieme zostaviť všetky existujúce booleovské funkcie. Funkčne úplnú množinu tvoria i členy AND a XOR s pridaním hodnoty logickej jednotky. Tieto výrazy s použitím členov AND, XOR a logická 1 tvoria základ Žegalkinovej algebry a nazývajú sa aj Ž-výrazy. Dôkaz úplnosti tejto množiny:

$$0 = x \oplus x$$
$$\overline{x} = x \oplus 1$$
$$x + y = [(x \oplus 1)(y \oplus 1)] \oplus 1$$
$$\overline{x + y} = (x \oplus 1)(y \oplus 1)$$
$$\overline{xy} = (xy) \oplus 1$$

Ďalšími funkčnými úplnými množinami sú:

- NAND
- NOR
- AND, OR a NOT
- AND-NOR a NAND
- OR a XOR

Obdobným spôsobom sa u nich dá dokázať, že je s týchto členov samotných možné poskladať všetky existujúce booleovské funkcie.

2.3 Logický zisk člena

Predstavuje počet vstupov logických členov, ktoré je možné pripojiť na výstup jedného logického člena bez straty napäťových úrovní logických hodnôt. Pri pripojení viacerých vstupov logických členov na daný výstup už nie sú garantované napäťové hodnoty a môžu sa dostať do zakázaného pásma, čo predstavuje hodnotu napätia medzi hodnotami 0 a 1. Vstup nasledujúceho logického člena tak nevie určiť, či má na vstupe hodnotu logickej 0 alebo 1. U bežných hradiel 7400 je logický zisk rovný 10, existujú i výkonové hradlá u ktorých je možné na výstup pripojiť 20 alebo 30 ďalších členov [1].

2.4 Princíp syntézy logických sietí

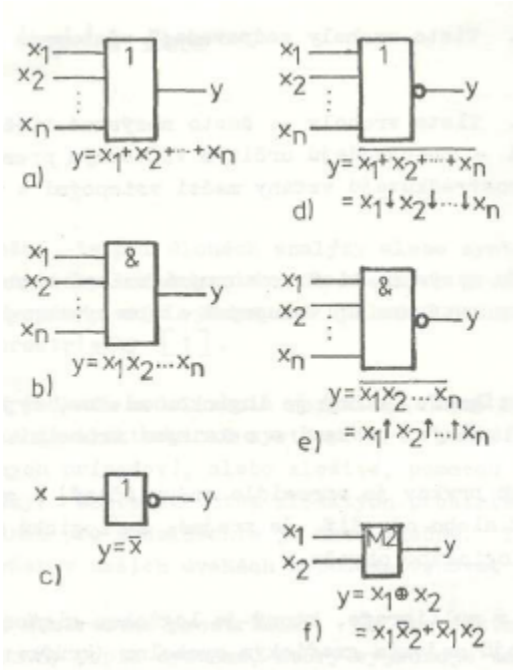
V tejto časti sú uvedené len základné myšlienky syntézy kombinačných logických sietí. Za syntézu kombinačnej logickej siete budeme považovať riešenia takejto úlohy:

Je zadaná skupina B – funkcií f_j (j = 1, 2, ..., m), kde m >= 1; ďalej je zadaný súbor typov logických členov, prípadne typ siete, a treba nájsť logickú sieť, ktorá zadanú skupinu B – funkcií realizuje.

Táto úloha nemá jednoznačné riešenie, t. j. pre daný súbor B – funkcií existuje mnoho rozličných sietí, ktoré zadanú skupinu funkcií realizujú. Z toho dôvodu sa hľadá optimálne riešenie podľa vopred zadaných kritérií. Kritériá optimálnosti môžu byť rozličné, najčastejšie je to napr. minimálny celkový počet logických členov v sieti.

Pri syntéze postupujeme zvyčajne tak, že sa najprv hľadá skupina výrazov, zodpovedajúca zadanej skupine B – funkcií, pričom sa hľadajú vhodné typy výrazov, zodpovedajúce východiskovému súboru typov logických členov a vyžadovaným kritériám optimálnej syntézy. Riešenie tejto úlohy je vo všeobecnosti vážny problém. Jeho algoritmické riešenie existuje len pri niektorých typoch sietí a triedach výrazov, napr. v triedach normálnych foriem NF g_1/g_2, a pri niektorých kritériách optimálnosti.

Po zostavení vhodnej skupiny výrazov pre zadané B – funkcie pristupujeme k tvorbe hľadanej logickej siete. Riešenie tejto úlohy nie je v princípe zložité, ak uvážime, aký je vzťah medzi výrazmi a sieťou a ďalej, ak v súbore logických členov, z ktorých môžeme logickú sieť realizovať, existujú všetky potrebné členy.
Na obrázku (Obr. 15) je znázornený vzťah medzi výrazmi a logickými členmi.



Obr. 15 Niektoré elementárne logické členy (a) OR, b) AND, c) NOT, d) NOR, e) NAND, f) XOR)

2.5 Existujúce riešenia

Táto kapitola obsahuje opis vybraných existujúcich riešení problematiky simulácie a verifikácie logických obvodov, ktoré sú dostupné na Internete. Existuje veľa aplikácií, ktoré slúžia na tento účel. Mnohé sú voľne šíriteľné, ale existujú aj komerčné riešenia. Nie všetky aplikácie je potrebné nainštalovať na počítač, pretože niektoré sú spustiteľné priamo cez webový prehliadač. Nie vždy to však môžeme považovať za výhodu, pretože takéto aplikácie sú zväčša funkčne obmedzené. Taktiež nie vždy máme k dispozícii pripojenie k Internetu a potrebujeme pracovať v režime offline.

Preto môžeme existujúce riešenia rozdeliť do štyroch skupín:

- Voľne dostupné riešenia bez potreby inštalácie
- Voľne dostupné riešenia s potrebou inštalácie
- Komerčné riešenia (s potrebou inštalácie)
- Bakalárske práce

2.5.1 Voľne dostupné riešenia bez potreby inštalácie

Ako sme už spomínali, tieto aplikácie sú zväčša spustiteľné cez webový prehliadač. Táto voľba riešenia umožňuje používať aplikáciu bez potreby inštalácie. Z toho môžeme konštatovať vysokú prenositeľnosť týchto aplikácií. Problémom však je, ak chceme používať takéto aplikácie z počítačov bez internetového pripojenia. V nasledujúcej časti dokumentu opíšeme bližšie niektoré aplikácie.

2.5.1.1 Logic Gate Simulator

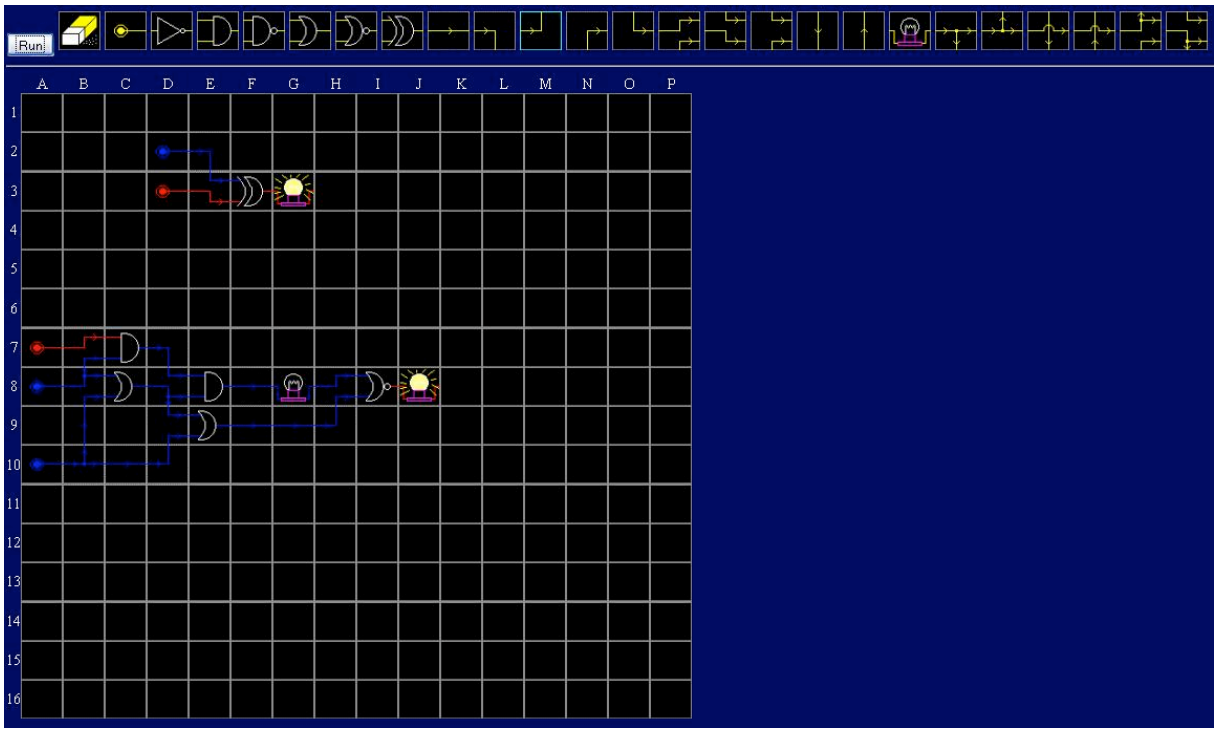
Táto aplikácia je naprogramovaná pomocou skriptovacieho jazyka JavaScript. Používateľské rozhranie tvorí pracovná plocha s rozmerom 16 x 16 štvorčekov. Do každého štvorčeka môže používateľ vložiť súčiastku z panela, ktorý sa nachádza v hornej časti používateľského rozhrania, nad štvorčekovou plochou.

Možné súčiastky:

- **Guma** – slúži na vymazanie štvorčeka, do ktorého bola už vložená súčiastka.
- **Prepínač** – slúži na prepínanie logickej hodnoty, ktorú posiela na vstup. Kliknutím na túto súčiastku sa prepne hodnota a zmení sa farba súčiastky. Červená farba znázorňuje logickú 1 a modrá znázorňuje logickú 0.
- **Logické hradlá** – k dispozícii sú logické členy NOT, AND, NAND, OR, NOR, XOR.

- **Vodiče** – aplikácia ponúka výber smeru vodičov. Vodič mení farbu podľa logickej hodnoty, ktorú prenáša.
- **Žiarovka** – slúži na zdôraznenie logickej hodnoty, ktorá prichádza na vstup tejto súčiastky. Táto súčiastka má neobvykle konektory na oboch stranách, takže môžeme pripojiť súčiastky aj za žiarovky.

Na obrázku č. 16 je zobrazené používateľské rozhranie programu Logic Gate Simulator.



Obr. 16 Používateľské rozhranie programu Logic Gate Simulator

Táto aplikácia je jednoduchá a intuitívna. Pri tvorbe logického obvodu používateľ myšou klikne na súčiastku, ktorú chce vložiť na plochu. Následne klikne myšou na tie štvorčeky plochy, do ktorých chce túto súčiastku vložiť. Ak chce používateľ simulovať vytvorený obvod, tak klikne na tlačidlo „Run“. Po každej zmene vstupnej hodnoty na prepínači, je potrebné znovu spustiť simuláciu tlačidlom „Run“.

Autor sa taktiež pokúsil o možnosť uložiť vytvorené zapojenie súčiastok. Po kliknutí na tlačidlo „Save“ sa v textovom poli, ktoré sa nachádza na stránke, má zobraziť matica z indexom, pre každý štvorček zobrazovacej plochy. Toto textové pole sa následne dá skopírovať do súboru a uložiť v počítači. Následne môže používateľ zo súboru skopírovať maticu do textového poľa a kliknutím na tlačidlo „Load“ zobraziť znovu vytvorené zapojenie súčiastok. Táto funkcia ukladania však pri testovaní aplikácie nefungovala.

Táto aplikácia obsahuje množstvo nedostatkov. Jedným z nich je obmedzená veľkosť zobrazovacej plochy, do ktorej sa zmestia naozaj len jednoduché obvody. Ďalším nedostatkom je obmedzené zobrazenie

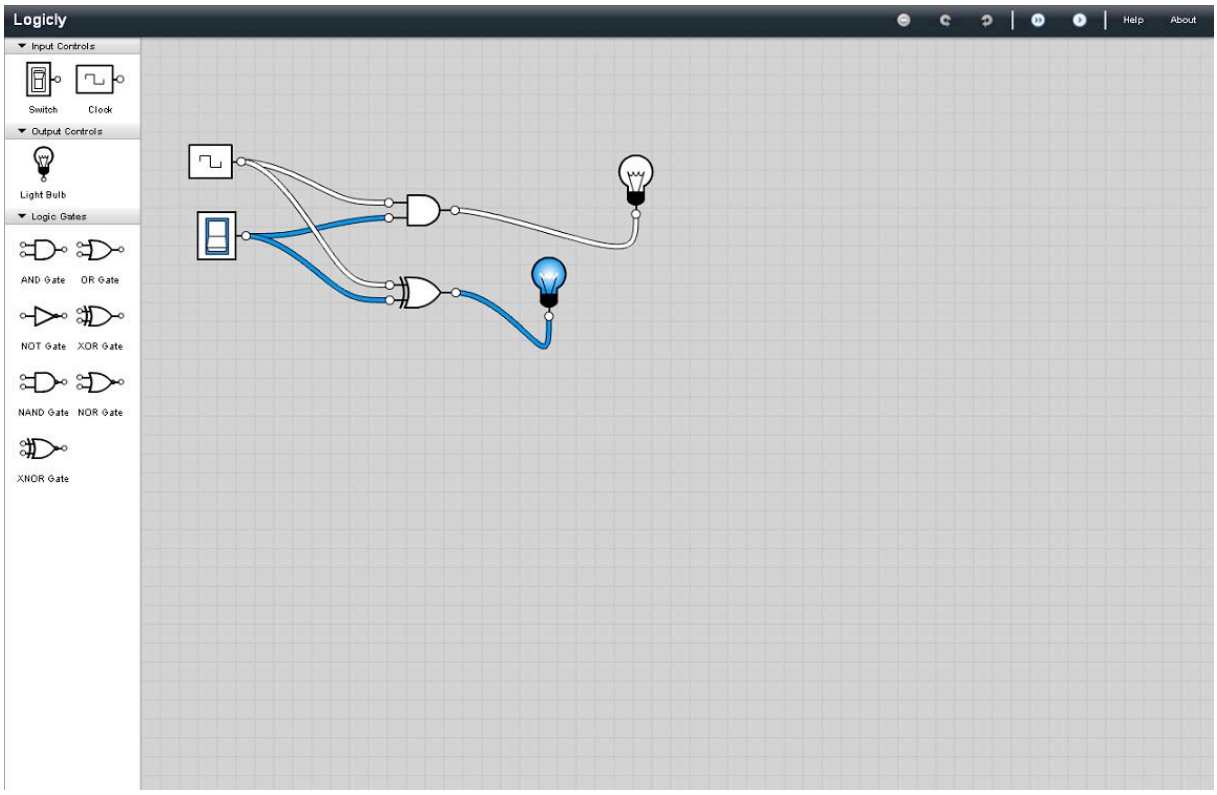
vodiča. Aplikácia ponúka len 16 možností vedení vodiča. K nedostatkom patrí aj to, že aplikácia umožňuje navrhnúť obvod len v smere zľava doprava. Tiež je nedostatkom, že ponúkané logické členy sú len dvojvstupové.

Ako z analýzy tohto riešenia vyplýva, táto aplikácia umožňuje simuláciu a teda aj verifikáciu len veľmi jednoduchých logických obvodov. Táto simulácia je prehľadná a použiteľná, ale návrh zapojenia logického obvodu je veľmi obmedzený.

2.5.1.2 Logicly

Ďalšou webovou technológiou, ktorá sa priamo ponúka na prácu s modelovaním obvodu je *Adobe Flash* a najmä jeho odnož *Adobe Flex,* ktorá je kolekciou technológií určených pre vývoj tzv. *„rich internet aplikácií“*, teda aplikácii, ktoré majú funkcie a vlastnosti bežných aplikácií pre osobný počítač.

Používateľské rozhranie tejto aplikácie tvorí takisto pracovná plocha, na ktorej vytvárame daný obvod. V ľavej časti používateľského rozhrania sa nachádza panel so súčiastkami. V hornej časti sa nachádza ešte panel s ovládacími tlačidlami pre vymazanie označenej súčiastky, krokovanie simulácie, spustenie simulácie, pomoc a informácie o programe. Na obrázku č. 17 je zobrazené používateľské rozhranie programu Logicly.



Obr. 17 Používateľské rozhranie programu Logicly

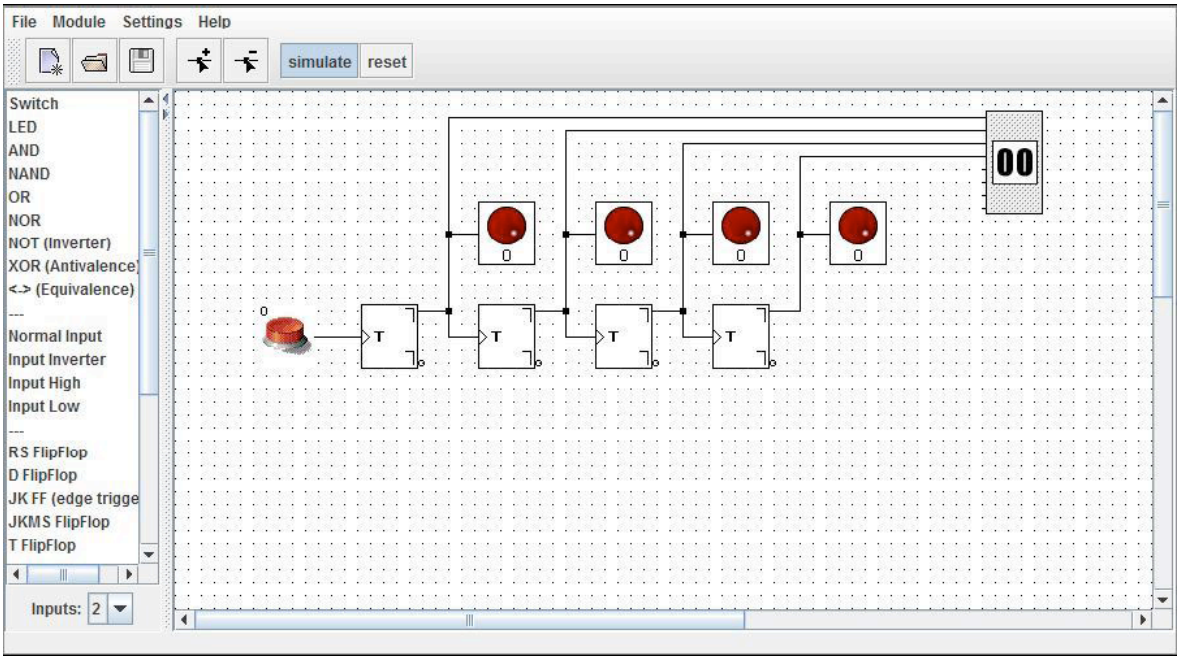
Toto používateľské rozhranie je veľmi príjemné, intuitívne a jednoduché. S aplikáciou sa naozaj pohodlne a rýchlo pracuje. Zaujímavosťou je, že táto aplikácia obsahuje aj súčiastku hodiny, ktorá môže slúžiť na otestovanie funkcionality preklápacích obvodov.

Jedným z mála nedostatkov tohto programu je nemožnosť ukladať si navrhnutý obvod, a teda používateľ musí daný obvod zapojiť na jeden raz. Táto skutočnosť znemožňuje, alebo minimálne znepohodlňuje využívanie tejto aplikácie na zložitejšie obvody. Tiež ako nedostatok môžeme brať len dvojvstupové logické členy.

Po analýze tohto riešenia môžeme zhodnotiť, že táto aplikácia je použiteľná na rýchly a jednoduchý návrh alebo verifikáciu (príp. simuláciu) nie veľmi zložitých logických obvodov. Aplikácia obsahuje aj návod na použitie, čo ale pri jej intuitívnosti ani nie je potrebné.

2.5.1.3 LogicSim

Nasledujúcim webovým riešením je *LogicSim Applet*, ktorý je zástupcom Javy. *LogicSim* nie je až tak celkom pravý applet, nakoľko je ho možné spustiť aj ako samostatnú *Java* aplikáciu. Vtedy poskytuje niekoľko funkcií navyše (napr. uloženie a načítanie zapojenia). Aplikácia je naprogramovaná pomocou grafických Java knižníc swing. Už na prvý pohľad ponúka veľa funkcií a javí sa ako doteraz najlepšia voľba pri webových riešeniach. Naľavo sa nachádzajú prvky, ktoré je možno do obvodu pridávať, v hornej časti je tento krát menu, na ktoré sme zvyknutý z bežných aplikácii. Pracovná plocha tvorí najväčšiu časť používateľského rozhrania. Veľa ovládacích prvkov predpovedá veľa možností programu. Logické brány sa dajú zobraziť v dvoch verziách: IEC a US. Na obrázku č. 18 je zobrazené používateľské rozhranie programu LogicSim.



Obr. 18 Používateľské rozhranie programu LogicSim

Používanie programu je v celku jednoduché aj intuitívne, avšak je tu pár nedostatkov. Napríklad kreslenie vodiča je veľmi nedomyslené a neintuitívne. Keď používateľ začne kresliť vodič, tak sa mu nedá nič iné robiť, až kým tento vodič nepripojí do koncového konektoru. Kreslenie prebieha pomocou záchytných (zlomových) bodov. Pomocou klávesy Esc sa dá kreslenie prerušiť, pri každom stisku sa odoberie posledný záchytný bod. Táto funkcia je potrebná, avšak neintuitívna.

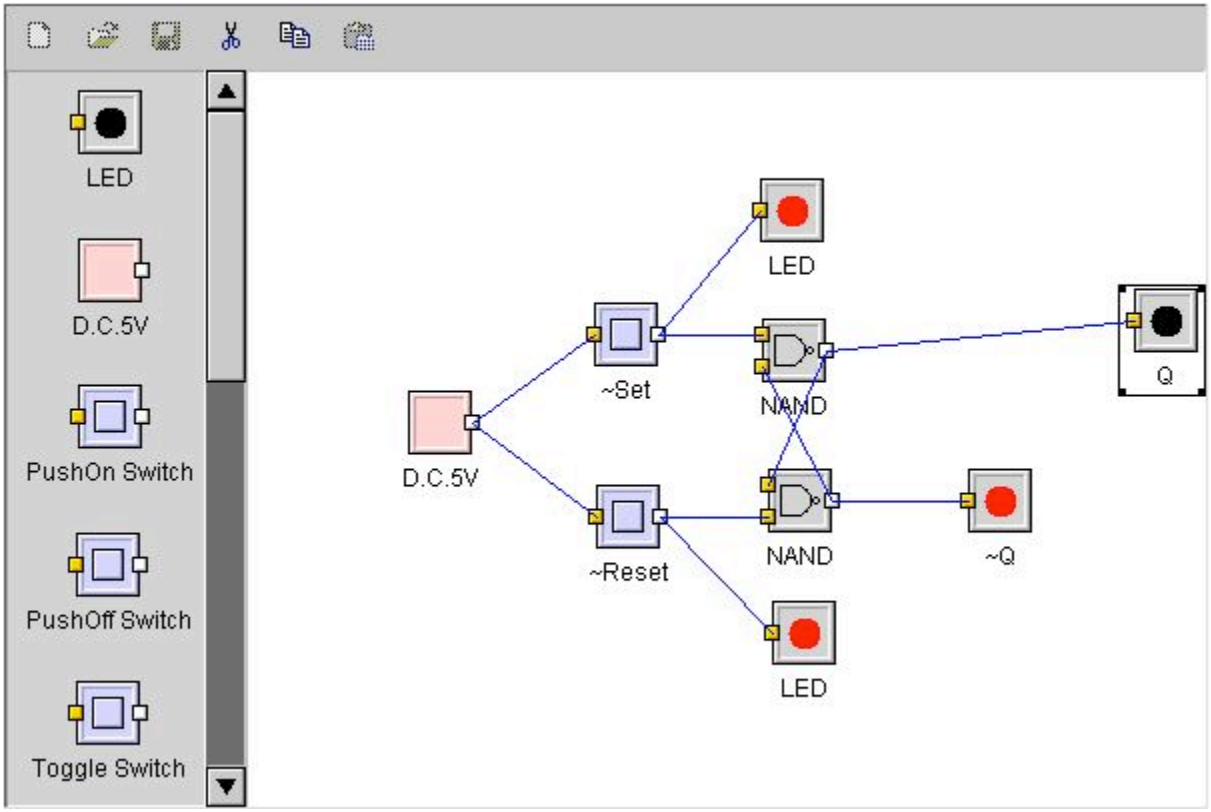
Aplikácia má na rozdiel od predošlých dvoch omnoho viacej možností. Okrem toho, že poskytuje napríklad aj preklápacie obvody, logické členy môžu mať od 2 do 5 vstupov. Takisto program okrem toho poskytuje komponent hodiny, 7 segmentový displej. Takisto užitočná je možnosť písať popisy k jednotlivým komponentom, respektíve hocikde na plochu.

Keď je aplikácia spúšťaná nie ako applet, používateľ má možnosť vytvárať moduly, čo sú v podstate vlastné súčiastky. Z menu sa jednoducho vyberie možnosť vytvoriť nový modul, modul si užívateľ vytvorí, následne ho uloží do priečinka medzi ostatné moduly. Okamžite sa táto súčiastka objaví v ľavom menu a je pripravená na použitie.

Ako z analýzy tejto aplikácie vidíme, je vhodná na schématické znázornenie, simuláciu (resp. verifikáciu) aj zložitejších logických obvodov. Zaujímavosťou práve tohto riešenia je uvedená možnosť vytvárať si vlastné súčiastky a následne ich používať. To nám umožňuje abstrahovať sa na vyššiu úroveň. Pri práci s touto aplikáciou je vhodné sa oboznámiť s funkcionalitou vytvárania vodičov, lebo používateľovi môžu veľmi znepríjemniť prácu.

2.5.1.4 Simcir

Toto riešenie pracuje tiež na princípe Java Applet. Taktiež umožňuje stiahnuť do počítača klasickú (newebovú) aplikáciu, ktorá umožňuje aj ukladať a otvárať schémy obvodov. Používateľské prostredie aplikácie je veľmi jednoduché. Vo vrchnej časti sa nachádza riadiaci panel, na ktorom sú ovládacie prvky pre vytvorenie nového obvodu, uloženie a načítanie obvodu, vystrihnutie, kopírovanie a prilepenie časti obvodu. V ľavej časti sa nachádza panel s použiteľnými súčiastkami. Ako i v ostatných aplikáciách, tak aj v tejto najväčšiu časť rozhrania tvorí pracovná plocha na zapojenie súčiastok do obvodu. Na obrázku č. 19 je zobrazené používateľské rozhranie programu Simcir.



Obr. 19 Používateľské prostredie programu Simcir

Je to skutočne jednoduchý a intuitívny program, ktorý umožňuje podobné funkcie ako prvé dve aplikácie. Nevyznačuje sa žiadnou zvláštnou vlastnosťou, až na existenciu netradičných prepínačov. Autor v tomto programe extrahoval zdroj napätia od prepínača. Aplikácia obsahuje tri druhy prepínačov:

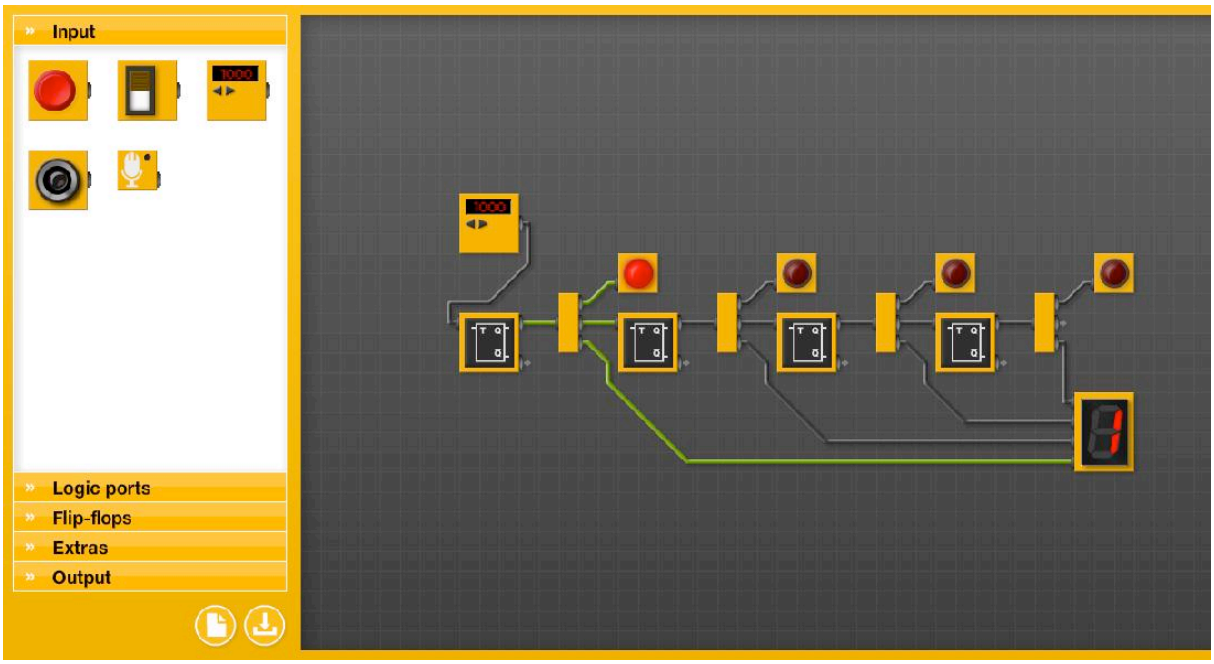
- PushOn Switch – konektory sú rozpojené, kým je prepínač stlačený, tak sú spojené.
- PushOff Switch – konektory sú spojené, kým je prepínač stlačený, tak sú rozpojené.
- Toggle Switch – klasický prepínač, ktorý drží hodnotu (logickú), po stlačení hodnotu prepne.

Výhodou je možnosť meniť názov súčiastok. Nevýhodou sú len dvojvstupové logické členy. K nedostatkom aplikácie patrí obmedzená veľkosť pracovnej plochy aplikácie a tým znemožnenie tvorby zložitejších obvodov.

Po analýze tohto riešenia môžeme zhodnotiť, že táto aplikácia je tiež vhodná na verifikáciu jednoduchých logických obvodov. Má obmedzené funkcie.

2.5.1.5 The Logic Lab

Ďalšie riešenie pracujúce na platforme flash. V ľavej časti používateľského rozhrania sa nachádza panel so súčiastkami usporiadanými v kategóriách. Pravú, väčšiu časť rozhrania tvorí opäť pracovná plocha. Na obrázku č. 20 je zobrazené používateľské rozhranie programu The Logic Lab.



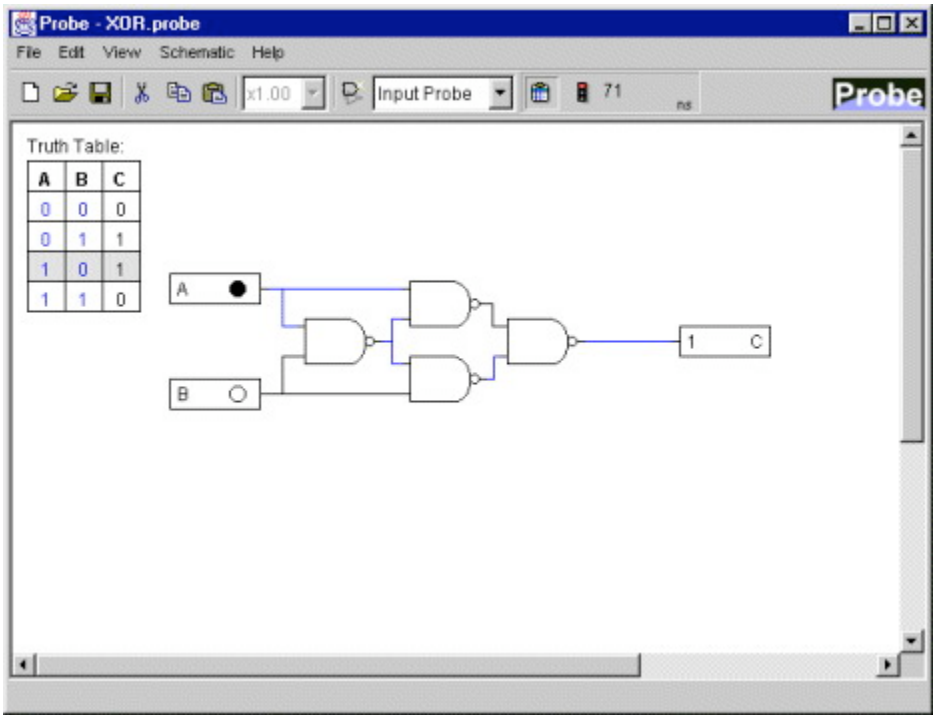
Obr. 20 Používateľské rozhranie programu The Logic Lab

Táto aplikácia obsahuje množstvo netradičných súčiastok, ako nastaviteľné hodiny, displeje, reproduktory, a iné. Taktiež obsahuje netradičné vetvenie vodičov, ako môžeme vidieť aj na obrázku. Z programom sa jednoducho, rýchlo a intuitívne pracuje. K nedostatkom je možné zaradiť nedostatočné informácie o funkcii súčiastok, minimálne názov chýba. Logické hradlá v tejto aplikácii sú tiež len dvojvstupové. Taktiež aj v tejto aplikácii je veľkosť pracovnej plochy obmedzená. K nedostatkom patrí aj neintuitívne odstránenie súčiastok a to takým spôsobom, že ich potiahneme mimo pracovnú plochu. Program taktiež obsahuje tlačidlo pre nový návrh logického obvodu, čím zmažeme predchádzajúce zobrazenie, a tlačidlo pre uloženie vytvoreného logického obvodu. Toto uloženie je neštandardne vymyslené tak, že sa logický obvod uloží na webový server autora a používateľ dostane k dispozícii webovú adresu, na ktorej tento obvod nájde. Je to neštandardné riešenie, ale funguje.

Po analýze programu môžeme povedať, že sa dá využiť na simulovanie malých logických obvodov, aj s neštandardnými súčiastkami (mikrofón, reproduktor, ...). Vzhľadom na obmedzenú veľkosť pracovnej plochy sú simulácie väčších logických obvodov nemožné.

2.5.1.6 Probe

Táto aplikácia je vytvorená znovu pomocou Java Applet. Používateľské rozhranie obsahuje ako v klasickej aplikácii menu, panel s nástrojmi a pracovnú plochu. Aplikácia neobsahuje žiadne zložitejšie súčiastky. Na obrázku č. 21 je zobrazené používateľské rozhranie programu Probe.



Obr. 21 Používateľské rozhranie programu Probe

K zaujímavostiam tejto aplikácie patrí možnosť zobrazenia pravdivostnej tabuľky. Práca s programom je jednoduchá, avšak mohla by byť aj viac intuitívnejšia. Používateľské rozhranie nie je veľmi prehľadné. K nedostatkom patrí neprehľadnosť zobrazenia, súčiastky a vodiče sa môžu prekrývať. Veľkosť pracovnej plochy je aj v tomto prípade obmedzená. Logické hradlá sú len dvojvstupové. Ako výhodu by bolo možné uviesť možnosť uloženia a načítania obvodu.

Táto aplikácia je tiež použiteľná len pre simuláciu tých najjednoduchších obvodov. Prínosom tejto aplikácie je možnosť zobrazenia pravdivostnej tabuľky, ktorú si používateľ môže porovnať s jeho vlastnou a tak rýchlejšie objaviť chybu návrhu. Preto môžeme tento nástroj hodnotiť aj ako dobrý verifikátor jednoduchých logických obvodov.

2.5.2 Voľne dostupné riešenia s potrebou inštalácie

Tieto riešenia sa vyznačujú tým, že ich je potrebné stiahnuť do počítača a nainštalovať (príp. rozbaliť). Oproti predchádzajúcim riešeniam majú horšiu prenositeľnosť, ale ako klasické aplikácie môžu obsahovať viac funkcií a možností. V tejto časti sú opísané niektoré z takýchto aplikácií.

2.5.2.1 LOG

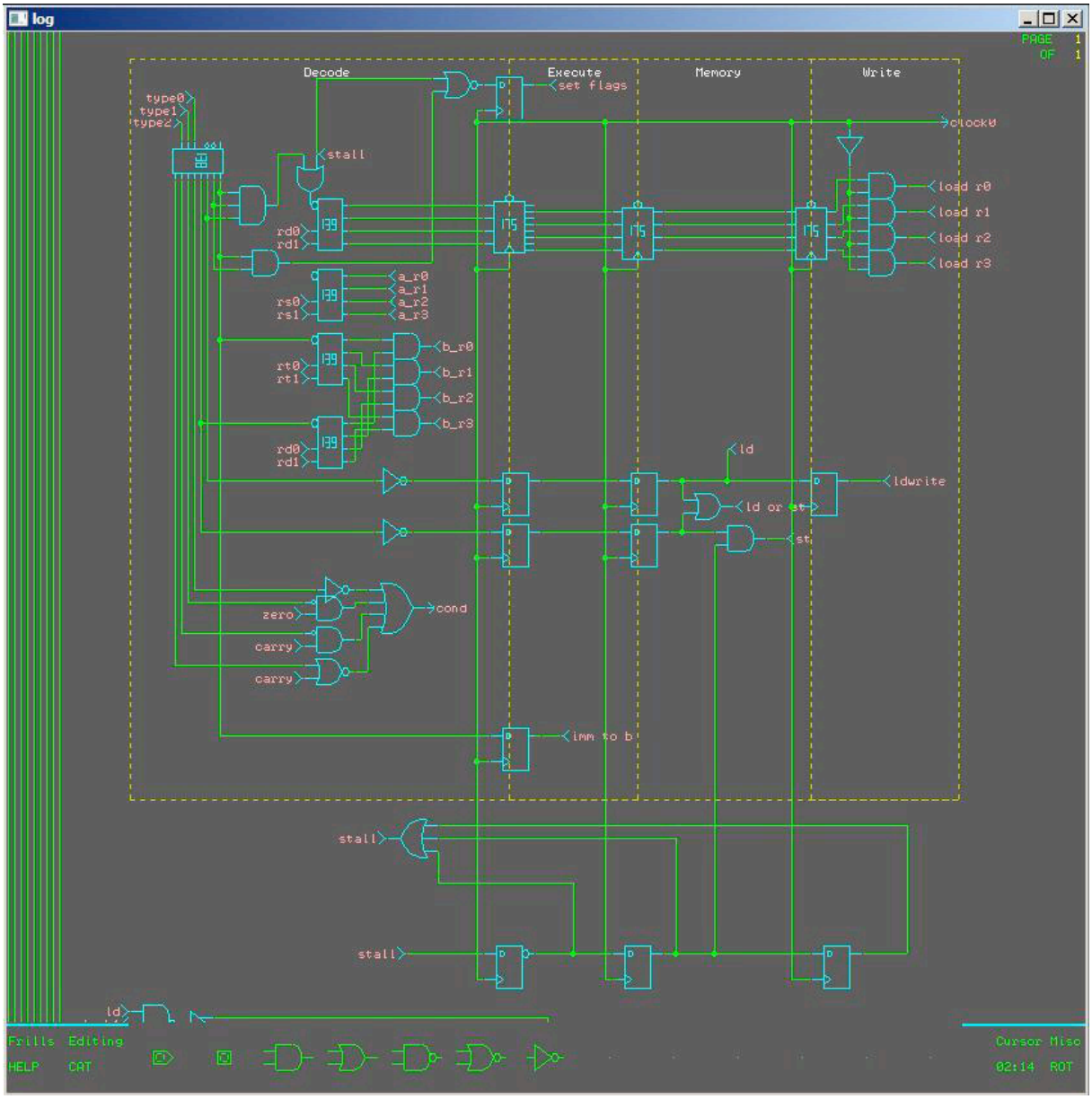
Tento programový prostriedok je dosť starý, preto aj vzhľad používateľského rozhrania tomu zodpovedá. V dolnej časti obrazovky sa nachádzajú ovládacie prvky programu, takisto ako aj jednotlivé logické hradlá, ktoré je možné do programu vkladať. V strede sa nachádza pracovná plocha, na ktorú je možné obvod kresliť. Jeho veľkosť je v podstate neobmedzená, nakoľko sa dá touto plochou hýbať pomocou šípok na klávesnici. Kreslenie vodičov je spravené asi najlepším spôsobom, pričom pomyselná mriežka, na ktorú sa obvod kreslí, je veľmi jemná, čo umožňuje kresliť vodiče tesne vedľa seba a šetriť tak priestorom. Jednotlivými prvkami je možné rotovať, čo umožňuje veľkú flexibilitu pri kreslení obvodov, čím sa z logu stáva naozaj použiteľný nástroj pre kreslenie schém logických obvodov a ich simuláciu.

Log poskytuje veľké množstvo funkcií a nástrojov pre zdokonalenie a uľahčenie návrhu logických obvodov. Prvou z funkcii, skrývajúcich sa pod menu „cursor“, je funkcia logickej sondy, kedy sa nám kurzor zmení na logickú sondu a ukazuje hodnotu práve skúmaného vodiča. Ďalej môžeme zdôrazniť takzvaný „glow mode“, ktorý tak ako pri predošlých aplikáciách ukazuje logickú hodnotu v jednotlivých segmentoch obvodu. Červená pre hodnotu 1, čierna pre hodnotu 0. Ďalšou možnosťou je zapnutie vizualizácie kríža, ktorého stred sa nachádza priamo na kurzore, ktorý zvyšuje presnosť kreslenia a zapne iný kresliaci mód vodiča. Zvyšné tlačidlá tohto menu umožňujú navigáciu cez kresliacu plochu.

Samozrejmosťou je načítanie už rozpracovanej práce zo súboru. Ďalej log poskytuje možnosť sledovania výstupných signálov vo forme „waveform“ pomocou funkcie „scope“. Ďalšia funkcia v menu „misc“ je funkcia „plot“, ktorá umožňuje obvod presne zakresliť a exportovať do formátu „postscript“.

Log umožňuje takisto popisovať, ohraničovať, značiť jednotlivé komponenty, čo určite prispieva k väčšej prehľadnosti celého návrhu. Poskytuje rôzne zobrazenia obvodu od už spomínaného „glow mode“ až po zobrazenie bez popisov, zobrazenie čisto iba popisov, alebo ľubovoľná kombinácia týchto zobrazení. Log takisto podporuje vytváranie vlastných modulov a ich následné použitie pri návrhu.

Avšak najväčšia prednosť tohto programu je veľmi obsiahly katalóg prvkov. Log ponúka naozaj široké spektrum rôznych komponentov, pričom nie sú výnimkou ani logické brány obsahujúce viac ako 2 vstupy, hodinový signál, 7segmentový displej, obvody rady 7400 a mnoho ďalších. Na obrázku č. 22 je zobrazené používateľské rozhranie programu Log.



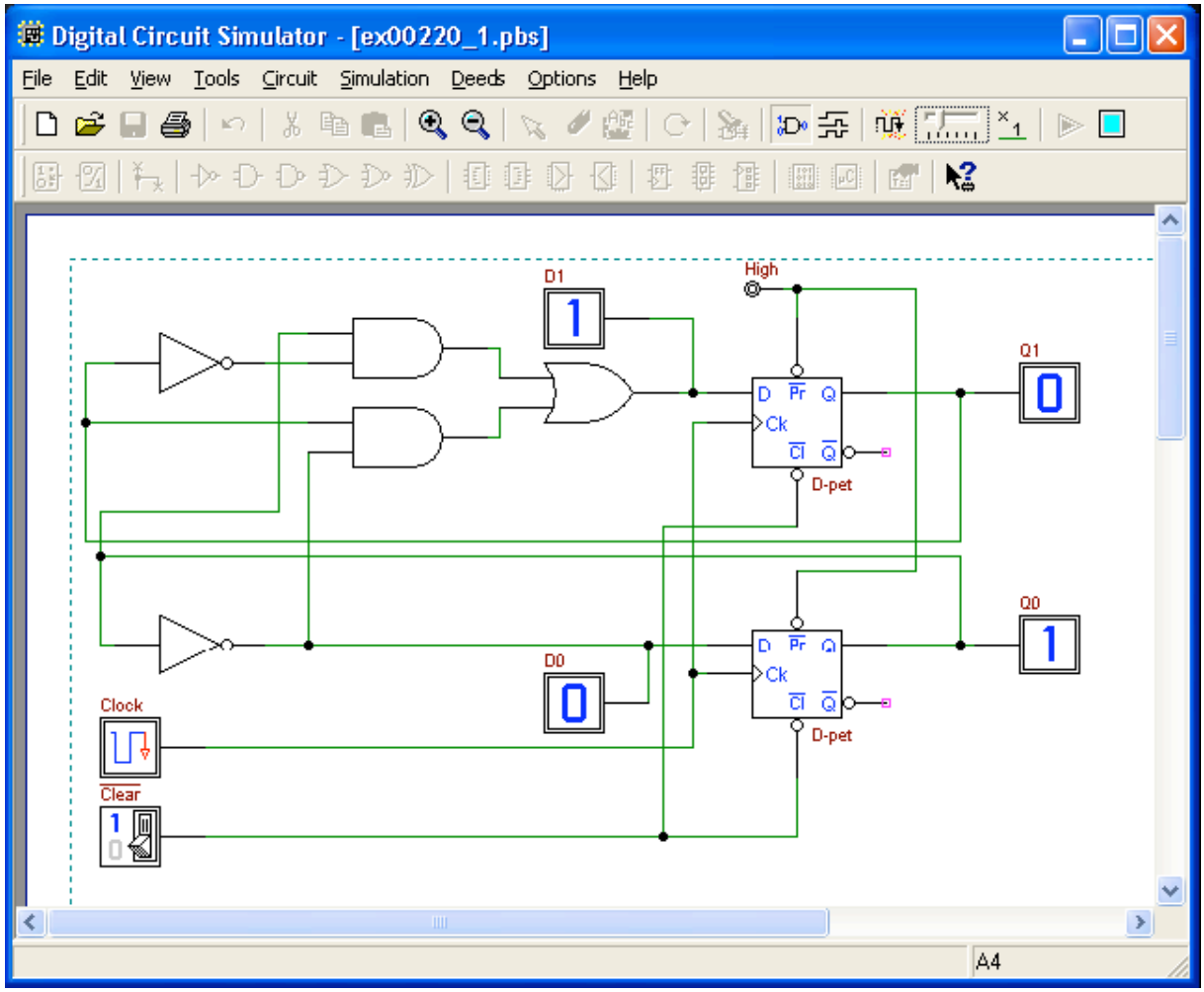
Obr. 22 Používateľské rozhranie programu Log

Používateľské rozhranie a práca s programom je veľmi neintuitívna a nie je jednoduchá, preto používateľ by sa mal najskôr naučiť pracovať s programom. To mu umožní používateľská príručka obsiahnutá v dokumentácii programu. Jeho hlavným nedostatkom je neustále prekresľovanie aplikácie, tým aj veľká náročnosť na procesor, ako aj neintuitívne ovládanie aplikácie pri kreslení schém.

Po analýze tohto riešenia môžeme zhodnotiť, že tento programový prostriedok je veľmi silným nástrojom na návrh a simuláciu aj zložitých logických obvodov. Prvému použitiu programu avšak musí predchádzať štúdium práce s programom. Keď sa používateľ naučí pracovať s programom, tak mu tento program, aj napriek svojmu zastaranému vzhľadu, môže veľmi pomôcť pri práci.

2.5.2.2 Deeds

Deeds (Digital Electronics Education and Design Suite) – simulátor a výučbový prostriedok logických obvodov vyvinutý University of Genoa, Italy. Okrem samotných simulácií zapojení ponúka aj výučbové materiály k téme logických obvodov. Ponúka tiež simulácie vstupno – výstupných portov, 8-bitových CPU, RAM a ROM pamäti. Na obrázku č. 23 je zobrazené používateľské rozhranie programu Deeds.



Obr. 23 Používateľské rozhranie programu Deeds

Používateľské prostredie tohto programu má na prvý pohľad veľké množstvo možností. Program je menej intuitívny, hlavne tvorba vodičov. Predtým, ako používateľ chce kresliť vodič, je potrebný jeho výber na súčiastkovom panely používateľského rozhrania. Veľkosť pracovnej plochy je ohraničená veľkosťou papiera. Používateľ má na výber veľkosti A3, A4 a A5. Toto obmedzenie znemožňuje tvorbu väčších logických obvodov, ale zároveň je prospešné, ak používateľ používa tento program pri tvorbe dokumentácie k návrhu logických obvodov.

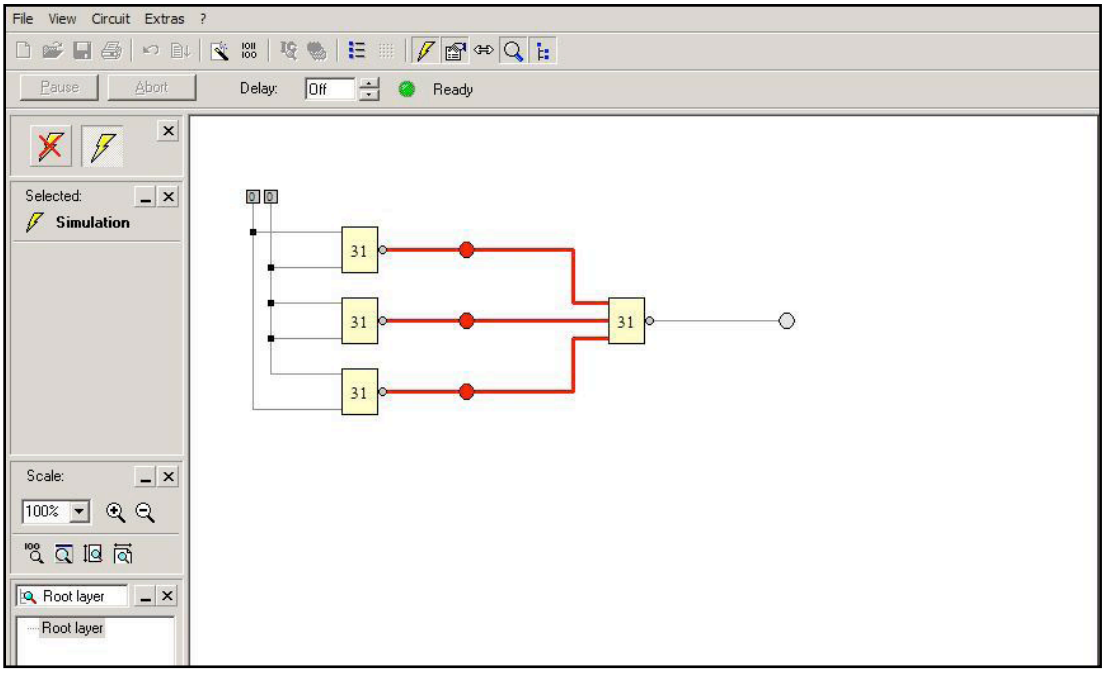
Tento program je veľmi silným prostriedkom podporujúcim návrh a simuláciu logických obvodov, využiteľná hlavne pri tvorbe dokumentácie k návrhu. Vzhľadom k množstvu podporovaných súčiastok možno konštatovať, že umožňuje aj tvorbu zložitejších logických obvodov, ale s obmedzenou veľkosťou.

2.5.3 Komerčné riešenia

Komerčné riešenia sú programy, ktoré je potrebné nainštalovať na počítač. Tieto programy sú spoplatnené, a preto testované boli len ich demoverzie. V nasledujúcej časti dokumentu budú analyzované niektoré z takýchto riešení.

2.5.3.1 LOGiX

LOGiX je riešenie na profesionálnejšej úrovni pre vývoj a testovanie diskrétnych logických obvodov. Môže byť použitý na vytvorenie elektronických zariadení všetkého druhu, od jednoduchých kalkulačiek a semaforov, až po komplexné integrované obvody. Na obrázku č. 24 je zobrazené používateľské rozhranie programu LOGiX.



Obr. 24 Používateľské rozhranie programu LOGiX

LOGiX umožňuje virtuálne kresliť logické obvody na obrazovku za použitia jednoduchého a ľahko naučiteľného rozhrania. Za použitia simulačného módu je možné dokonca vizualizovať oneskorenie toku elektrickej energie. LOGiX je komerčná aplikácia, ktorej plná verzia stojí 49.90 $.

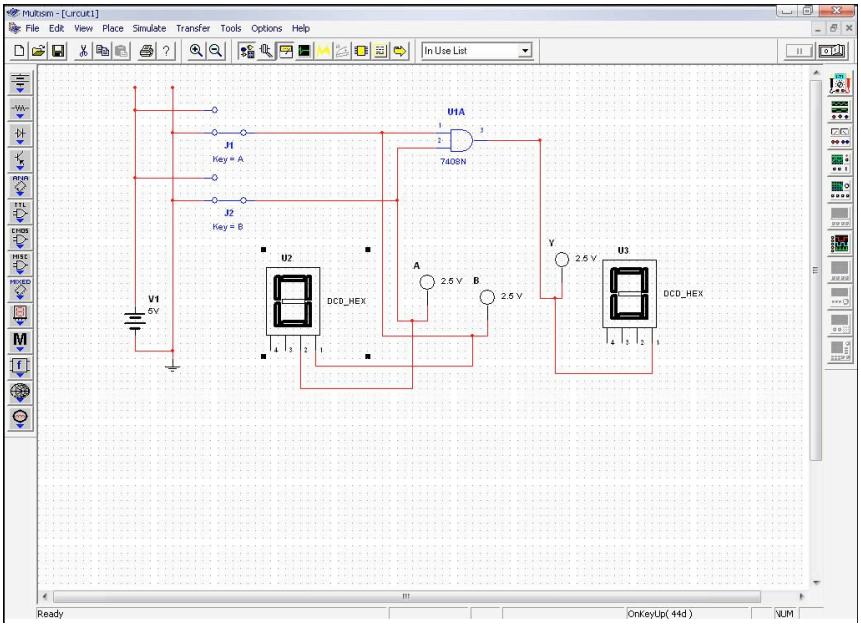
V ľavej časti používateľského rozhrania sa nachádzajú panely s nástrojmi: ovládací panel, panel s možnosťami, panel na ovládanie pracovného hárka, panel na ovládanie veľkosti komponentov, panel na ovládanie vrstiev. V hornej časti je menu, z ktorého je možné vyvolať takmer všetky funkcie aplikácie. Kreslenie obvodov je na najvyššej úrovni. Zaujímavá je možnosť modifikácie komponentov pomocou značiek negácie, vďaka ktorým je sa dá jednoducho vyrobiť z logického člena OR logický člen NOR, poprípade znegovať vstup niektorého komponentu a podobne. Kreslenie spojov medzi jednotlivými komponentmi je veľmi intuitívne a jednoduché. Veľmi vhodná možnosť programu je krok späť, ktorým sa dajú vrátiť posledné úpravy, čo zpohodlňuje prácu používateľa. K zaujímavým vlastnostiam programu patrí aj možnosť vygenerovania logického obvodu z booleovskej funkcie, alebo pravdivostnej tabuľky. Takisto program vygeneruje pravdivostnú tabuľku z booleovskej funkcie. Tento program tiež podporuje tvorbu vlastných modulov a ich použitie v ďalšej práci.

Po zanalyzovaní funkcionality tohto programu možno konštatovať, že tento program je naozaj silný prostriedok pre návrh a simuláciu (príp. verifikáciu) logických obvodov. Pre používateľa je práca s týmto programom veľmi jednoduchá a pohodlná, no kvôli jeho komplexnosti je vhodné prvotné preštudovanie návodu na použitie.

2.5.3.2 Electronics Workbench Multisim

Ide o rozsiahly softvérový produkt od firmy National Instruments, ktorý umožňuje simulovať činnosť analógových i číslicových elektronických obvodov a komponentov.

Na obrázku č. 25 je zobrazené používateľské rozhranie programu Electronics Workbench Multisim.



Obr. 25 Používateľské rozhranie programu Multisim

Program umožňuje zapojiť a odskúšať číslicové logické obvody, ktoré môžu byť zostrojené zo všetkých dostupných súčiastok ktoré sú momentálne na trhu s elektronikou. Okrem najpoužívanejších logických členov založených na technológii TTL ponuka aj CMOS technológiu. Funkčne vychádza z predchodcu WorkBench (používaná skratka EWB), ktorý bol veľmi rozšíreným návrhovým prostriedkom logických obvodov. Nevýhodou tohto programu je najmä to, že je to komerčné riešenie, a teda pre jeho neobmedzenú funkčnosť je potrebné zaplatiť poplatok 25 $.

Program je vhodný na komplikovanejšie a obsiahlejšie návrhy a analýzy, nakoľko okrem samotných logických hodnôt simuluje aj všetky analógové signály prechádzajúce logickými členmi obvodov.

2.5.4 Porovnanie existujúcich riešení

V nasledujúcej časti sú analyzované existujúce riešenia porovnané v prehľadnej tabuľke. Pri každom programe sú k nemu uvedené jeho základné výhody a nevýhody.

Program	Výhody	Nevýhody
Logic Gate Simulator	• Jednoduchosť • Intuitívnosť • Prenositeľnosť	• Obmedzená veľkosť pracovnej plochy • Návrh logického obvodu

	• Cena	len zľava doprava • Kreslenie vodičov • Zle vymyslená simulácia • Len dvojvstupové hradlá • Len americké zobrazenie súčiastok
Logicly	• Jednoduchosť • Intuitívnosť • Prenositeľnosť • Rýchle a pohodlné používanie • Cena	• Len dvojvstupové hradlá • Nemožnosť ukladania logického obvodu • Len americké zobrazenie súčiastok
LogicSim	• Jednoduchosť • Prenositeľnosť • Cena • Vytváranie modulov • Veľká množina súčiastok	• Kreslenie vodičov
Simcir	• Jednoduchosť • Intuitívnosť • Prenositeľnosť • Cena	• Obmedzená veľkosť pracovnej plochy • Len dvojvstupové hradlá • Len americké zobrazenie súčiastok
The Logic Lab	• Jednoduchosť • Prenositeľnosť • Cena • Zaujímavá množina súčiastok	• Obmedzená veľkosť pracovnej plochy • Len dvojvstupové hradlá • Svojské zobrazenie súčiastok
Probe	• Jednoduchosť • Prenositeľnosť • Cena • Pravdivostná tabuľka	• Obmedzená veľkosť pracovnej plochy • Len dvojvstupové hradlá • Len americké zobrazenie súčiastok • Prekrývanie súčiastok

		a vodičov
Log	• Cena • Veľká množina súčiastok • Nízka pamäťová náročnosť na pevnom disku • Prehľadnosť zobrazenia	• Ovládanie aplikácie • Náročnosť na CPU • Len americké zobrazenie súčiastok
Deeds	• Cena • Zaujímavá množina súčiastok • Jednoduchosť • Možnosti ladenia nastavení v assembler režime • Výučbové materiály	• Väčšia hardvérová náročnosť • Len americké zobrazenie súčiastok • Obmedzená veľkosť pracovnej plochy • Neznázornenie logickej hodnoty vodiča pri simulácii
LOGiX	• Pohodlná práca • Generovanie logického obvodu • Generovanie pravdivostnej tabuľky • Veľká množina súčiastok	• Cena • Zložitosť – množstvo funkcií • Veľká hardvérová náročnosť
Electronics Workbench Multisim	• Jednoduchosť používania • Veľká množina súčiastok • Intuitívnosť	• Cena • Len americké zobrazenie súčiastok • Veľká hardvérová náročnosť

Tab. 8 Porovnanie existujúcich riešení

2.5.5 Bakalárske práce

2.5.1.1 Virtuálny verifikačný panel s členmi AND-NOR a NAND

Virtuálny verifikačný panel je programový nástroj pre simuláciu logického kombinačného obvodu. Kombinačný obvod v danom programe je možné poskladať iba z logických členov AND-NOR a NAND. Logické členy AND-NOR a NAND spolu tvoria úplný súbor logických členov (úplnú funkčnú množinu).

Program je implementovaný v programovacom jazyku Java. Na ovládanie a prácu s virtuálnym verifikačným panelom slúži prehľadné grafické rozhranie. GUI na prvý pohľad pripomína, dobre známu aplikáciu Log avšak program je až príliš jednoduchý. Používateľovi neumožňuje žiadne pridané funkcie ale iba tie najzákladnejšie potreby pre prácu s virtuálnym verifikačným panelom. Program neumožňuje ukladanie alebo export navrhnutého kombinačného obvodu. Vodivé prepojenia medzi logickými členmi je možné kresliť iba v jednom smere, a to od výstupu logického člena na vstup.

Výhody:

- grafické spracovanie

Nevýhody:

- absentuje možnosť uloženia
- jednosmerné kreslenie vodivých spojov
- žiadna funkcionalita s pridanou hodnotou

2.5.1.2 Virtuálny verifikačný panel s členmi XOR a OR

Ďalším programovým nástrojom pre simuláciu logického kombinačného obvodu je virtuálny verifikačný panel s členmi XOR a OR. Logické členy XOR a OR spolu tvoria úplný súbor logických členov, takže môžeme pomocou nich vyjadriť akúkoľvek booleovskú funkciu.

Program bol naprogramovaný v jazyku Java a používateľovi slúži na prácu s ním prehľadné grafické rozhranie. Po spustení programu sa nám zobrazí matica dvadsiatich logických členov XOR a OR. Grafická stránka programu je spracovaná dobre. Pozitívne hodnotím horný ovládací panel a menu editácie, ktoré poskytuje používateľovi užitočné funkcie pri kreslení navrhnutého obvodu. Menu ponúka možnosti uloženia návrhu vo formáte *.xml*, vtedy je možné návrh opätovne otvoriť a pokračovať v jeho editovaní. Druhou možnosťou je exportovať navrhnutý obvod ako obrázok do súboru .jpg. Menu pre editáciu nám ponúka kreslenie a vymazávanie vodivých spojov, pridávanie pevných vstupov logických členov a vkladanie textových

popisov.

Pri spájaní logických členov vodivými spojmi menia spojenia farbu podľa logickej hodnoty, čo je užitočná funkcia. Taktiež na logických členoch vidíme hodnotu ich výstupu. Vodivé prepojenia medzi logickými členmi je možné kresliť iba v jednom smere, a to od výstupu logického člena na vstup. Keď sa prepneme na mód vymazávania, môžeme vymazať vodivý spoj kliknutím na jeho koniec. Originálna je funkcia, ktorá nám umožní pridať pevný vstup na logický člen a jeho hodnotu môžeme potom meniť. Môže to byť užitočné pri úprave návrhu, alebo odhaľovaní chýb v našom návrhu logického obvodu. Ďalej oceňujem priradenie klávesových skratiek jednotlivým funkciám programu.

Výhody:

- ukladanie rozpracovaných návrhov
- export do formátu *.jpg*
- klávesové skratky
- pridanie pevného vstupu

Nevýhody:

- jednosmerné kreslenie vodivých spojov

2.5.1.3 Virtuálny verifikačný panel s členmi NOR

Tretím skúmaným virtuálnym verifikačným panelom je program umožňujúci simuláciu zapojenia logických členov NOR. Pomocou logických členov NOR vieme vyjadriť akúkoľvek booleovskú funkciu, pretože tvoria úplný súbor logických členov.

Aplikácia bola naprogramovaná v jazyku Java. Hneď po spustení programu aplikácie používateľ vidí, že je veľmi prehľadná a jej používanie je intuitívne. Na rozdiel od predchádzajúcich verifikačných panelov, v tomto programe sa nám na začiatku nezobrazí matica logických členov. Pracovná plocha je prázdna a logické členy na ňu ukladáme výberom z pravého bočného menu. Takéto riešenie poskytuje väčšiu flexibilitu. Taktiež vstupy a výstupy logických signálov si vyberáme z menu. Pozitívom v aplikácii je možnosť kreslenia vodivých spojov obojsmerne, to znamená zo strany vstupu na výstup aj opačne. V bočnom menu ešte nájdeme možnosť prepnutia pohľadu na návrh, či chceme vidieť hodnoty na vodivých spojoch alebo nie. A je tu aj prepínač pre vymazávanie komponentov z návrhu. V hornom menu nájdeme možnosť pre ukladanie a otváranie rozpracovaných návrhov a ďalej tie isté možnosti ako v bočnom menu. Program má podrobne vypracovanú používateľskú príručku - help.

Výhody:

- ukladanie logických členov spôsobom *drag & drop*
- ukladanie rozpracovaných návrhov
- podrobná používateľská príručka
- obojsmerné kreslenie vodivých spojov

3. Špecifikácia

3.1 Funkcionálne požiadavky

Základnými funkcionálnymi požiadavkami vyplývajúcimi zo zadania je tvorba schém logických obvodov. Jedná sa o tvorbu vodivých ciest medzi jednotlivými členmi tvoriacimi dané obvody, možnosťou zmazania už vytvorených ciest, ovládanie virtuálnych tlačidiel zmenou ich stavu medzi stavom vypnutý/zapnutý a verifikácia formou vizualizácie výstupov virtuálnymi žiarovkami stavmi svieti/nesvieti. Dôraz je pritom kladený na jednoduchosť editácie daných obvodov a možnosť hľadania chýb v navrhnutých zapojeniach a bezpodmienečne aj na správnosť zobrazených údajov s nulovou toleranciou chýb, nakoľko sa jedná o verifikačný nástroj, ktorý si nemôže dovoliť zobrazovanie nesprávnych údajov, pretože je referenčným.

Uvedený systém by sa mal čo najviac priblížiť reálnym logickým členom, ktoré sa používajú na simuláciu. Nie je však vzhľadom na rozsah a ohraničenia projektu možné do detailov simulovať všetky vlastnosti reálnych logických členov ako napríklad oneskorenie výstupu vzhľadom na vstup (hodnoty týchto oneskorení u najviac používaných obvodov TTL sa pohybujú rádovo v desiatkach nano sekúnd [2]), zákmity signálov, napájanie logických obvodov, chyby súčiastok ako i mnohé iné veličiny. V prípade potreby je možné dané funkcie členov doplniť dodatočne. Pri hlavnom použití na verifikáciu zapojení a funkcionality návrhu funkcií sú dané veličiny zanedbateľné. Vhodnou vlastnosťou na implementovanie je logický zisk členov, ktorý určuje počet členov, ktoré je možné pripojiť na výstup predchádzajúceho člena.

Logické členy AND, NAND, OR a NOR majú ponúkať voliteľný počet vstupov to znamená vstupný vektor premenných môže mať väčšiu dĺžku ako 2. Vhodnou voľbou je použitie takých hodnôt počtu vstupov, ako sa vyskytujú u reálnych súčiastok členov AND 2, 3, 4, 5, alebo 8 vstupov. Pri zmene počtu vstupov súčiastky je treba dbať na skutočnosť, že nepripojený vstup sa správa nedefinovaným stavom Z (v závislosti od použitej technológie) a preto môže ovplyvniť výstup člena. Napríklad pri pripojení 2 vstupov u 2-vstupového člena bude výsledok korektný. Pri zvýšení počtu vstupov člena sa nám nové nepripojené vstupy nachádzajú v stave vysokej impedancie Z a celkový výsledok člena bude tiež Z. Preto je vhodné tieto vstupy ošetriť a spojiť ich napríklad s iným už pripojeným vstupom.

3.2 Používateľské rozhranie

Z dôvodu jednoduchosti ovládania je vhodné grafické používateľské rozhranie, u ktorého už po spustení programu budú k dispozícii rozmiestnené logické členy podľa voľby používateľa v pomere, ktorý si sám zvolí.

Používateľ má možnosť spájať vstupy členov a virtuálnych žiaroviek s výstupmi členov, tlačidiel a logickými hodnotami 0 a 1. Taktiež má možnosť meniť stavy tlačidiel zapnutý/vypnutý. Výstupy signalizujú žiarovky svojimi stavmi svieti/nesvieti. Taktiež je vhodná možnosť odstránenia ciest, ktoré boli vytvorené chybne, alebo už nie sú potrebné. V rámci zhody s medzinárodnými štandardmi označovania logických členov je vhodné pokiaľ bude systém podporovať americkú normu označovania logických členov. Taktiež je pre potreby verifikácie zapojení a odstraňovania porúch vhodné zobrazovať prehľadným spôsobom vstupné a/alebo výstupné hodnoty každého používaného člena, ako aj zobrazovanie funkcie, ktorú dané zapojenie simuluje. Vhodnou možnosťou je uloženie a neskoršie načítanie vytvorených zapojení.

3.3 Systémové požiadavky

Implementácia systému by mala zaručovať jeho prenositeľnosť medzi rôznymi operačnými systémami i keď sa jeho použitie v rámci cvičení najviac predpokladá na systémoch triedy Microsoft Windows. Z dôvodu obmedzených práv zo strany študentov na počítačoch používaných na cvičeniach je vhodné, aby program nebolo potrebné inštalovať a bol priamo vykonateľný. Samotné rozlíšenie obrazovky aplikácie musí počítať s použiteľnosťou na starších počítačoch v rámci cvičení a je preto vhodné použitie maximálneho rozlíšenia 1024 na 768 obrazových bodov. Procesorová a pamäťová náročnosť by pochopiteľne mala byť na najnižšej možnej úrovni nepresahujúc 32MB RAM pamäti počas behu programu.

3.4 Špecifikácia požiadaviek na verifikačný panel z hľadiska analýzy existujúcich simulátorov logických obvodov

- Jednoduchosť a intuitívnosť – používateľ by mal byť schopný plnohodnotne používať aplikáciu bez potreby študovania návodu na používanie. Aplikácia by mala byť jednoduchá z hľadiska dizajnu a jej funkcie ľahko dostupné z hlavného, prípadne kontextového menu.
- Prenositeľnosť – aplikácia by mala byť rýchlo a jednoducho prenositeľná medzi pracovnými stanicami, či už dostupná priamo cez internet, alebo spustiteľná bez potreby zdĺhavej inštalácie programu. Používateľ by mal byť schopný s aplikáciou pracovať kdekoľvek a kedykoľvek potrebuje.
- Cena – program by mal byť bez poplatkov a voľne šíriteľný, dostupný študentom pre ich potreby.
- Rýchle a pohodlné používanie – Aplikácia by mala byť zhotovená tak, aby používateľ pohodlne a rýchlo mohol dosiahnuť výsledok svojej práce. Táto požiadavka je úzko prepojená s intuitívnosťou, aby používateľ nezabral veľa času nájdením požadovanej funkcie programu.
- Väčšia množina súčiastok – aplikácia by mohla poskytovať používateľovi väčšiu množinu súčiastok, aby

mohol používateľ verifikovať aj zložitejšie obvody (súčiastky ako hodinový signál, preklápacie obvody, displej, ...).

- Neobmedzená veľkosť pracovnej plochy – používateľ by nemal byť obmedzený veľkosťou pracovnej plochy, teda množstvom súčiastok, ktoré môže použiť.
- Viacvstupové hradlá – program by mal poskytovať možnosť si zvoliť počet vstupov pre logické členy.
- Dvojaké zobrazenie súčiastok – súčiastky by malo byť možné zobraziť ako v európskej, tak aj v americkej norme.
- Generovanie pravdivostnej tabuľky – aplikácia by mohla poskytovať možnosť vygenerovania pravdivostnej tabuľky zo zapojenia, ktoré vytvoril používateľ. Táto možnosť by pomohla používateľovi overiť, či sa v zapojení nepomýlil, porovnaním vygenerovanej tabuľky s jeho vlastnou.
- Prehľadné zobrazenie – jednotlivé súčiastky a vodiče by mali byť zobrazené tak, aby sa neprekrývali a aby používateľ mohol jednoznačne identifikovať prepojenie súčiastok.
- Nízka náročnosť na pamäť a CPU – aplikácia by mala byť navrhnutá a implementovaná tak, aby používateľa čo najmenej obmedzovala. A teda by mala mať nízku pamäťovú aj výpočtovú náročnosť. Súvisí to aj s požiadavkou rýchleho používania. Nemalo by sa stať, že počas testovania zapojenia by používateľ musel čakať na výpočet logickej hodnoty prenášanej vodičmi, a pod.
- Znázornenie logickej hodnoty vodiča pri simulácii – Pre jednoduchšiu verifikáciu by mala byť zobrazená logická hodnota prenášaná v danom úseku zapojenia (zmenou farby alebo popisom).

3.5 Špecifikácia požiadaviek dizajnu

Keďže predpokladáme široké spektrum používateľov programu, je potrebné zabezpečiť, aby aplikácia spĺňala základné dizajnové požiadavky. Základnými dizajnovými požiadavkami myslíme grafickú reprezentáciu programu, ako aj použiteľnosť, teda návrh interakcie užívateľa s programom.

Z hľadiska interakcie aplikácie s užívateľom je potrebné dosiahnuť rôzne ciele. Prvým cieľom je jednoduchosť celej aplikácie. Jednoduchosťou sa rozumie nie príliš veľká zložitosť užívateľského rozhrania programu. Celý program musí mať logické rozloženie používateľských prvkov, ktoré budú rozložené do zmysluplných kategórií (napríklad vstupy vľavo, výstupy vpravo, pracovná plocha v strede, logické názvy položiek v menu a ich rozdelenie do logicky pomenovaných kategórií).

Z jednoduchosti vyplýva intuitívnosť aplikácie a naopak. Pokiaľ je program navrhnutý s ohľadom na jeho intuitívnosť, bude sa užívateľom jednoducho ovládať. Intuitívnosť dosiahneme takým rozložením ovládacích prvkov, ktoré je zaužívané v iných programoch, zameraných na rovnaký účel jako naša aplikácia. Intuitívnosť bude teda splnená navrhnutím položiek užívateľského menu podľa štandardov v iných programoch (Súbor ->

Uložiť a podobne), a logickým rozmiestnením logických obvodov a ovládacích prvkov na pracovnej ploche programu. Výsledný program musí taktiež zobrazovať stav obvodu na očakávaných miestach (napríklad v blízkosti logických obvodov a ovládacích prvkov).

Pokiaľ bude program jednoduchý a intuitívny, bude z veľkej časti splnená aj ďalšia požiadavka na dizajn, a to je pohodlnosť používania aplikácie. Ak má užívateľ k dispozícií program s intuitívnym rozložením a použitím a jednoduchosťou, je potrebné z hľadiska pohodlnosti už myslieť len na pokročilé možnosti programu. Medzi takéto možnosti patrí napríklad možnosť ukladať vytvorené zapojenia.

Dobrý dizajn aplikácie je podporený dostupnosťou nápovedy a vysvetliviek cez užívateľské menu aplikácie. Výsledkom bude aplikácia, ktorá bude dobre použiteľná.

Aplikácia musí byť taktiež prehľadná. Prehľadnosť vyplýva a je zabezpečená 41splnením horeuvedených požiadaviek, ako sú jednoduchosť a intuitívnosť. Z hľadiska prehľadnosti je však potrebné, zvlášť si dávať pozor na jednoznačné označenie a odlíšenie ciest aplikácie, a jednoznačné označenie a odlíšenie jednotlivých ovládacích prvkov.

K dobrej použiteľnosti aplikácie je taktiež potrebné, aby bolo možné jednoducho ladiť vytvorené zapojenie. Pohodlná práca s programom bude podporená použitím klávesových skratiek a rýchlosťou aplikácie.

Z grafického hľadiska musí byť program navrhnutý tak, aby jednoznačne odlišoval ovládacie prvky. Všetky prvky by však mali sĺňať určité estetické požiadavky. Grafický návrh jednotlivých ovládacích prvkov by mal byť podobný, čo samozrejme neovplyvňuje funkcionalitu programu, ale celkový dojem z používania programu bude v dôsledku zjednoteného dizajnu lepší. Farba ciest musí byť výrazná oproti farbe pozadia. V ideálnom prípade by si užívateľ mohol vyberať typ pracovného designu celého programu podľa jeho preferencií.

Jednotlivé prvky programu musia byť jednoznačne rozpoznateľné na všetkých nových a drvivej väčšine starých počítačov. Všetky prvky preto musia mať odpovedajúce grafické rozlíšenie.

Program by mal byť optimalizovaný tak, aby aj na počítačoch s nízkym rozlíšením nedochádzalo k deformácií programu, alebo nedostupnosti jednotlivých ovládacích prvkov, kvôli ich umiestneniu mimo dostupnú zobrazovaciu plochu monitoru. Grafická optimalizácia prebehne podľa údajov o používaných rozlíšeniach v tab. 9 a tab. 10.

Dátum / Rozlíšenie	Vyššie	1024x768	800x600	640x480	Neznáme
Jan 1, 2009	57%	36%	4%	0%	3%
Jan 1, 2008	38%	48%	8%	0%	6%
Jan 1, 2007	26%	54%	14%	0%	6%
Jan 1, 2006	17%	57%	20%	0%	6%
Jan 1, 2005	12%	53%	30%	0%	5%
Jan 1, 2004	10%	47%	37%	1%	5%
Jan 1, 2003	6%	40%	47%	2%	5%
Jan 1, 2002	6%	34%	52%	3%	5%
Jan 1, 2001	5%	29%	55%	6%	5%

Jan 1, 2000	4%	25%	56%	11%	4%

Tab. č. 9 Používané rozlíšenia

Dátum / Početfarieb	16,777,216	65,536	256
Jan 1, 2009	95%	4%	1%
Jan 1, 2008	90%	8%	2%
Jan 1, 2007	86%	11%	2%
Jan 1, 2006	81%	16%	3%
Jan 1, 2005	72%	25%	3%
Jan 1, 2004	65%	31%	4%
Jan 1, 2003	51%	44%	5%
Jan 1, 2002	43%	50%	7%
Jan 1, 2001	37%	55%	8%
Jan 1, 2000	34%	54%	12%

Tab. 10 Používané rozlíšenia

3.6 Špecifikácia nových funkcií

Nové funkcie univerzálneho verifikačného panelu budú vychádzať z existujúcich riešení. Porovnaním zistíme, aké funkcie by aplikácia ešte mala obsahovať, ktoré sú či už časovo alebo náročnosťou zvládnuteľné.

Medzi prvé funkcie, ktoré sú potrebné patrí export vytvoreného logického obvodu do formátu obrázka .jpg, ktorý je v súčasnej dobe veľmi používaný a rozšírený. Veľmi dobrou a nápomocnou by bola funkcia, ktorá by vedela vytvorený logický obvod prepísať do jazyka VHDL. Ďalšou funkciou by bolo možnosť skryť nepoužité členy, aby na obrazovke ostali len členy patriace do vytvoreného obvodu. Skrytie týchto členov by spôsobilo, že vyexportovaný obrázok by vyzeral lepšie len s potrebnými členmi. Prijemnou funkciou by bolo aj možnosť vykresliť obvod len z pravdivostnej tabuľky, pripadne vedieť vygenerovať pravdivostnú tabuľku z pravé nakresleného logického obvodu. Nemenej dôležitou funkciou, ktorú by program mal obsahovať je aj podpora klávesových skratiek.

3.7 Zhrnutie

Funkcionálne požiadavky	Používateľského rozhrania	Systémové požiadavky
Tvorba schém logických obvodov	Rozmiestnené logické členy po spustení	Prenositeľnosť
Tvorba ciest medzi členmi, mazanie ciest Zmena stavu tlačidiel vypnutý/zapnutý	Zmena stavu tlačidiel zapnutý/vypnutý Výstupy signalizované stavmi žiaroviek	Priamo vykonateľný program Max. rozlíšenie 1024x768
Verifikácia schémy Jednoduchosť editácie schém	Odstránenie ciest Podpora americkú normu označovania logických členov	Max. použitá pamäť 32 MB
Logický zisk členov Voliteľný počet vstupov logických členov	Zobrazovanie vstupných a výstupných hodnôt členov Uloženie a načítanie vytvorených zapojení	

Tab. 11 Špecifikácia požiadaviek 1

Z hľadiska analýzy existujúcich riešení	Designu	Nových funkcií
Jednoduchosť	Jednoduchosť	Export do obrázku
Intuitívnosť Prenositeľnosť	Logické rozloženie používateľských prvkov Intuitívnosť aplikácie	Generovanie pravdivostnej tabuľky Skryť nepoužité členy
Cena Rýchle a pohodlné používanie	Pohodlnosť používania Nápoveda a vysvetlivky	Prepis do jazyka VHDL Klávesové skratky
Väčšia množina súčiastok Neobmedzená veľkosť pracovnej plochy	Prehľadnosť Jednoznačné odlíšenie ovládacích prvkov	
Viacvstupové hradlá Dvojaké zobrazenie súčiastok	Odpovedajúce grafické rozlíšenie	
Generovanie pravdivostnej tabuľky Prehľadné zobrazenie		
Nízka náročnosť na pamäť a CPU		

Tab. 12 Špecifikácia požiadaviek 2

4. Návrh

4.1 Analýza špecifikovaných požiadaviek

Na základe definovaných požiadaviek realizujem návrh riešenia začínajúci systémovými požiadavkami, ktoré určujú základnú vlastnosť ktorou je voľba vývojového prostredia.

4.1.2 Systémové požiadavky

Vzhľadom na predchádzajúce skúsenosti s programovacími technológiami, používaním systému pod OS Windows, použitie grafického používateľského rozhrania a rozšírenosti daných technológií v súčasnosti sa ako najvhodnejšie technológie na implementáciu javia C++, Java, .NET. S prihliadnutím na platformovú nezávislosť aplikácie a používanie systému aj študentmi na štúdium aj na domácich počítačoch, kde môžu používať iný operačný systém je najvhodnejším použitie technológie Java, ktorá je platformovo nezávislá. Jedinou nevýhodou je potreba mať nainštalovaný Java prekladač, ktorý je z dôvodu rozšírenosti tejto technológie na internetových stránkach bežne nainštalovaný na väčšine osobných i prenosných počítačoch. Týmto bol zvolený objektovo-orientovaný prístup k vývoju aplikácie. Vhodnou optimalizáciou zdrojového kódu je tiež možné dosiahnuť veľmi nízku náročnosť na použitie procesora ako i pamäte RAM počítača. Rozlíšenie obrazovky by malo mať 1024x768 obrazových bodov s dôvodu kompatibility so staršími počítačmi. Uvedené rozlíšenie postačuje na rozmiestnenie dostatočného počtu logických členov i ostatných potrebných prvkov na danú plochu. Tým, ale nie je vylúčené zväčšiť dané rozlíšenie v prípade potreby do budúcna, prípadne vygenerovať veľkosť pracovnej plochy podľa toho, aké rozlíšenie používateľ má na svojej pracovnej stanici.

4.1.3 Používateľské rozhranie

V hornej časti obrazovky sa bude nachádzať Menu podporujúce klávesové skratky s týmito voľbami:

A. Súbor

a) Nový panel (Ctrl+N)

Otvorí nový panel pre kreslenie ďalšieho zapojenia. Pred jeho spustením sa spýta na typ panelu a pomer rozloženia súčiastok. V prípade, že existujúce zapojenie nie je uložené upozorní na to.

b) Uložiť zapojenie (Ctrl+S)

Umožňuje uložiť existujúce zapojenie do súboru.

c) **Uložiť ako obrázok** (Ctrl+I)

Umožňuje uložiť existujúce zapojenie ako obrázok vo formáte .jpeg, .png, .gif, alebo .bmp. Pri ukladaní umožní skryť nepoužité členy.

d) **Export do VHDL** (Ctrl+V)

Umožňuje export existujúceho zapojenia opísaním jazykom VHDL

e) **Otvoriť zapojenie** (Ctrl+O)

Otvorí uložené zapojenie zo súboru

f) **Koniec** (Ctrl+X)

Ukončí program. V prípade, že nie je existujúce zapojenie uložené upozorní na to.

B. Nastavenia

a) **Zobrazovanie** (Ctrl+Z)

Obsahuje nastavenia zobrazovania informácií, či už voľbu americkej normy súčiastok, zobrazovanie vstupných a výstupných hodnôt a zobrazovanie výstupnej funkcie.

b) **Dizajn aplikácie** (Ctrl+D)

Umožňuje zvoliť vzhľad aplikácie z hotových dizajnov a nastaviť farby ciest podľa ich stavu.

c) **Logický zisk** (Ctrl+L)

Zobrazí okno pre zadanie novej hodnoty logického zisku všetkých členov.

d) **Zjednodušovať funkcie** (Ctrl+J)

Povolí, alebo zakáže zjednodušovanie funkcií na výstupe.

C. Zobraziť

a) **Pravdivostnú tabuľku** (Ctrl+P)

Zobrazí pravdivostnú tabuľku aktuálneho zapojenia.

b) **Funkciu** (Ctrl+F)

Zobrazí logickú funkciu na každej žiarovke.

c) **Karnaughova mapa** (Ctrl+K)

Zobrazí Karnaughovu mapu aktuálneho zapojenia.

D. O programe

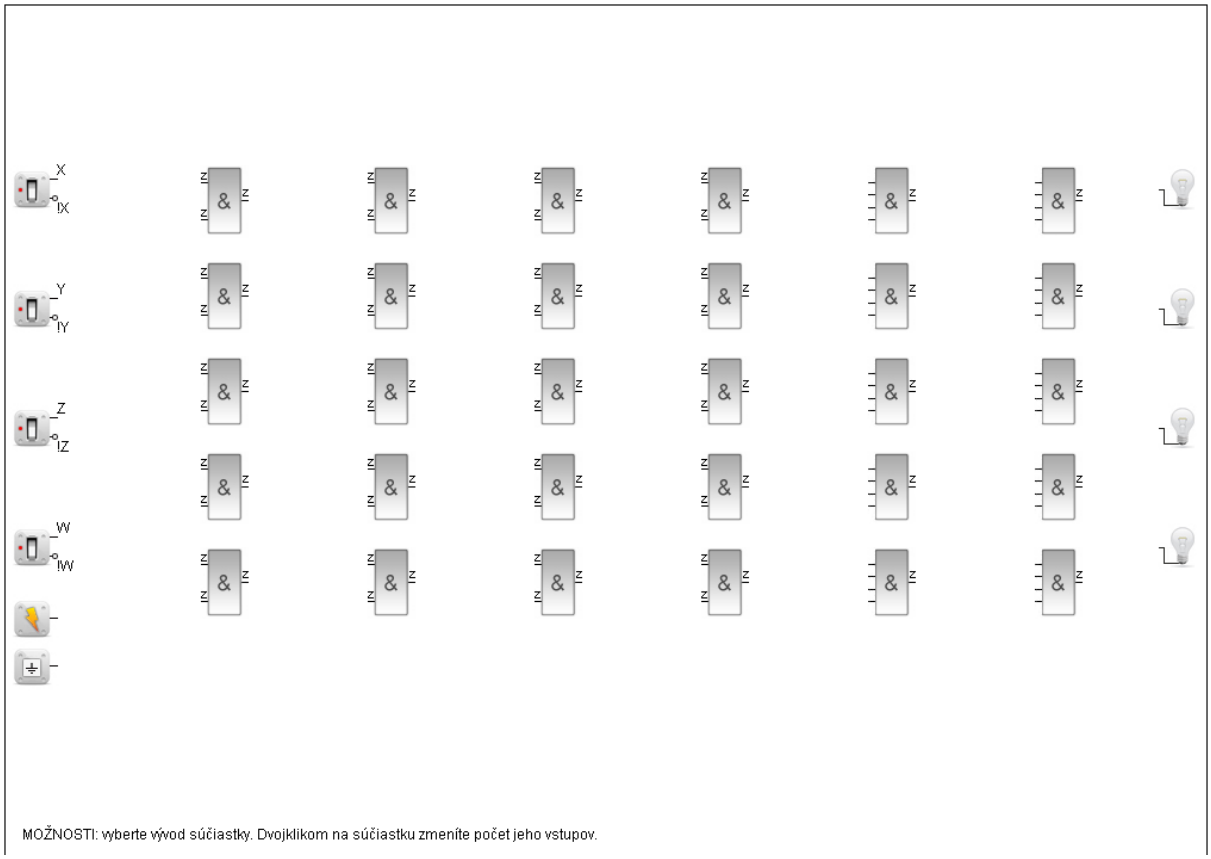
Základné informácie o programe a help slúžiaci ako používateľská príručka.

a) **Autori** (Ctrl+A)

b) **Program** (Ctrl+T)

c) **Help** (Ctrl+H)

Pod menu bude hlavné plátno vykresľujúce danú schému, na ktorom sa ihneď po spustení programu zobrazia logické členy, ktoré sú používateľovi k dispozícii. Vľavo umiestnené virtuálne tlačidlá, ktoré budú ponúkať vstupné premenné v priamom i negovanom tvare, aby nebolo potrebné používať dodatočné negácie. Pod tlačidlami v prípade potreby daného panelu umiestnené tiež hodnoty logickej 1 a 0, nakoľko napríklad uvedená kombinácia AND, XOR sa stáva úplnou funkčnou množinou až s použitím logickej jednotky. Umiestnenie vľavo je z toho dôvodu, že aj texty sa u nás čítajú zľava doprava a tým sú používatelia zvyknutí čítať schémy týmto smerom od vstupu po výstup. Uvedený výstup budú vpravo tvoriť 4 virtuálne žiarovky, ktoré sa po pripojení logickej jednotky rozsvietia. Vodivé cesty medzi členmi budú tvoriť nakreslené čiary. Na obrázku č. 26 je zobrazený návrh používateľského rozhranie pre aplikáciu.



Obr. 26 Návrh používateľského rozhrania.

V spodnej časti pod vykreslenou schémou bude informačný panel vypisujúci aktuálnu voľbu používateľa

ktorú spravil. Taktiež chyby a upozornenia vypísané červenou farbou. Bolo by vhodné upozornenia vypisovať aj novootvoreným varovným oknom, kde by bolo potrebné potvrdiť toto okno, aby sa zatvorilo no z hľadiska interakcie programu s používateľom by to pôsobilo skôr rušivo a obmedzujúco. Taktiež používateľ u danej aplikácie a tvorby schém, ktorá slúži práve na cvičenie si naučených vedomostí, nemá čo pokaziť a každá zlá voľba sa dá jednoducho dodatočne opraviť. Posledný riadok informačného panelu tvorí riadok „Možnosti“ zobrazujúci aké voľby a možnosti má pri tvorbe a úprave schémy používateľ k dispozícii v závislosti od aktuálne vytvoreného zapojenia. V spodnej časti pod aktuálnou schémou sa tiež budú zobrazovať logické funkcie na vybratej ceste, nakoľko ich vypisovanie pri dlhších funkciách priamo v schéme by bolo neprehľadné.

4.1.4 Funkcionálne požiadavky

Základnou funkcionálnou požiadavkou je samotné správanie sa logického člena, ktoré by malo byť ľahko verifikovateľné. Z toho dôvodu je vhodné u každého člena ponúknuť zobrazenie vstupných a výstupných hodnôt daného člena. Taktiež systém ponúkne možnosť zobrazovať farbu vodivej cesty, spájajúcej prvky obvodu podľa aktuálne logickej hodnoty, ktorá prechádza daným vodičom po príslušnou spustení tejto funkcie v Menu. V tomto prípade bude červená farba signalizovať logickú jednotku, modrá farba logickú nulu a čierna farba nedefinovaný stav vysokej impedancie Z, pričom uvedené hodnoty je možné voľbou cez Menu zmeniť. Vodivú cestu bude možné vytvoriť dvomi kliknutiami, pričom jedno definuje začiatok cesty a druhé koniec, pričom jedno kliknutie musí určovať výstup člena a druhé jeho vstup. Výber týchto prvkov bude možný v obidvoch možných poradiach. Spájanie dvoch výstupov, alebo dvoch vstupov nemá z pohľadu logických obvodov opodstatnenie a preto nebude možné. Pokiaľ u niektorého z výstupov kde chceme pripojiť vstup už je pripojený nejaký iný člen je možné ako cieľ cesty vybrať aj už vytvorenú cestu, pričom aplikácia nájde pri vytváraní cesty najbližšiu možnú cestu k danej vodivej ceste, kde sa na ňu pripojí a v mieste pripojenia vytvorí uzol. Z dôvodu možného väčšieho počtu vytvorených ciest je pre jednoduchú orientáciu možné vysvietenie konkrétnej cesty jednoduchým kliknutím na cestu, ktorá sa zvýrazní inou farbou. Pokiaľ sa jedná o časť cesty medzi výstupom člena a uzlom, ktorá je spoločná pre viacero členov budú vysvietené všetky cesty pripojené na tento výstup súčiastky. Týmto vieme jednoducho zistiť ktorý výstup je na konkrétny vstup člena pripojený ako i to, kde je pripojený výstup z daného člena.

Zmazať vytvorenú cestu je možné kliknutím pravým tlačidlom myši na danú cestu. Pokiaľ sa jedná o spoločnú časť cesty medzi uzlom a výstupom člena je vypísaná chybová správa a cesta nie je vymazaná, nakoľko nevieme presne určiť, ktorú z ciest chce používateľ odstrániť. Pri výbere ciest kliknutím na ne, či už pri pripájaní ďalšej cesty, zvýraznení cesty, alebo jej odstránení bude možné kliknúť aj o 1 pixel okolo danej cesty, nakoľko presne sa trafiť na vykreslenú cestu môže byť problém a nemusí sa to vždy podariť. Taktiež pri výbere

vstupu a výstupu členov nebude potrebné kliknúť presne na vykreslený vývod ale bude možná tolerancia až do 5 obrazových bodov okolo vývodu, u mnohovstupových súčiastok (AND, OR, NAND, NOR) bude táto hranica pochopiteľne s priestorových dôvodov nižšia.

Zmena stavu vstupu ako aj prepočítanie celého stavu výstupu sa uskutoční ihneď po pripojení cesty k danej súčiastke. Taktiež zmena stavu žiarovky nastane okamžite. Zmeniť stav tlačidla bude možné jednoduchým kliknutím na vykreslené tlačidlo, čím dôjde k preklopeniu jeho stavu zapnuté/vypnuté. Zmena počtu vstupov členov, ktoré túto funkciu umožňujú, je možná dvojklikom na daný člen čím sa zobrazí dialógové okno voľby počtu vstupov pre daný člen. Táto zmena však nie je možná, pri už pripojenom niektorom zo vstupov daného člena, nakoľko aj u reálnych obvodov nie je možné ich vymeniť pri už pripojených vstupoch.

Program tiež bude počas behu vytárať súbor s názvom *log.file*, do ktorého bude ukladať všetky voľby používateľa pre neskoršiu analýzu, odstraňovanie nečakaných stavov a prípadnú rekonštrukciu správania používateľa pre výskumné účely prípadného zdokonaľovania aplikácie.

4.2 Hrubý návrh riešenia

Z dôvodu voľby objektovo-orientovaného jazyka Java bude aj návrh riešenia orientovaný objektovo.

4.2.1 Objekty

Každý prvok z členov schémy t.j. logické členy, tlačidla, žiarovky i vodivé cesty budú predstavovať samostatné objekty programu. Interakciu medzi objektmi zabezpečujú ich metódy.

Objekty používané v aplikácií sú nasledovné:

1. **and** – objekt súčiastky logického člena and s voliteľným počtom vstupov o hodnotách 2, 3, 4, 5 a 8 vstupov a jedným výstupom. Vnútornou funkciou objektu je vynásobiť vstupné logické hodnoty a upraviť aj vstupné reťazce na výstupný ako súčin vstupných funkcií.
2. **or** – objekt súčiastky logického člena or s voliteľným počtom vstupov o hodnotách 2, 3, 4, 5 a 8 vstupov a jedným výstupom. Vnútornou funkciou objektu je sčítať vstupné logické hodnoty a upraviť aj vstupné reťazce na výstupný ako súčet vstupných funkcií.
3. **nand** – objekt súčiastky logického člena nand s voliteľným počtom vstupov o hodnotách 2, 3, 4, 5 a 8 vstupov a jedným výstupom. Vnútornou funkciou objektu je vynásobiť vstupné logické hodnoty a tento výsledok znegovať, tiež upraviť aj vstupné reťazce na výstupný ako negovaný súčin vstupných funkcií.

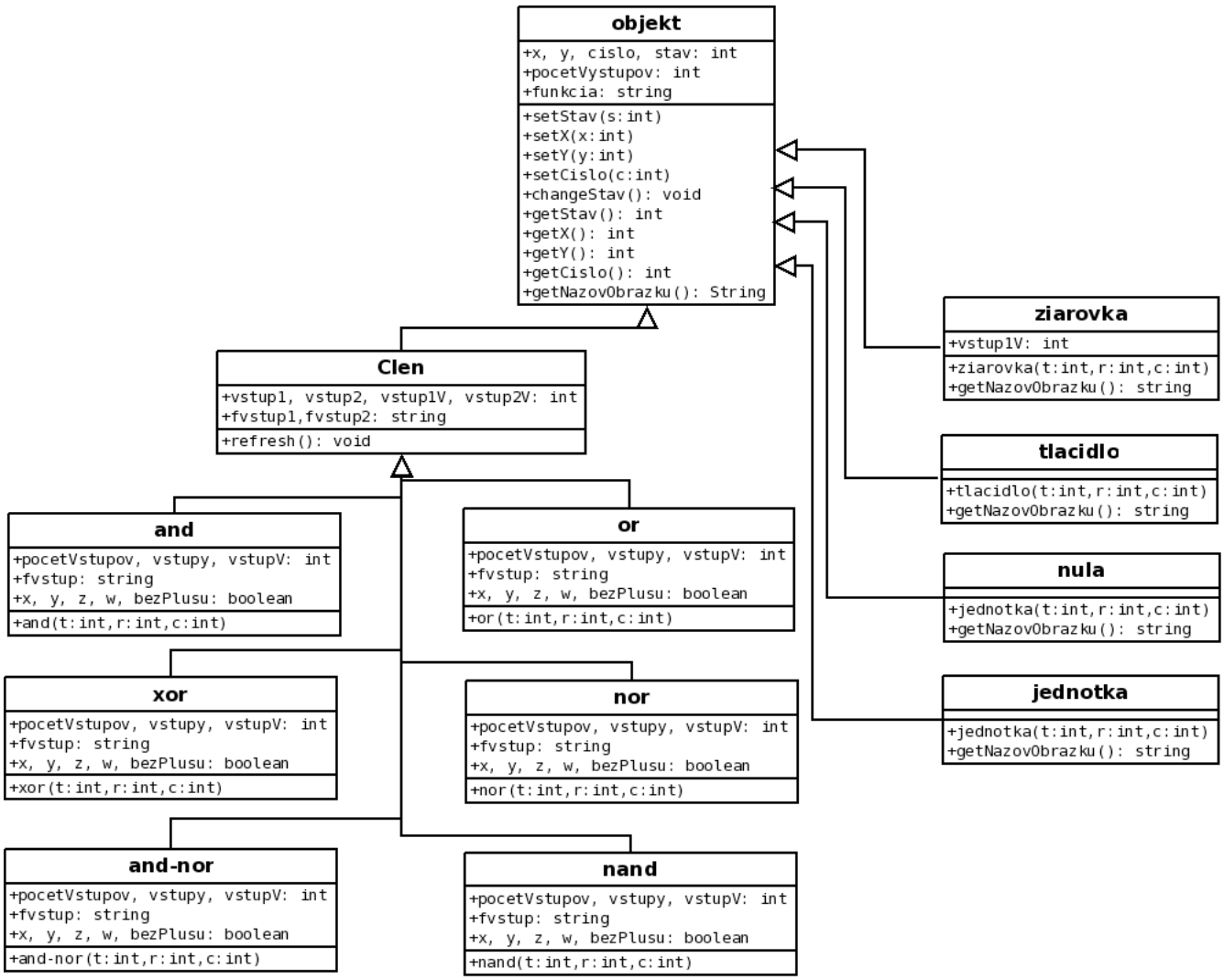
4. **nor** – objekt súčiastky logického člena nor s voliteľným počtom vstupov o hodnotách 2, 3, 4, 5 a 8 vstupov a jedným výstupom. Vnútornou funkciou objektu je sčítať vstupné logické hodnoty a výsledok znegovať, tiež upraviť aj vstupné reťazce na výstupný ako negovaný súčet vstupných funkcií.
5. **not** – objekt súčiastky logického člena not s jedným vstupom a jedným výstupom. Vnútornou funkciou objektu je znegovať vstupnú logickú hodnotu a upraviť aj vstupný reťazec na výstupný znegovaný.
6. **xor** – objekt súčiastky logického člena xor s dvomi vstupmi a jedným výstupom. Vnútornou funkciou objektu je zo vstupných logických hodnôt spraviť neekvivalenciu čiže logickú nezhodu a upraviť vstupné reťazce funkcií pridaním znaku plus v krúžku a celý tento výraz uzatvoriť do zátvoriek z dôvodu neskoršej možnej úpravy v nasledujúcom člene.
7. **tlačidlo** – objekt reprezentujúci virtuálne tlačidlo. Uchováva svoj vnútorný stav, prednastavený na logickú nulu. Pri kliknutí na obrázok tohto tlačidla preklopí svoj stav na opačný. Výstupy poskytuje v priamom tvare. Dojem negovaného tvaru bude tvoriť ďalšie tlačidlo, ktorého stav sa bude meniť presne opačne ako tlačidla, ktoré bude predstavovať priamy tvar. Táto možnosť bola zvolená z toho dôvodu, že už hlavná trieda *objekt* definuje súčiastky ako objekty iba s jedným výstupom. Dojem dvoch výstupov tak budú tvoriť dva navzájom prepojené objekty.
8. **žiarovka** – objekt predstavujúci virtuálnu žiarovku, ktorý bude meniť svoj názov obrázku v závislosti od stavu v ktorom sa nachádza. Svoj stav bude meniť podľa logickej hodnoty výstupu pripojenej súčiastky k žiarovke. V prípade, že vodivá cesta nebude k žiarovke pripojená bude sa nachádzať v stave nula.
9. **jednotka** – jednoduchý objekt, ktorého vnútorný stav bude vždy logická jednotka a bude poskytovať iba jeden výstup.
10. **nula** – objekt predstavujúci hodnotu logickej nuly, vnútorný stav objektu bude trvale nastavený na nulu.
11. **cesta** – objekt predstavujúci práve jednu vodivú cestu medzi dvomi súčiastkami. Bude si pamätať číslo výstupu aj vstupu kde je pripojená. U každej cesty je potrebné vypočítať kadiaľ sa bude vykresľovať. Predstavuje jediný objekt v rámci schémy, ktorý je možné vymazať s čím treba počítať pri vytváraní a odstraňovaní inštancií tohto objektu. Taktiež pri pripojení viacerých ciest na rovnaký výstup člena treba spojiť viacero ciest idúcich rovnakým smerom do jednej cesty a pri ich rozdvojení treba vykresliť uzol.

Okrem uvedených objektov sa v programe ešte budú nachádzať objekt plátna, na ktoré sa bude schéma vykresľovať a tiež objekty dialógových okien použitých počas fungovania programu, ktoré nie je potrebné podrobne opisovať a sú základnou súčasťou jazyka Java.

Aktívne prvky schémy ako logické členy, tlačidlá, logické hodnoty 1 a 0 i žiarovky budú mať spoločného predka s názvom *Objekt*. Tento objekt bude združovať všetky vlastnosti spoločné pre všetky aktívne prvky

schémy ako pozíciu daného prvku na plátne, logický stav v ktorom sa prvok nachádza (je totožný s výstupom logickej hodnoty člena, pokiaľ má člen výstup), logickú funkciu ako reťazec znakov na výstupe člena a počet pripojených členov k výstupu. Nezobrazuje všetky súčiastky v dôvodu prehľadnosti, ale všetky členy sú rovnaké ako člen and, pokiaľ nie je v schéme uvedené inak.

Na obrázku č. 27 je zobrazený fyzický model, ktorý predstavuje základný návrh tried programu



Obr. 27 Fyzický model

Od tejto triedy *Objekt* dedí abstraktná trieda s názvom c*len,* ktorá je nadtriedou pre konkrétne objekty *and, or, nand, nor, not* a *xor*. Nadtrieda *clen* dopĺňa vlastnosti dvoch vstupov, logických funkcií týchto vstupov ako reťazce, a tiež si pamätá, ktoré súčiastky sú pripojené na jej vstupy, čo je potrebné pre ovzorkovanie vstupov, aby sme pri prepočte stavov vedeli od ktorého prvku treba načítať jeho aktuálny stav. Objekty cesty a dialógových okien nie sú v hierarchii dedenia zobrazené, nakoľko do danej hierarchie nepatria a predstavujú samostatné objekty nezávislé od uvedených.

4.2.2 Návrh riešenia špecifických funkcií

Voľba panelu – ihneď po spustení aplikácie dostane používateľ na výber voľbu typu verifikačného panelu pomocou výberových tlačidiel.

Pomer súčiastok – v prípade, že zvolený panel obsahuje viac ako jeden typ súčiastky bude mu ponúknuté vybrať si presný počet jednotlivých členov s preddefinovanými hodnotami. Následne sa zobrazí zvolený panel.

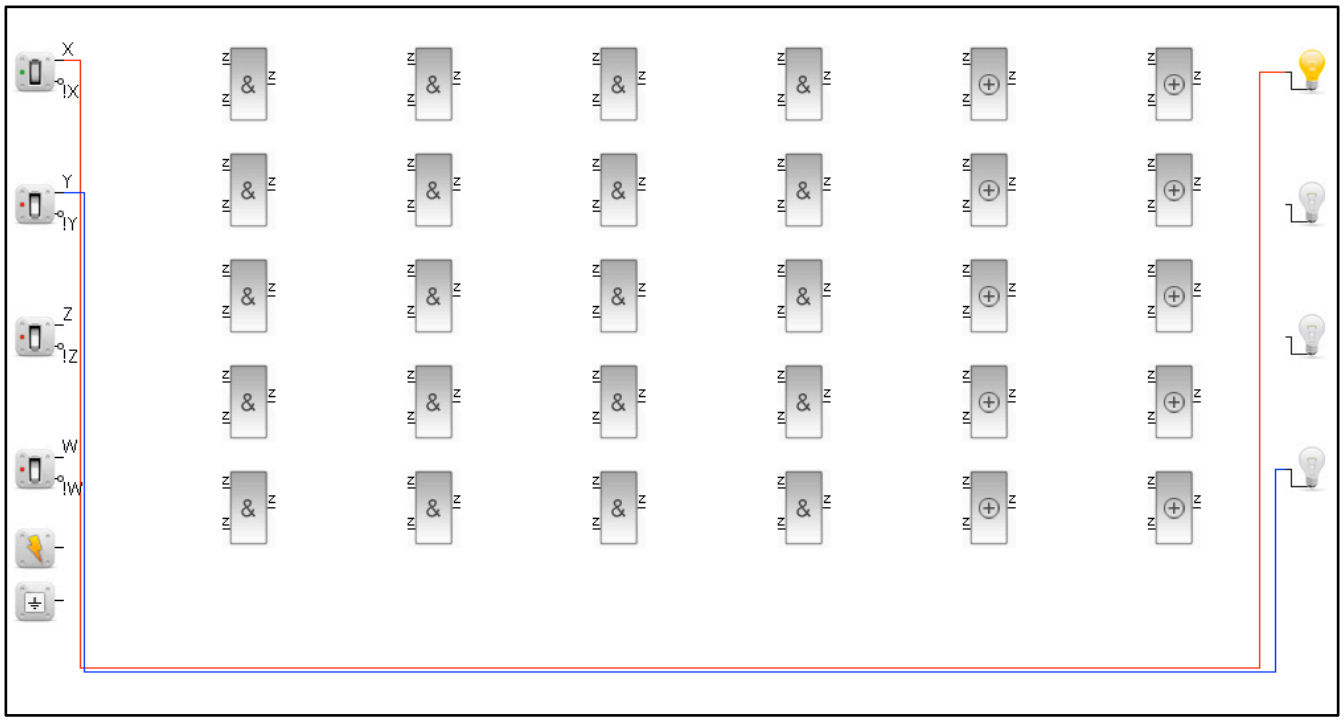
Klávesové skratky – počas celého behu programu bude možné použiť klávesové skratky, ktoré budú rovnocenné z voľbou ponuky z Menu aplikácie. Každá klávesová skratka bude vypísaná za danou možnosťou Menu a tiež popísané v Help časti aplikácie.

Uložiť ako obrázok – v prípade tejto voľby z Menu aplikácie sa používateľovi ponúkne, aby zvolil formát obrázku, v ktorom chce zapojenie uložiť. Tiež bude ponúknutá možnosť, či chce zo zapojenia odstrániť nepoužité a nepripojené členy, aby bolo zapojenie prehľadnejšie.

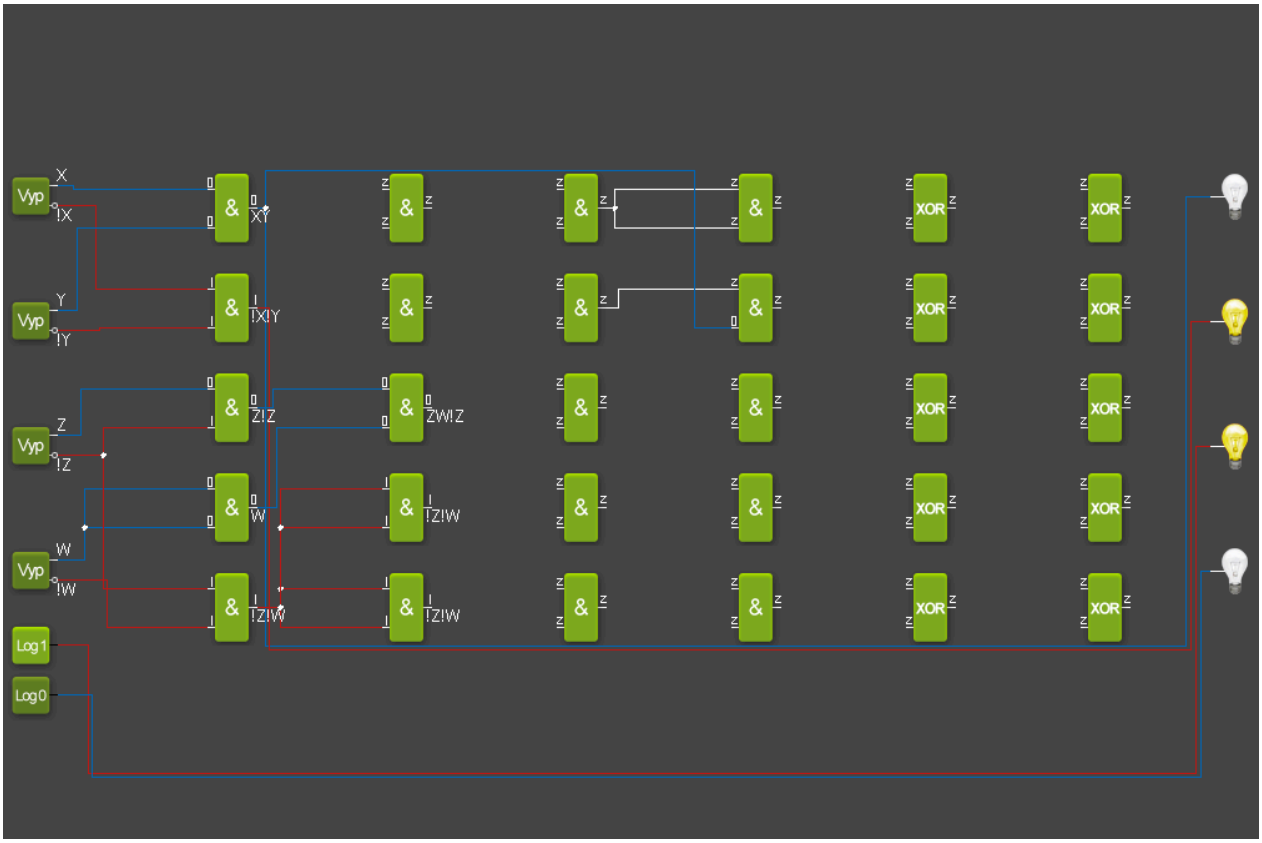
Verifikácia zapojenia – pre rýchle overenie správnosti zapojenia bude program schopný vygenerovať pravdivostnú tabuľku aktuálneho zapojenia, tiež jeho Karnaughovu mapu alebo funkciu na jednotlivých žiarovkách. Tým nie je potrebné prechádzať všetky kombinácie tlačidiel a správnosť kontroly daného zapojenia zo zadaním úlohy je okamžitá. Daný typ overenia sa zobrazí po voľbe z Menu v novom okne.

4.2.3 Návrh vizuálnej stránky programu

Nasledujúce dva obrázky (obrázok č. 28 a č. 29) predstavujú návrh vzhľadu verifikačného panelu. Vo výslednom programe by mal mať užívateľ na výber, ktorý z možných zobrazení programu preferuje a nastaviť si daný vzhľad.

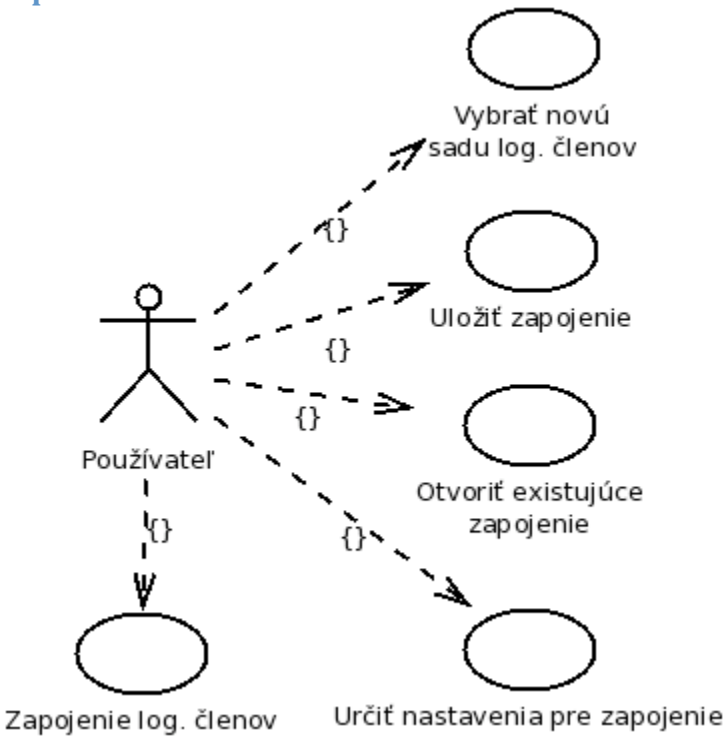


Obr. 28 Návrh vizuálnej stránky programu



Obr. 29 Návrh vizuálnej stránky programu

4.2.4 Model prípadov použitia

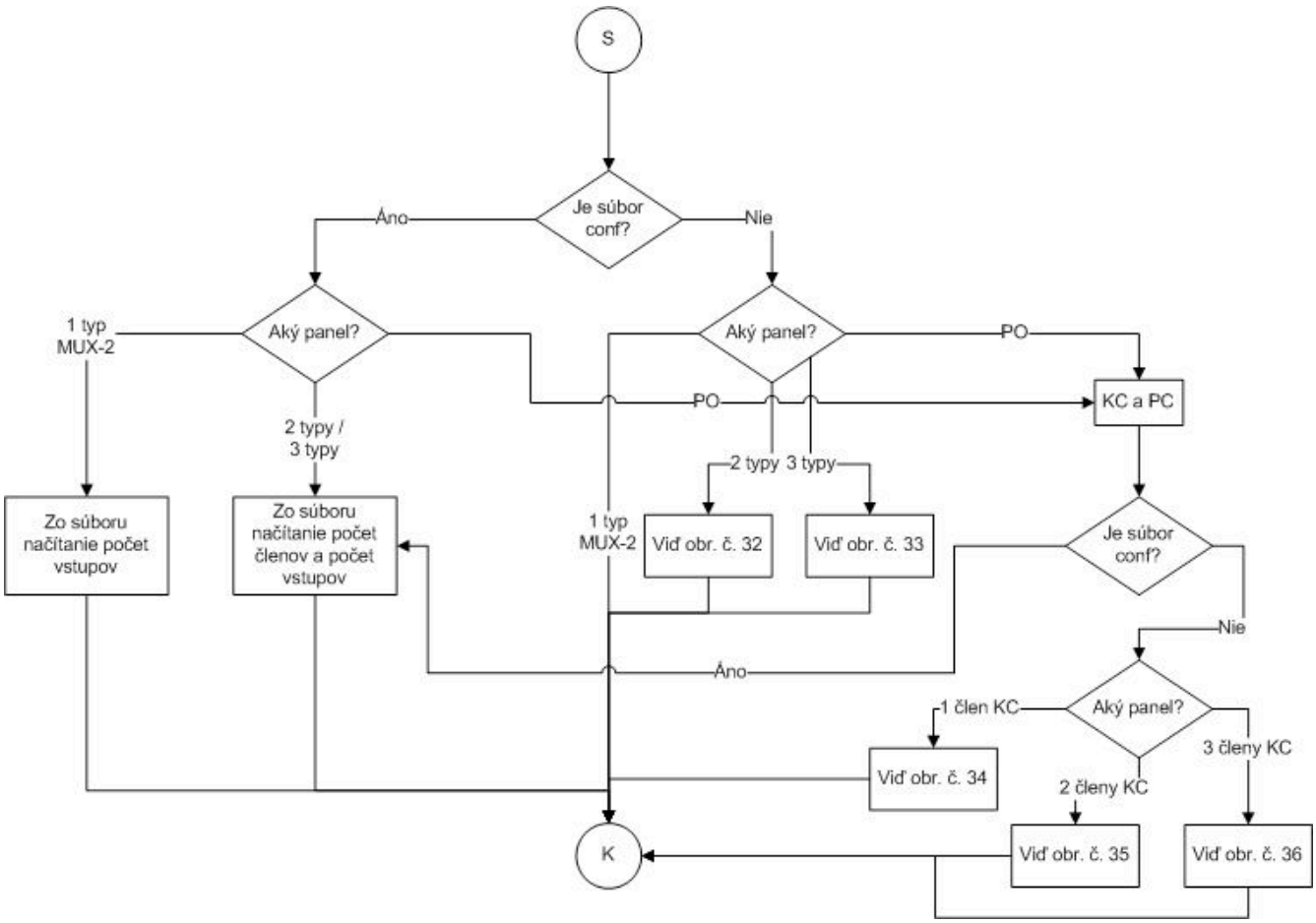


Obr. 30 Model prípadov použitia

Model prípadov použitia zobrazený na obrázku č. 30 približuje pohľad na služby, ktoré bude poskytovať používateľovi výsledná aplikácia. Hlavnými službami programu bude poskytnúť používateľovi vhodné prostredie pre zapájanie logických členov, možnosť vybrať si úplný súbor logických členov, s ktorými chce pracovať a pohodlne určovať nastavenia daného zapojenia. Ďalšie dôležité prípady použitia zvyšujúce pohodlie pri používaní aplikácie sú možnosť uložiť zapojenie a otvoriť existujúce zapojenie.

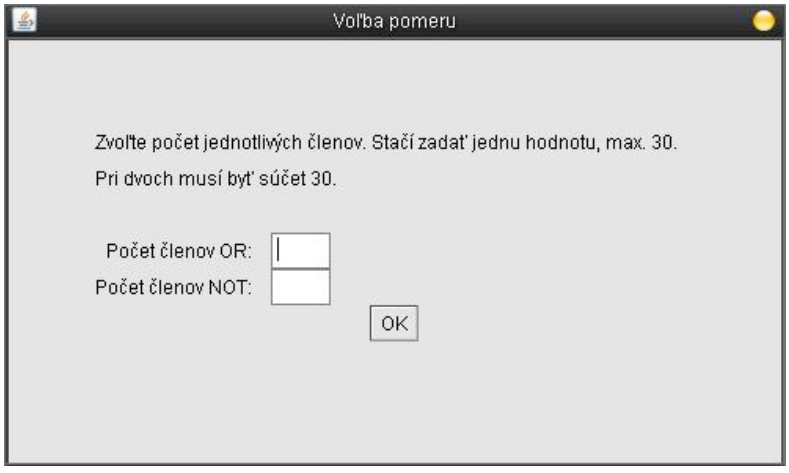
4.3 Voľba panelu

Na nasledujúcom obrázku č. 31 sa nachádza diagram, ktorý predstavuje rôzne spôsoby zobrazenia okna, v ktorom si užívateľ volí počet jednotlivých členov. To či sa dané okno objaví alebo nie, resp. pre aké členy sa bude dať zvoliť pomer, závisí od toho či existuje konfiguračný súbor conf.file a podľa zvoleného typu panelu.

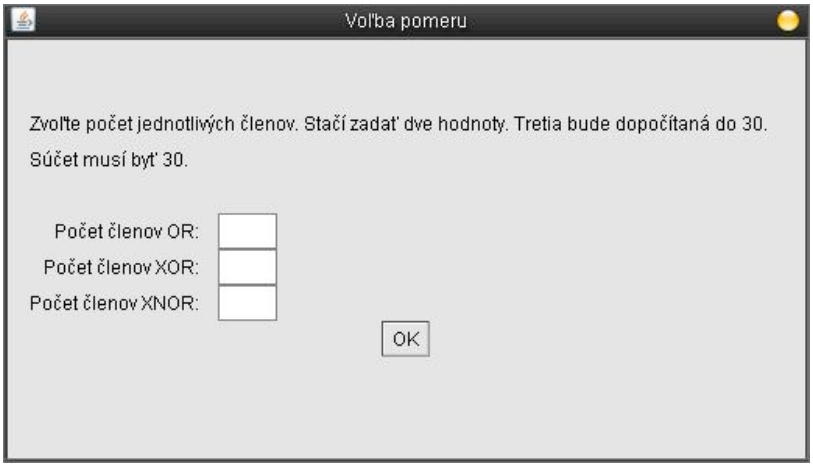


Obr. 31 Diagram voľby pomeru

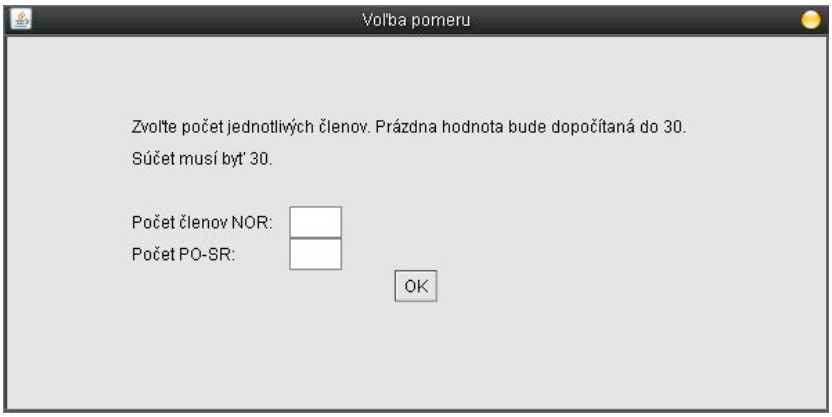
Na nasledujúcich obrázkoch č. 32 až č. 36 je možné vidieť rôzne spôsoby voľby pomeru na základe vybraného panelu.



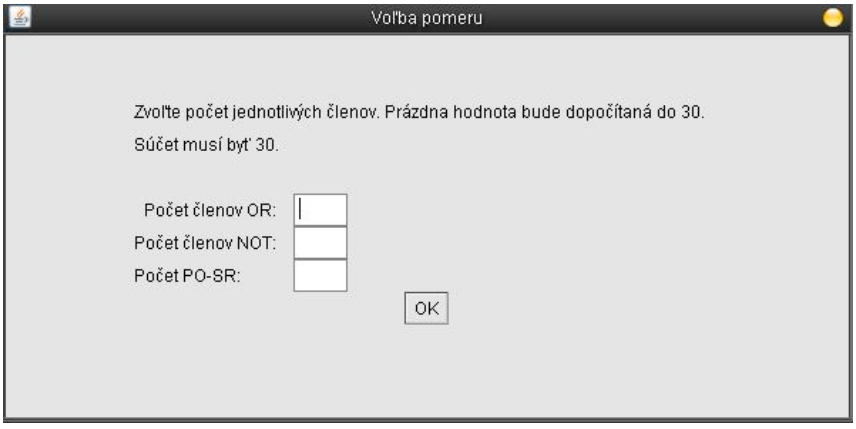
Obr. 32 Voľba pomeru pre dva typy členov



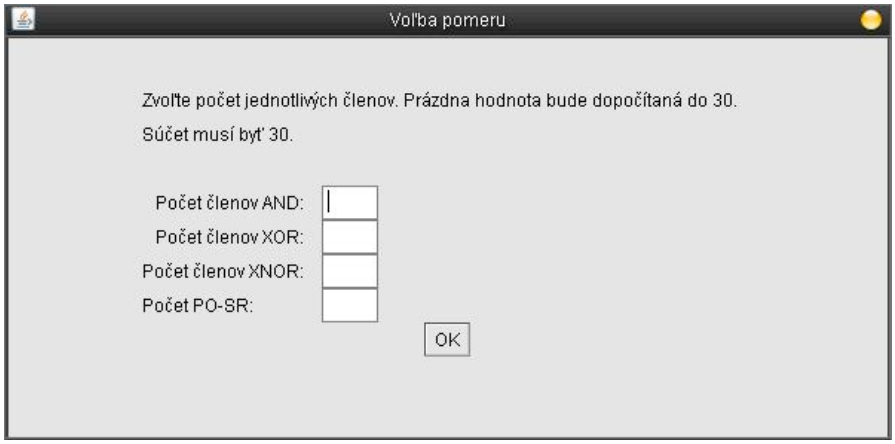
Obr. 33 Voľba pomeru pre tri typy členov



Obr. 34 Voľba pomeru pre jeden kombinačný člen a jeden preklápací člen



Obr. 35 Voľba pomeru pre dva kombinačné členy a jeden preklápací člen



Obr. 36 Voľba pomeru pre tri kombinačné členy a jeden preklápací člen

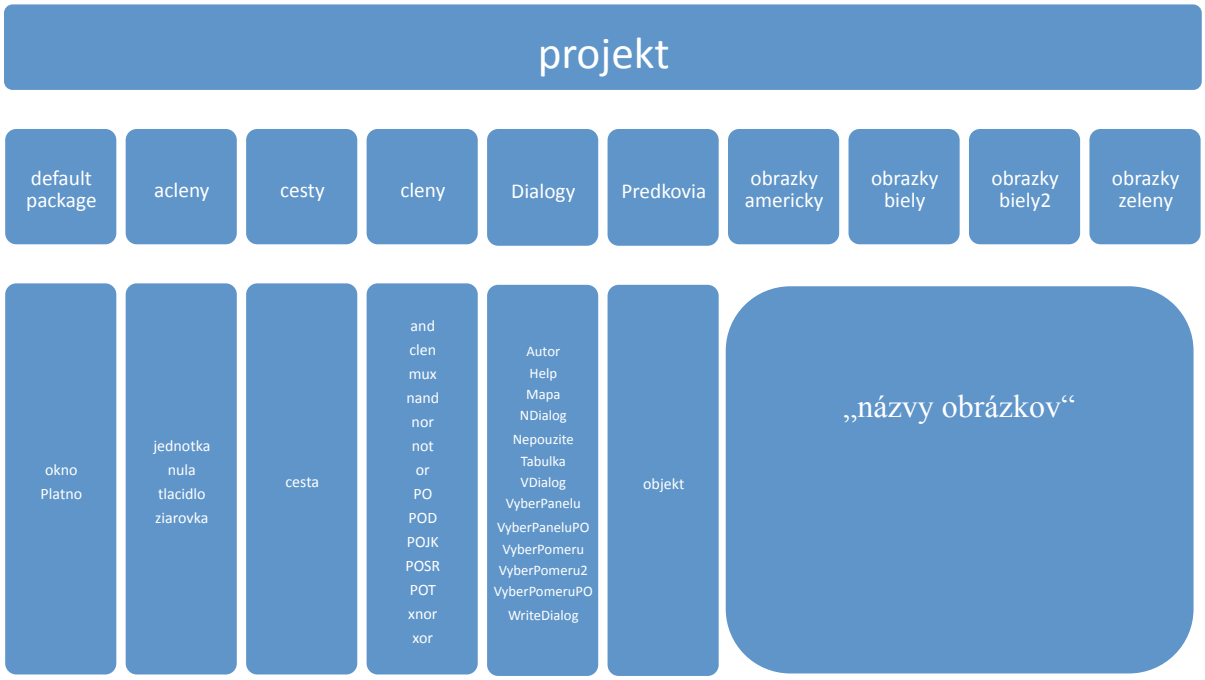
5. Implementácia

V tejto kapitole je opísaná architektúra vytvoreného systému, použité implementačné prostredie, zhrnutie implementovanej funkcionality, ako aj opísaný princíp niektorých použitých algoritmov.

5.1 Architektúra systému

Vytvorený systém bol implementovaný v prostredí operačného systému Microsoft Windows XP v programovacom jazyku Java vo vývojovom prostredí Eclipse. Týmto výberom je zaručená platformová nezávislosť, čiže systém je použiteľný pod rôznymi operačnými systémami. Na týchto operačných systémoch je potrebné, aby bola nainštalovaná Java, ktorá je k dispozícii väčšinou ako aktualizácia operačného systému, alebo je priamo zabudovaná do novších operačných systémov. Tento potrebný Java softvér sa tiež nazýva „Java Runtime Environment“ (JRE), „Java Virtual Machine“ (JVM), alebo „Java download“.

Systém je kvôli prehľadnosti, ale aj kvôli modulárnosti rozdelený na niekoľko častí prostredníctvom balíkov. Tieto balíky obsahujú navzájom priamo súvisiace komponenty (triedy, príp. obrázky). Môžeme ich rozdeliť do dvoch základných skupín, a to balíky obsahujúce triedy, ktoré predstavujú implementované súčasti systému, a balíky obsahujúce obrázky. Pre každý dizajn aplikácie je vytvorený nový balík s obrázkami obsahujúci iné znázornenie súčiastok systému. Obrázky pre rôzny dizajn sú v rôznych balíkoch, pretože tieto obrázky musia mať z hľadiska implementácie systému rovnaké názvy. Na obrázku (Obr. 37) je znázornená architektúra systému. V prvom riadku je názov projektu, v druhom názvy balíkov a v treťom súbory, ktoré balíky obsahujú.



Obr. 37 Architektúra systému

Default package

Obsahuje dve hlavné triedy aplikácie - *okno* a *Platno*. Základná trieda *okno*, obsahujúca aj spúšťaciu metódu *main*, slúži na zobrazenie všetkých prvkov v hlavnom okne typu „Frame“. Na zobrazenie prvkov používa prostredie „awt“. Obsahuje pomocné metódy, ako aj podtriedy slúžiace na reakcie tlačidiel prostredia. Trieda *Platno* slúži na zobrazenie plátna, na ktorom sa vykresľujú všetky vytvorené a upravované prvky. Trieda svojimi metódami zabezpečuje ich vykresľovanie a prekresľovanie využívajúc prostredie „awt“.

Acleny

Tento balík obsahuje triedy reprezentujúce tieto súčiastky:

napájacie napätie (logická jednotka) – reprezentuje trieda *jednotka*

uzemnenie (logická nula) – reprezentuje trieda *nula*

prepínač logickej hodnoty – reprezentuje trieda *tlacidlo*

zobrazovač logickej hodnoty (dióda) – reprezentuje trieda *ziarovka*

Tieto triedy dedia od spoločného predka – triedy *objekt*.

Cesty

Obsahuje jedinú triedu *cesta*. Táto trieda implementuje vodivú cestu slúžiacu na spojenie dvoch vývodov súčiastok.

Cleny

Tento balík obsahuje triedy reprezentujúce súčiastky logických členov použitých v tejto aplikácii. Názvy tried zodpovedajú názvom logických členov. V tomto balíku sa nachádzajú aj dve triedy, ktoré priamo nereprezentujú niektorú zo súčiastok, ale slúžia na zachytenie niektorých spoločných vlastností súčiastok. Ostatné triedy potom tieto vlastnosti od týchto dedia. Sú to triedy *clen* – táto trieda dedí od triedy *objekt* a od nej dedia triedy reprezentujúce súčiastky logických členov, ako aj trieda *PO*. Táto trieda je zároveň druhou spomínanou. Od triedy *PO* dedia triedy reprezentujúce preklápacie obvody.

Dialogy

V tomto balíku sú obsiahnuté triedy reprezentujúce dialógy v programe (vyskakujúce okná), ktoré slúžia na vstup od používateľa, resp. na zobrazenie určitého výstupu. Sú tu triedy ako *Autor* (zobrazuje informácie o autorovi aplikácie), *Help* (zobrazuje stručný návod na použitie), *VyberPomeru* (slúži na to, aby si používateľ mohol zadať počet jednotlivých súčiastok v danom paneli) *Mapa* (zobrazuje Karnaugh-ovu mapu), *Tabulka* (zobrazí pravdivostnú tabuľku), a pod..

Predkovia

Obsahuje už spomínanú triedu *objekt*, od ktorej dedia ďalšie triedy. Obsahuje informácie (vlastnosti) spoločné pre všetky objekty aplikácie.

Balíky obrázkov

Tieto balíky, ako už bolo spomínané, obsahujú obrázky použité na zobrazenie jednotlivých súčiastok. V týchto balíkoch sa nenachádzajú žiadne triedy, slúžia len na oddelenie obrázkov, aby bolo možné použiť rovnaké názvy. Podľa výberu dizajnu sa potom používa rozdielny balík obrázkov.

5.2 Implementovaná funkcionalita

Aplikácia umožňuje verifikovať používateľom vybraný panel úplnej množiny súčiastok. Počet jednotlivých typov súčiastok, ako aj počet vstupov pre jednotlivé súčiastky sa načítava z konfiguračného súboru, ktorý môže byť používateľom modifikovaný (len predpísaným spôsobom). Pokiaľ konfiguračný súbor nie je nájdený, tak je používateľ vyzvaný zadať počet typov súčiastok v dialógovom okne. Počet vstupov je prednastavený a je ho v prípade absencie konfiguračného súboru možné meniť dvojklikom na súčiastku.

Celkový počet súčiastok logických členov (spolu s preklápacími obvodmi) je konštantný a jednotlivé súčiastky nie je možné pridávať, alebo odstrániť z panela. Panel vždy obsahuje 4 prepínače (generátory) logických hodnôt označené X, Y, Z, W a 4 zobrazovače logických hodnôt (žiarovky). Súčiastky sú v paneli pevne

umiestnené a nie je možné ich presúvať. Taktiež nie je možné modifikovať ich funkcionalitu (s výnimkou zmeny počtu vstupov vo vyššie uvedenom prípade). Súčiastky sú umiestnené tak, aby sa medzi ne zmestil dostatočný počet vodičov.

Vodiče sú lomené čiary, ktoré spájajú jednotlivé vývody súčiastok. Každé takéto prepojenie má vyhradenú cestu, čiže sa nemôže stať, aby sa vodiče prekrývali. Umiestnenie vodičov sa tiež nedá dodatočne modifikovať, len pridať vodič a vymazať ho.

Menu bolo oproti návrhu trochu zmenené. V podmenu *Súbor* sa nachádzajú nasledujúce položky:

- *Nový* – otvorí dialógové okno na výber typu panela a následne ho zobrazí
- *Otvoriť zapojenie* – spustí okno na výber súboru, do ktorého bolo predtým zapojenie uložené
- *Uložiť zapojenie* – spustí okno na výber súboru, do ktorého má byť zapojenie uložené
- *Koniec* – ukončí činnosť programu

Podmenu *Nastavenia* obsahuje tieto položky:

- *Zobraz vstupné hodnoty* – nad každým vstupom súčiastky zobrazí logickú hodnotu
- *Zobraz výstupné hodnoty* – nad výstupom súčiastky zobrazí logickú hodnotu
- *Zobraz funkciu výstupu* – zobrazí funkciu výstupu, táto funkcia sa vypočíta zo vstupných premenných a funkcionality súčiastky
- *Zjednodušovať funkciu výstupu* – výstupnú funkciu zjednoduší, napr. namiesto X + 0 zobrazí X
- *Farby podľa hodnôt* – zobrazuje farby vodičov v závislosti od logickej hodnoty, ktorú prenášajú
- *Logický zisk súčiastok* – nastavuje počet súčiastok, koľko môže byť pripojených na výstup jednej súčiastky; po prekročení logického zisku aplikácia pracuje normálne ďalej, ale upozorňuje chybovým výpisom, že bol tento zisk prekročený, a teda, že v skutočnosti by takýto panel už nemusel fungovať korektne.
- *Neošetrený výstup sa správa ako* – voľba logickej 1, alebo stavu vysokej impedancie Z
- *Voľba dizajnu* – umožňuje vybrať dizajn aplikácie, teda akým spôsobom sa panel vykresľuje; na výber sú 4 dizajny (čierno-biely, bielo-sivý, zeleno-čierny a americká norma zobrazenia súčiastok)

V podmenu *Zapojenie* sú zahrnuté tieto položky:

- *Export do VHDL* – transformuje zapojenie do zdrojového kódu v jazyku VHDL a uloží do súboru „export.vhdl“. Vo VHDL predstavuje panel akoby jednu veľkú súčiastku, ktorej štruktúra je opísaná vytvoreným zapojením. Čiže sa do VHDL zapíšu len použité súčiastky. Tento celkový obvod má

štvorbitový vstupný vektor reprezentovaný prepínačmi X – Z a štvorbitový výstupný vektor reprezentovaný žiarovkami.

- *Zobraz pravdivostnú tabuľku* – zobrazí okno s pravdivostnou tabuľkou výstupných funkcií Z1 – Z4 reprezentovanými vstupnými funkciami žiaroviek.
- *Zobraz Karnaughovu mapu* – zobrazí okno so 4 Karnaugh-ovými mapami pre každú výstupnú funkciu Z1 – Z4.
- *Uložiť ako obrázok*... – umožňuje exportovať zapojenie do formátu obrázka, na výber sú formáty „JPG“, „BMP“, „PNG“ a „GIF“.

V poslednom podmenu *O programe* sa nachádzajú položky *Autor* (zobrazí okno s informáciami o autorovi) a *Help* (zobrazí stručnú pomoc pri tvorbe zapojenia).

Väčšina položiek v menu má k dispozícii aj klávesovú skratku, na ktorú reaguje, čo umožňuje používateľovi pohodlnejšiu a rýchlejšiu prácu s nástrojom.

Export do VHDL

Najprv sa do súboru zapíše použitie knižnice IEEE a jej balíka std_logic_1164. Potom postupne prechádzame všetky súčiastky a pre každý nový typ použitej súčiastky zapíšeme novú entitu do súboru s názvom „Obvod_*názov_súčiastky*“. Tieto súčiastky opíšeme vo VHDL správaním podľa funkcionality, ktorú daná súčiastka má. Po definovaní potrebných súčiastok zapíšeme do súboru entitu celkového obvodu s názvom „Obvod“. Pre všetky použité typy súčiastok zapíšeme komponenty do tejto entity. Následne zistíme počet potrebných signálov (vnútorných prepojení – prepojení medzi logickými členmi) a zadefinujeme. Nakoniec pre všetky použité súčiastky zapíšeme do súboru inštancie ich entít a namapujeme porty. Takýmto spôsobom realizujeme exportovanie vytvoreného zapojenia do opisu prostredníctvom jazyka VHDL.

5.3 Implementácia pracovného dizajnu

Súčasťou funkcie aplikácie je aj možnosť voľby pracovného dizajnu. Celkovo je možné vyberať z nasledovných prevedení :

- pôvodný
- jednoduchý bielo-sivý
- zeleno-šedý
- americký

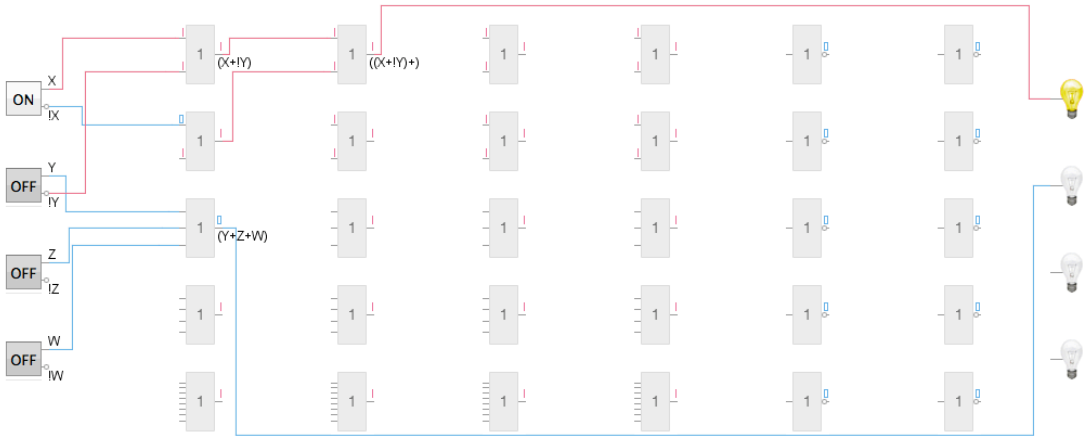
Posledný menovaný grafický návrh reprezentuje súčiastky podľa americkej normy. Ostatné návrhy

predstavujú schematické značky v "hranatej" forme. Každý grafický návrh obsahuje obrázok vo formáte gif, pre každý typ súčiastky. Taktiež je k jednotlivým grafickým návrhom priradená príslušná farba cesty a pozadia. Nakoľko je možné zvoliť, či farba cesty záleží od hodnoty, ktorá na nej je, má každý grafický návrh určené tri farby ciest :

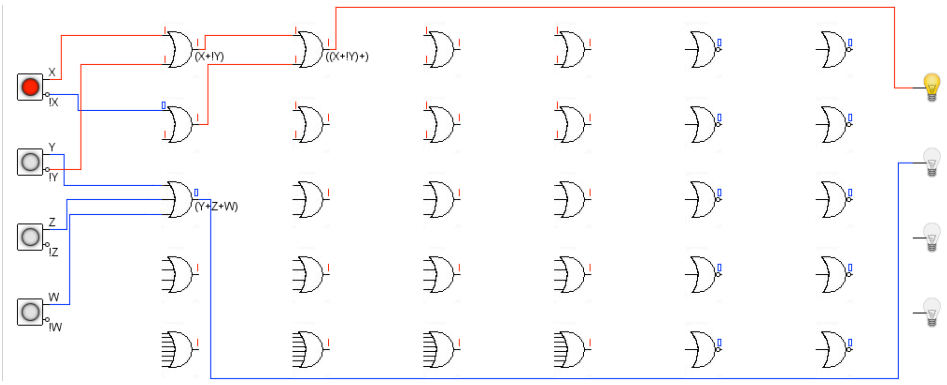
- neutrálna - na hodnote, ktorá je na ceste nezáleží
- 1 - cesta vedie hodnotu log. 1
- 0 - cesta vedie hodnotu log. 0

Spôsob zobrazovania ciest je možné zvoliť z menu aplikácie. Pracovný dizajn je možné meniť počas behu programu, aj keď je už vytvorené zapojenie.

Na obrázkach č. 32 a č. 33 je zobrazený jednoduchý bielo-sivý dizajn a dizajn predstavujúci americkú normu.



Obr. 38 Jednoduchý bielo-sivý dizajn



Obr. 39 Americká norma

6. Testovanie softvérového produktu

6.1 Akceptačné testy

V nasledujúcej sérii testov overíme funkčnosť a použiteľnosť nášho softvérového produktu pri výučbovom procese.

1. Testovanie funkčnosti Univerzálneho virtuálneho verifikačného panelu logických obvodov na vzorovom zadaní z cvičení na predmete Logické obvody

Zadanie: Navrhnite prevodník číslic 0-9 v kóde BCD8421 do kódu BCD8421+3. Prevodník realizujte s minimálnym počtom členov NAND. Kódy sú zobrazené na obrázku č. 40.

A = (b&c&!d) | (b&d) | (!a)
B = (b&!c&!d) | (!b&c&!d) | (!b&d)
C = (c&d) | (!c&!d)
D = !d

	BCD8421				BCD8421+3			
	a	b	c	d	A	B	C	D
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Obr. 40 Zakódované číslice v kóde BCD8421 a BCD8421+3

V tabuľke č. 13 je zobrazený test softvérového produktu na vzorovom zadaní.

ID	1	**Názov**	Test softvérového produktu na vzorovom zadaní
Rozhranie		Aplikácia / Používateľ	
Účel		Overenie správnej funkčnosti logických členov, ich zapájania a kontrola výslednej pravdivostnej tabuľky	
Vstupné podmienky		Správny návrh logického obvodu	
Výstupné podmienky		Hodnoty na výstupe sa prepínajú na základe vstupných hodnôt podľa uvedenej prevodovej tabuľky	

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Výber verifikačného panelu z log. členov NAND	Vytvorenie panelu, ktorý obsahuje log. členy NAND s rôznym počtom vstupov	**√**
2	Správne zapojenie jednotlivých log. členov podľa návrhu	Zapájanie log. členov je funkčné	**√**
3	Označenie cesty kliknutím ľavým tlačidlom myši	Označená cesta ľavým tlačidlom zmení farbu	**√**
4	Vymazanie cesty kliknutím pravým tlačidlom myši	Označená cesta pravým tlačidlom je vymazaná	**√**
5	Prepínanie vstupov obvodu.	Na výstupe sledujeme hodnoty podľa uvedenej prevodovej tabuľky.	**√**
6	Zapojenie → Zobraz pravdivostnú tabuľku	Zobrazí sa pravdivostná tabuľka, ktorá bude zodpovedať uvedenej prevodovej tabuľke.	**√**

Tab. 13 Test softvérového produktu na vzorovom zadaní

2. Testovanie funkčnosti uloženia vytvoreného zapojenia logických členov s vybranými nastaveniami a jeho opätovného otvorenia je zobrazené v tabuľke č. 14.

ID	2	**Názov**	Test uloženia a otvorenia vytvoreného zapojenia s vybranými nastaveniami
Rozhranie		Aplikácia / Používateľ	
Účel		Overenie správnej funkčnosti ukladania a otvárania zapojení aj so zmenami dizajnu alebo zobrazovaných funkciách	
Vstupné podmienky		Vytvorené zapojenie logických členov	
Výstupné podmienky		Otvorené zapojenie bude vizuálne a funkčne zodpovedať ukladanému zapojeniu	

Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Nastavenia → Nezobrazovať vstupné hodnoty	Na paneli sa nebudú zobrazovať logické hodnoty na vstupe členov	**√**
2	Nastavenia → Nezobrazovať výstupné hodnoty	Na paneli sa nebudú zobrazovať logické hodnoty na výstupe členov	**√**
3	Nastavenia → Nezobrazovať funkciu výstupu	Na paneli sa nebudú zobrazovať funkcie výstupu logických členov	**√**
4	Nastavenia → Farby podľa hodnôt	Cesty spájajúce logické členy zmenia farbu podľa logickej hodnoty	**√**
5	Súbor → Uložiť zapojenie	Otvorenie okna pre výber miesta uloženia zapojenia	**√**
6	Uloženie zapojenia a zatvorenie programu		**√**
7	Súbor → Otvoriť zapojenie	Otvorí sa funkčné uložené zapojenie so všetkými vybranými nastaveniami	**√**

Tab. 14 Test uloženia a otvorenia vytvoreného zapojenia s vybranými nastaveniami

3. Testovanie funkčnosti a správnosti generovania kódu VHDL sa nachádza v tabuľke č. 15.

ID	3	**Názov**	Test funkčnosti exportu do VHDL	
Rozhranie		Aplikácia / Používateľ		
Účel		Overenie funkčnosti exportu do VHDL a overenie správnosti vygenerovaného VHDL kódu		
Vstupné podmienky		Vytvorené zapojenie logických členov		
Výstupné podmienky		Kód VHDL úspešne prešiel kompilátorom v ModelSim		
Krok	**Akcia**		**Očakávaná reakcia**	**Skutočná reakcia**
1	Zapojenie → Export do VHDL		V priečinku so spúšťaným programom sa vytvorí textový súbor *export.vhdl*	**√**
2	Kontrola vytvoreného súboru *export.vhdl*		V priečinku so spúšťaným programom bol vytvorený súbor *export.vhld*	**√**
3	Kompilácia kódu VHDL v ModelSim		Kód VHDL úspešne prešiel kompilátorom v ModelSim	**√**

Tab. 15 Test funkčnosti exportu do VHDL

4. Testovanie funkčnosti exportu zapojenia do obrázku a zmeny dizajnu je zobrazené v tabuľke č. 16.

ID	4	**Názov**	Test funkčnosti zmeny dizajnu a exportu do obrázkov	
Rozhranie		Aplikácia / Používateľ		
Účel		Overenie funkčnosti exportu do VHDL a overenie správnosti vygenerovaného VHDL kódu		
Vstupné podmienky		Vytvorené zapojenie logických členov		
Výstupné podmienky		Výsledný obvod exportovaný do formátov - .jpg, .gif, .png, .bmp		
Krok	**Akcia**		**Očakávaná reakcia**	**Skutočná reakcia**
1	Nastavenie → Voľba dizajnu		Zmení sa vizuálna stránka verifikačného panelu	**√**
2	Zapojenie → Uložiť ako obrázok		Používateľ dostane na výber z rôznych formátov	**√**
3	Používateľ si vyberie formát		Používateľ dostane možnosť skryť všetky nepoužité členy verifikačného panelu	**√**
4	Používateľ nájde exportovaný obrázok v priečinku s programom			**√**

Tab. 16 Test funkčnosti zmeny dizajnu a exportu do obrázkov

5. Testovanie funkčnosti a správnosti klávesových skratiek je zobrazené v tabuľke č. 17.

ID	5	**Názov**	Test funkčnosti klávesových skratiek
Rozhranie		Aplikácia / Používateľ	
Účel		Overenie funkčnosti exportu do VHDL a overenie správnosti vygenerovaného VHDL kódu	
Vstupné podmienky		Vytvorené zapojenie logických členov	
Výstupné podmienky			

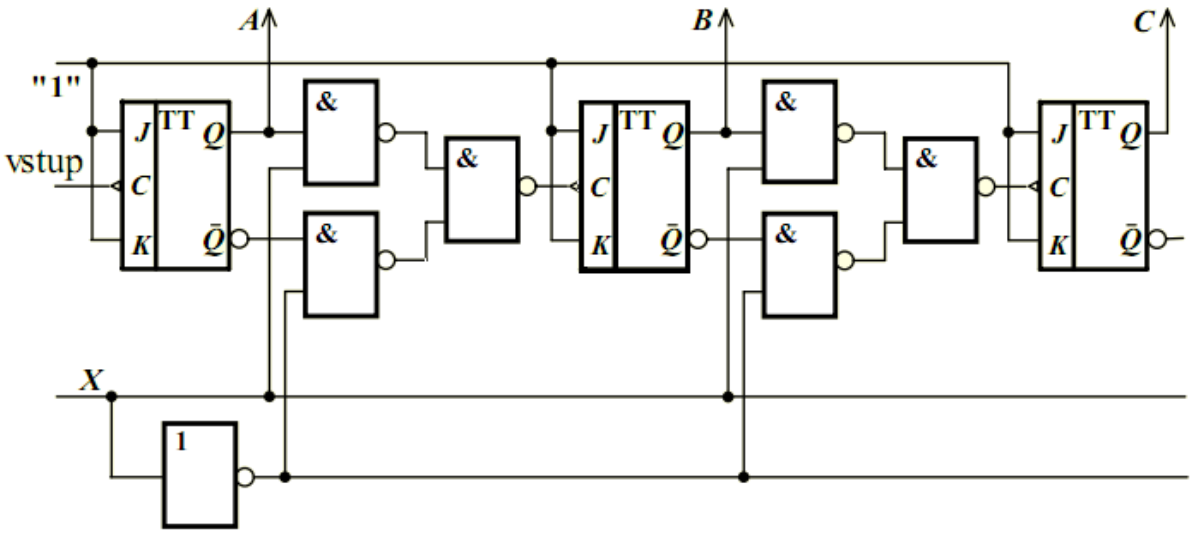
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Ctrl + N	Otvorenie nového súboru	**√**
2	Ctrl + O	Otvorenie uloženého zapojenia	**√**
3	Ctrl + S	Uloženie vytvoreného zapojenia	**√**
4	Ctrl + K	Ukončenie programu	**√**
5	Ctrl + L	Nastavenie logického zisku súčiastok	**√**
6	Ctrl + V	Export do VHDL	**√**
7	Ctrl + T	Zobrazenie pravdivostnej tabuľky	**√**
8	Ctrl + M	Zobrazenie Karnaughovej mapy	**√**
9	Ctrl + A	Informácie o autoroch	**√**
10	Ctrl + H	Help	**√**
11	Ctrl + J	Export do obrázku vo formáte .jpg	**√**
12	Ctrl + P	Export do obrázku vo formáte .png	**√**
13	Ctrl + B	Export do obrázku vo formáte .bmp	**√**
14	Ctrl + G	Export do obrázku vo formáte .gif	**√**

Tab. 17 Test funkčnosti klávesových skratiek

6. Testovanie funkčnosti Univerzálneho virtuálneho verifikačného panelu logických obvodov na vzorovom zadaní, ktoré je zhrnuté v tabuľke č. 18.

Zadanie: Navrhnite odrátavajúci obvod od 7 do 0 pomocou preklápacích obvodov J-K

Návrh: Na obr. č. 41 je možné vidieť návrh odrátavajúceho obvodu od 7 do 0 pomocou preklápacích obvodov J-K.



Obr. 41 Návrh odrátavajúceho obvodu

ID	6	**Názov**	Test softvérového produktu na vzorovom zadaní
Rozhranie		Aplikácia / Používateľ	
Účel		Overenie správnej funkčnosti logických členov a preklápacích obvodov J-K	
Vstupné podmienky		Správny návrh logického obvodu	
Výstupné podmienky		Hodnoty na výstupe sa zostupne menia s prepínaním hodinového signálu	

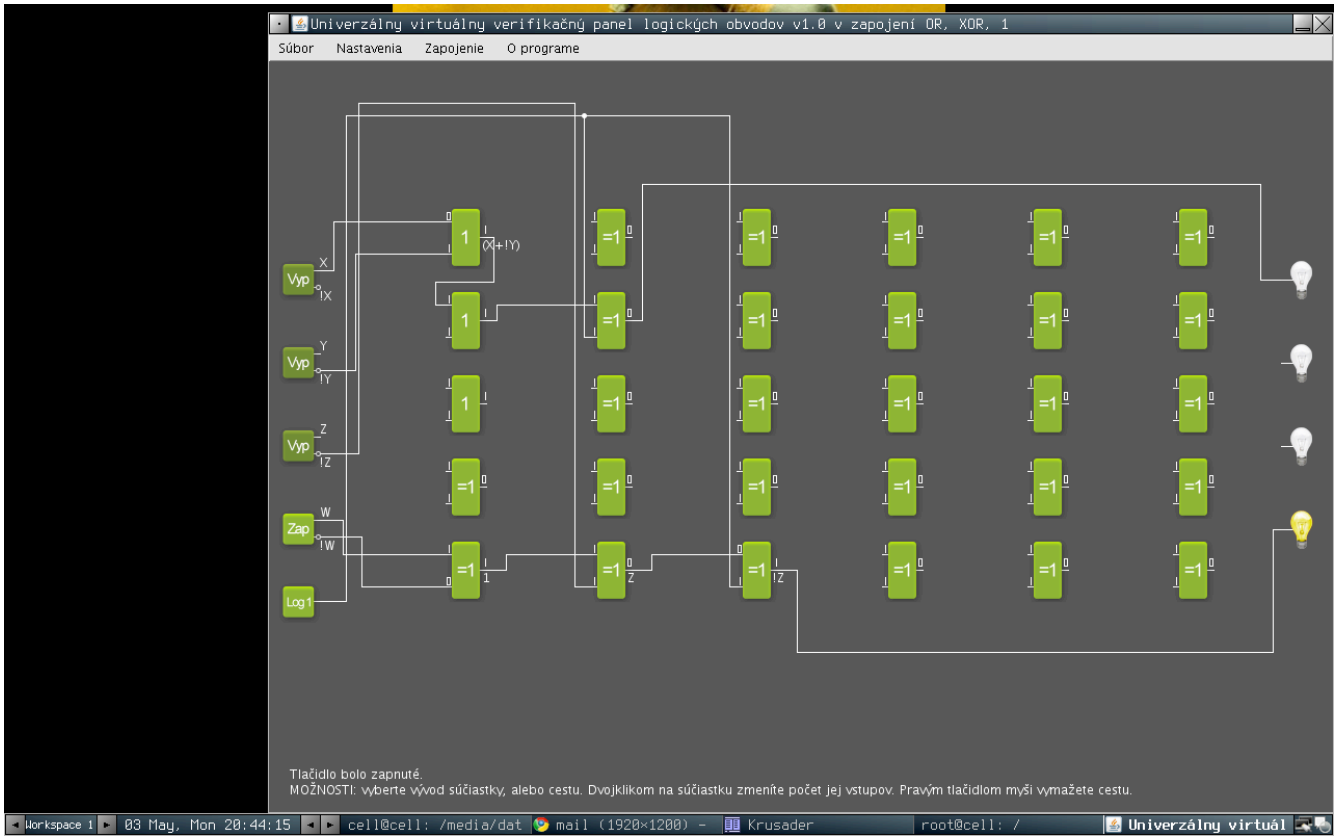
Krok	**Akcia**	**Očakávaná reakcia**	**Skutočná reakcia**
1	Výber verifikačného panelu zloženého z preklápacích obvodov J-K a log. členov NAND	Vytvorenie panelu, ktorý obsahuje log. členy NAND s rôznym počtom vstupov a preklápacie obvody J-K	**√**
2	Správne zapojenie jednotlivých log. členov podľa návrhu	Zapájanie log. členov je funkčné	**√**
3	Prepínanie hodinového signálu	Hodnoty na výstupe sa zostupne menia s prepínaním hodinového signálu	**√**

Tab. 18 Test softvérového produktu na vzorovom zadaní

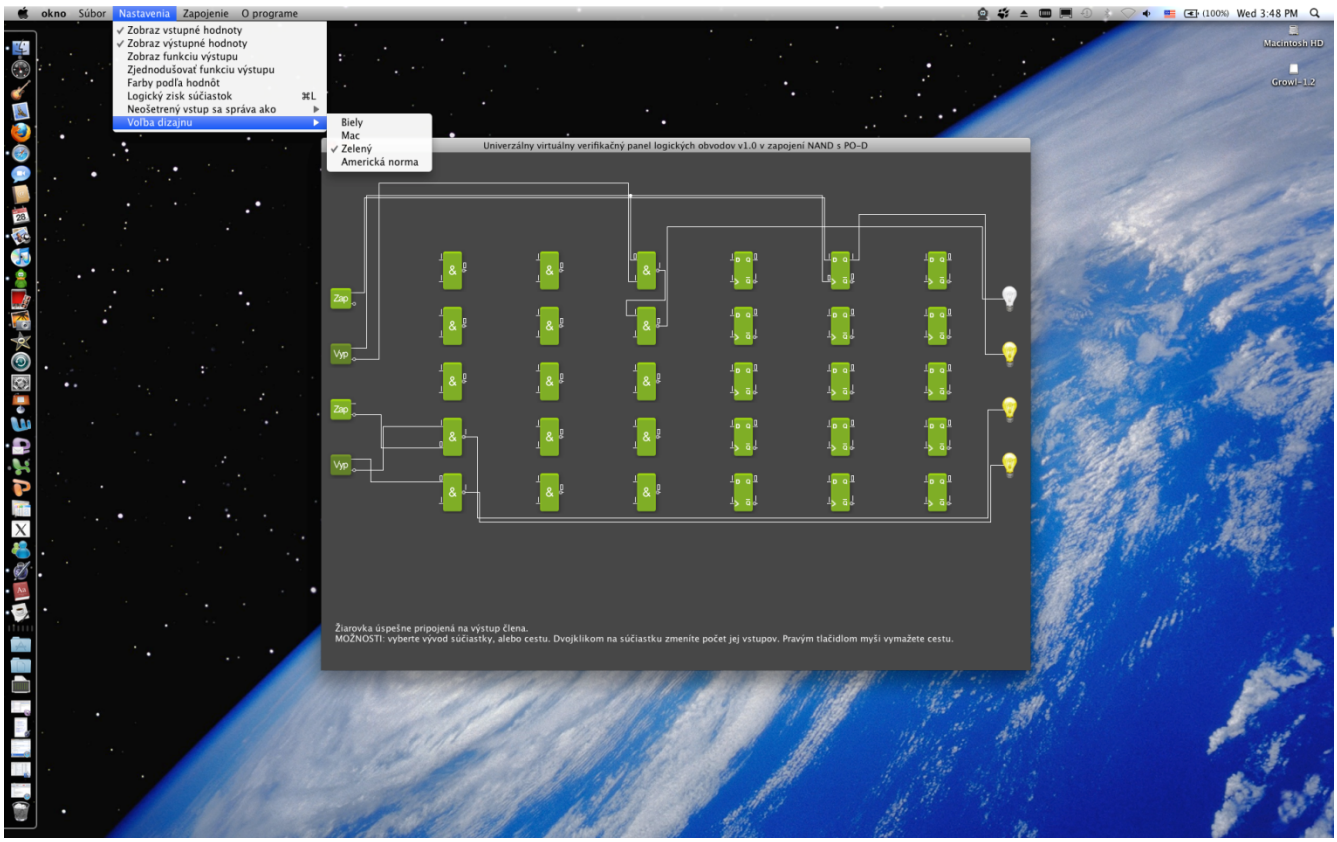
6.2 Testovanie v rôznych podmienkach

Okrem testovania aplikácie opísanej v predchádzajúcej kapitole (6.1) boli urobené aj iné testovania. Aplikácia bola odskúšaná na viacerých nasledujúcich operačných systémoch:

- Windows XP
- Windows 7
- Debian GNU/Linux 5.0, jadro 2.6.30 (obr. č. 42)
- MacOS X so systémom Snow Leopard (obr. č. 43)



Obr. 42 Screenshot aplikácie z OS Linux



Obr. 43 Screenshot aplikácie z OS Mac

Na všetkých vyššie spomenutých operačných systémoch aplikácia bežala bez problémov, prípadne bez žiadnych neočakávaných ukončení aplikácie. Jediné čo bolo potrebné je, aby na všetkých operačných systémoch bola nainštalovaný program Java Runtime Environment (JRE), ktorý v sebe obsahuje Java Virtual Machine (JVM), ktorá je potrebná pre spustenie súboru s príponou *.jar.

Okrem operačných systémov bolo testovanie uskutočnené aj na aktuálnych študentoch predmetu Logické obvody, ktorý mali možnosť priamo na cvičení odskúšať vytvorenú aplikáciu. Po odskúšaní boli študenti poprosení o vyplnenie dotazníkov ohľadom testovanej aplikácie. Vyplnené dotazníky obsahujúce ich spomenuté nedostatky aplikácie a ich návrhy na zlepšenie sú priložené v dokumentácii riadenia v prílohe A. Všetky chyby a nedostatky aplikácie, ktoré študenti našli boli úspešne vyriešené a zapracované do aplikácie.

7. Záver

Cieľom počas týchto dvoch semestrov bolo vytvoriť aplikáciu na tému Univerzálny virtuálny verifikačný panel logických obvodov. Našim zámerom bolo spraviť čo najlepšiu, ľahko ovládateľnú aplikáciu s potrebnými funkciami a potrebnou logikou tak, aby ju bolo možné nasadiť a používať ako školskú pomôcku určenú primárne pre študentov predmetu Logické obvody.

Najprv sme podrobne roznalyzovali danú problematiku do potrebnej hĺbky aj s analýzou už existujúcich podobných riešení. Následne na to sme spravili potrebný návrh a všetky požiadavky, ktoré sme špecifikovali sme implementovali v maximálnej možnej miere. Všetky ciele, ktoré sme si vytýčili, aby aplikácia spĺňala sa nám podarilo naimplementovať a dôsledne otestovať.

Vychádzali sme s už vytvorenej aplikácie člena tímu Miroslava Sieberta, ktorý ju vytvoril ako výstup bakalárskej práce. Na tomto základe sme začali stavať ďalšiu potrebnú funkcionalitu. Aplikácia ponúka na výber všetky úplne súbory logických členov, multiplexory a preklápacie obvody, pomocou ktorých je možné vyskladať všetky druhy obvodov, ktoré daný užívateľ potrebuje spraviť. Vyskladané obvody si používateľ môže následne aj skontrolovať pomocou funkcií zobrazenia Karnaughových máp a pravdivostných tabuliek aktuálne vytvorených obvodov s jeho Karnaughovými mapami a pravdivostnými tabuľkami. Pre pokročilejších používateľov aplikácia poskytuje aj funkciu exportu vytvoreného zapojenia do VHDL kódu, ktorý sa môže hodiť pre študentov vyšších ročníkov napríklad na predmete Špecifikačné a opisné jazyky.

Náš tím si je vedomý, že aplikácia nie je perfektná, keďže stále je tam čo zlepšovať a je tam miesto pre implementáciu ďalších pokročilejších funkcií. Veríme, že sme za tieto dva semestre spravili všetko čo bolo v našich silách, aby sme vytvorili aplikáciu, ktorá nebude len obyčajným zadaním, ktoré bolo vytvorené presne podľa zadania, ale že sme urobili aj funkcie a celkovo aplikáciu, ktorá presahuje stanovené hranice zadania.

Počas práce na tomto zadaní si všetci členovia tímu osvojili nielen základné postupy práce v tíme ale aj hlbšie poznatky z danej témy. Spoločne sa nám podarilo vytvoriť aplikáciu, ktorá má veľmi dobré predpoklady pre jej reálne nasadanie a používanie v školskom prostredí na cvičeniach z predmetu Logické obvody ale aj pre osobné účely.

8. Použitá literatúra

[1] FRIŠTACKÝ, N., KOLESÁR, M., KOLENIČKA, J., HLAVATÝ, J.: Logické systémy. 1. vyd. Bratislava : Alfa, 1986. 592 s.

[2] ŽATKOVIČ, A.: Logické obvody [online] [cit 2008-11-15]. Dostupné na Internete: http://alzat.szm.sk/Ttl_mos/Log_obv/log_obv.htm

[3] KAPRÁLIK, P., GALANOVÁ, J., POLAKOVIČ, M.: Logické systémy. 1. vyd. STU Bratislava, 2007. 168s. ISBN 978-80-227-2589-7

[4] HEROUT, P.: Učebnice jazyka Java. 2. vyd. České Budějovice : KOPP, 2006. 343 s. ISBN 80-7232-115-3

[5] HEROUT P.: Java – grafické uživatelské prostředí a čeština. 1. vyd. České Budějovice : KOPP, 2004. 316 s. ISBN 80-7232-237-0

[6] University of Genoa, Italy: Stránka projektu Deeds [online][cit 2008-11-23]. Dostupné na Internete: www.esng.dibe.unige.it/netpro/Deeds/Index.htm

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

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

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

[10] FRIŠTACKÝ, N., KOLENIČKA, J., KOLESÁR, M.: Logické systémy: Kombinačné obvody. Bratislava: ŠVŠT, 1986.

[11] Logic Gate Simulator. Dostupné na Internete: http://richardbowles.tripod.com/dig_elec/tools/sim/sim.htm

[12] Logicly. Dostupné na Internete: http://joshblog.net/projects/logic-gate-simulator/Logicly.html

[13] LogicSim. Dostupné na Internete: http://www.tetzl.de/logicsim_applet.html

[14] Simcir. Dostupné na Internete: http://www.d-project.com/simcir/simcir.html

[15] The Logic Lab. Dostupné na Internete: http://www.neuroproductions.be/logic-lab/index.php?id=52

[16] Probe. Dostupné na Internete: http://www.scit.wlv.ac.uk/~cm1970/probe/webpage/probeapp.html

[17] LOGiX. Dostupné na Internete: http://www.simtel.net/product.php%5Bid%5D90288%5Bcid%5D86%5BSiteID%5Dsimtel.net

[18] Electronics Workbench Multisim. Dostupné na Internete: http://www.ni.com/multisim/

Príloha A

1. Používateľská príručka k prototypu

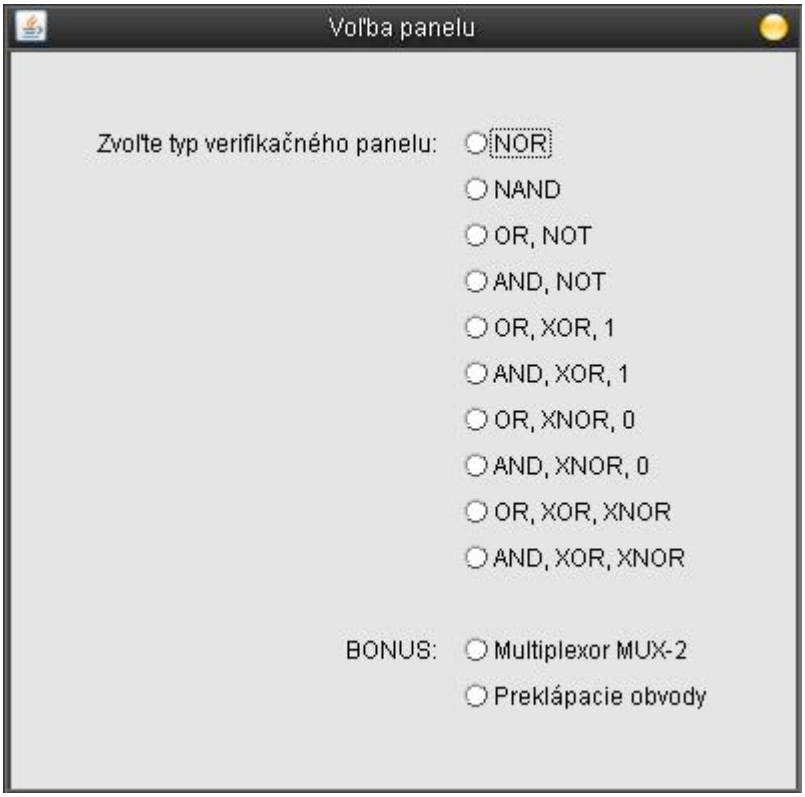
1.1 Inštalácia systému

Pre spustenie aplikácie je najprv potrebné nainštalovať prekladač jazyka java t.j. Java Runtime Environment (JRE aspoň verzie 5.0), ktorý je voľne dostupný na web stránke spoločnosti Sun www.java.com, prípadne možné nájsť po zadaní výrazu „download java“ do ľubovoľného internetového vyhľadávača. Následne je možné aplikáciu spustiť prostredníctvom súboru prototyp.jar.

Následne je aplikácia plne funkčná a pripravená na používanie a tvorbu schém zapojení. Po ukončení aplikácie je vygenerovaný súbor log.file uložený v mieste spustenia aplikácie obsahujúci pomocné výpisy pre prípad zlyhania aplikácie, aby bolo možné danú chybu simulovať a odstrániť ju.

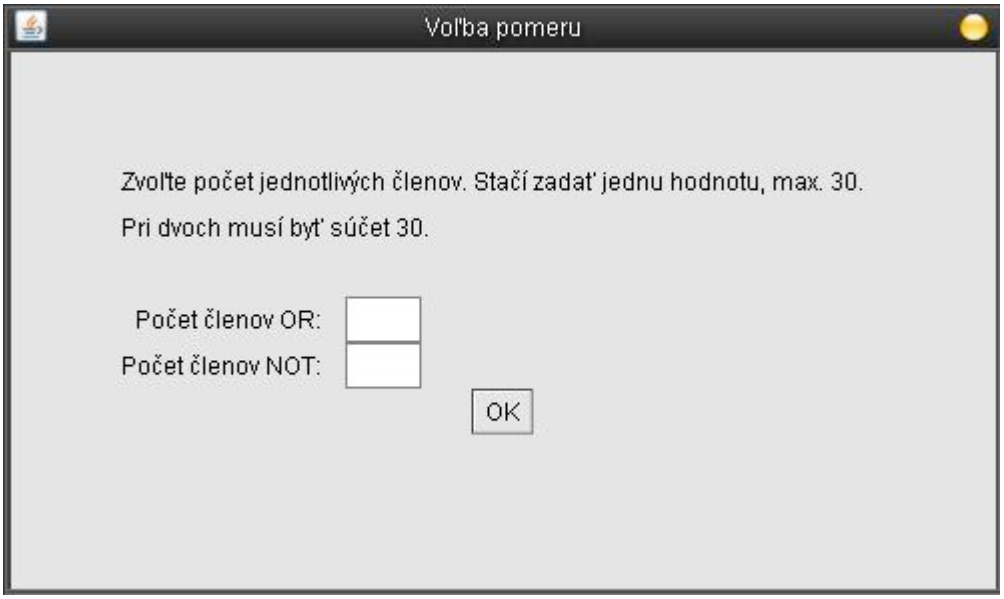
1.2 Používateľská príručka

Program samotný sa spúšťa spustiteľným súborom prototyp.jar. Po spustení sa používateľovi zobrazí okno (Obr. 44) Voľba panelu, kde si používateľ vyberie typ verifikačného panelu, ktorý požaduje.



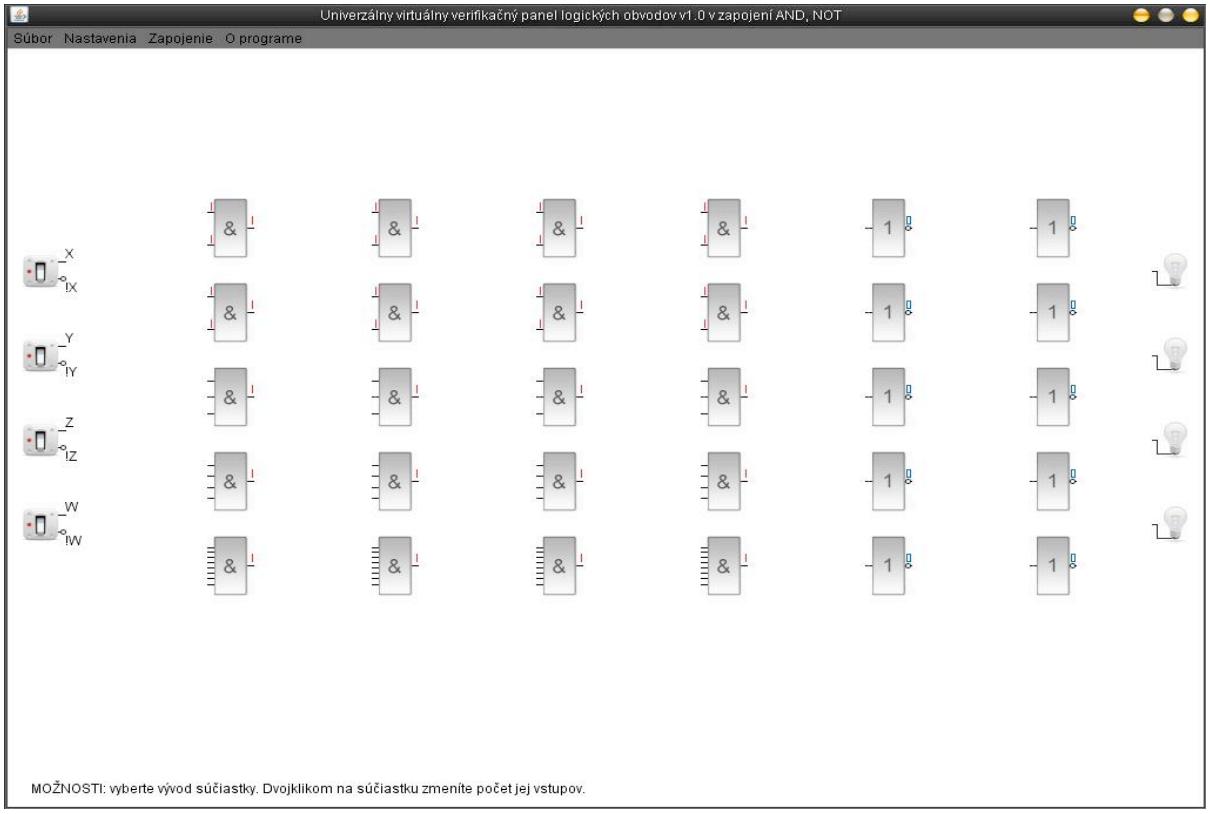
Obr. 44 Okno Voľba panelu

V prípade, že si používateľ vyberie verifikačný panel, ktorý obsahuje len jeden člen, tak sa mu zobrazí okno aplikácie, ktorého dominantným prvkom je kresliace plátno. Avšak ak je vybraný typ panelu, ktorý pozostáva z dvoch alebo troch členov, tak sa používateľovi zobrazí okno na Voľbu pomeru, ktoré je zobrazené na obr. 45.



Obr. 45 Okno Voľby pomeru

Používateľ si môže zvoliť počet jednotlivých členov, tak aby ich súčet bol vždy 30. Možné je zadať len počet pre jeden člen a počet pre druhý člen vypočíta program, tak aby ich súčet bol 30. Následne po potvrdení sa používateľovi zobrazí okno aplikácie, ktoré je zobrazené na obr. 46.



Obr. 46 Okno aplikácie

Všetky možné voľby dostupné pre používateľa v danom okamžiku behu programu sú vypisované v spodnom textovom riadku plátna. Pri editovaní zapojenia má používateľ nasledujúce možnosti:

- vytvorenie vodivej cesty medzi dvoma vývodmi súčiastok postupným kliknutím na tieto vývody v ľubovoľnom poradí
- pripojenie súčiastky na už existujúcu cestu postupným kliknutím na vývod člena a následne na pripájanú cestu
- vysvietenie cesty kliknutím na túto cestu
- kliknutím na spoločnú cestu (keď z jedného výstupu idú viaceré cesty) sa vysvietia všetky cesty z daného výstupu a ich cesta kam idú
- zmazanie vodivej cesty kliknutím pravým tlačidlom myši na túto cestu
- zmeniť stav tlačidla kliknutím na tlačidlo
- zmeniť počet vstupov súčiastky dvojitým kliknutím na súčiastku

Ovládanie nastavení aplikácie je možné pomocou menu programu umiestneného v hornej lište okna. Menu programu má nasledujúce voľby:

A. Súbor

a) Nový (Ctrl + N)

Otvorí nové plátno na kreslenie, pričom ak existujúce zapojenie nie je uložené, tak na to upozorní.

b) Otvoriť zapojenie (Ctrl + O)

Otvorí uložené zapojenie.

c) Uložiť zapojenie (Ctrl + S)

Uloží vytvorené zapojenie do súboru.

d) Koniec (Ctrl + K)

Ukončí program. V prípade, že nie je existujúce zapojenie uložené upozorní na to.

B. Nastavenia

a) Zobraz vstupné hodnoty

Pri každom vstupe logického člena vypíše aktuálnu logickú hodnotu.

b) Zobraz výstupné hodnoty

Pri každom výstupe logického člena vypíše jeho aktuálnu logickú hodnotu

c) Zobraz funkciu výstupu

Pri výstupe logického člena zobrazí aktuálnu funkciu, ktorú tvorí zapojenie po daný bod schémy.

d) Zjednodušovať funkciu výstupu

Namiesto aktuálnej funkcie zobrazí aktuálnu logickú hodnotu (0 alebo 1).

e) Farby podľa hodnôt

Povolí zobrazovanie farieb ciest podľa toho, aká logická hodnota vodičom prechádza. Pri zrušení tejto voľby budú všetky cesty vykreslené čiernou farbou.

f) Logický zisk súčiastok (Ctrl + L)

Zobrazí okno pre zadanie novej hodnoty logického zisku všetkých členov.

g) Neošetrený vstup sa správa ako

Umožňuje nastaviť neošetrený vstup, aby sa správal ako logická 1 alebo Z (stav vysokej impedancie).

h) Voľba dizajnu

Obsahuje v sebe podmenu, ktoré ponúka na výber 4 rôzne druhy dizajnu (čierno-biely, bielo-sivý, zeleno-čierny, americká norma), ktoré je možné si vybrať a použiť aj počas kreslenia. Výber iného dizajnu nevymaže nakreslené zapojenie.

C. Zapojenie

a) Krok späť (Ctrl + Q)

Funkcia, ktorá umožňuje používateľovi vrátiť urobený krok v zapojení späť. V prípade, že sa nedá ísť späť je táto položka menu znefunkčnená, ale je v menu stále zobrazená.

b) Krok vpred (Ctrl + W)

Funkcia, ktorá umožňuje používateľovi vrátiť krok, ktorý predtým odstránil funkciou krok späť do pôvodného stavu. V prípade, že sa nedá ísť vpred je táto položka menu znefunkčnená, ale je v menu stále zobrazená.

c) Export do VHDL (Ctrl + V)

Vytvorené zapojenie opíše pomocou VHDL kódu a uloží do súboru “export.vhdl”, ktorý je vytvorený v priečinku odkiaľ sa aplikácia spúšťa.

d) Zobraz pravdivostnú tabuľku (Ctrl + T)

Zobrazí okno s pravdivostnou tabuľkou výstupných funkcií Z1 – Z4 reprezentovanými vstupnými funkciami žiaroviek.

e) Zobraz Karnaughovu mapu (Ctrl + M)

Zobrazí okno so 4 Karnaugh-ovými mapami pre každú výstupnú funkciu Z1 – Z4.

f) Uložiť ako obrázok

Umožní vytvorené zapojenie vyexportovať ako obrázok. K dispozícii sú nasledovné formáty:

- .jpg (Ctrl + J)
- .bmp (Ctrl + B)
- .png (Ctrl + P)
- .gif (Ctrl + G)

D. O programe

Základné informácie o programe a help slúžiaci ako používateľská príručka.

a) Autor (Ctrl + A)

Zobrazí okno s informáciami o autorovi aplikácie.

b) Help (Ctrl + H)

Zobrazí jednoduchý návod, ktorý v krátkosti vysvetľuje ako vytvoriť a zmazať jednotlivé spojenia a tiež poskytuje krátke informácie o spojeniach a legendu farieb spojení.

Ukončenie programu je možné buď pomocou menu, alebo znakom X v pravom rohu okna. Pri zatváraní neuloženého zapojenia sa program opýta, či je používateľ naozaj rozhodnutý stratiť vytvorené zapojenie. Po skončení programu sa v adresári spustenia vygeneruje súbor log.file slúžiaci na kontrolu v prípade zlyhania behu programu.

2. Opis zdrojového kódu prototypu

Po spustení programu sa vykoná funkcia **public static void main(String[] args)** triedy **public class okno extends Frame**. Táto trieda je najdôležitejšia, pretože sa v jej konštruktore vykonávajú všetky najdôležitejšie inicializácie a obsluha funkcií programu (položiek menu).

Najprv sa objaví dialógové okno, v ktorom používateľ zvolí typ verifikačného panelu, ktorý chce použiť. Táto akcia sa vykoná vytvorením inštancie triedy **public class VyberPanelu extends Dialog implements**

ItemListener,ActionListener. V tomto dialógovom okne sa objaví 10 „radiobutton“ tlačidiel (súčasne iba jedno zvolené). Každý „radiobutton“ predstavuje inú súčiastkovú základňu verifikačného panelu a každému prislúcha aj index. Sú k dispozícii:

1. NOR
2. NAND
3. OR, NOT
4. AND, NOT
5. OR, XOR, 1
6. AND, XOR, 1
7. OR, XNOR, 0
8. AND, XNOR, 0
9. OR, XOR, XNOR
10. AND, XOR, XNOR

Pomocou funkcie **public int getVolba()** sa prenesie používateľom vybraná množina súčiastok panelu do hlavnej triedy (okno) a určí sa z nej typ verifikačného panelu. Celkový počet súčiastok v paneli je 30, a preto v prípade 2-súčiastkovej základne sa následne spustí dialógové okno triedy **public class VyberPomeru extends Dialog implements ActionListener**, kde používateľ vyberie počet súčiastok jedného a druhého typu tak, aby ich bolo spolu 30. Podobne pre 3-súčiastkovú základňu sa použije trieda **public class VyberPomeru2 extends Dialog implements ActionListener**. Podľa zisteného typu panelu sa v hlavnej triede rozmiestnia na plátno súčiastky použitím switch:

```
switch (pl.typPanelu){   //pl.typPanelu je pemenná, v ktorej je uložený index typu
case 1: //index pre typ panelu zo súčiastok NOR
        for(int a=0;a<30;a++){   //vytvorí sa 30 súčiastok
                pl.prvky[a] = new nor((170+(int)(a/5)*140),(140+(a%5)*80)); //vytvorí sa inštancia súčiastky nor na daných súradniciach
                pl.zapisDoSuboru("Zvoleny panel v zapojení NOR\n"); //do logu sa zapíše správa
        }break;
case 2:  //index pre typ panelu zo súčiastok NAND
        for(int a=0;a<30;a++){   //vytvorí sa 30 súčiastok
                pl.prvky[a] = new nand((170+(int)(a/5)*140),(140+(a%5)*80));  //vytvorí sa inštancia súčiastky nand na daných súradniciach
                pl.zapisDoSuboru("Zvoleny panel v zapojení NAND\n"); //do logu sa zapíše správa
```

}**break**;

... }

Plátno je oblasť, kde sa vykresľuje panel. Plátno je reprezentované triedou **public class Platno extends Canvas**.

Každý typ súčiastky je reprezentovaný osobitnou triedou. Napríklad súčiastka AND je reprezentovaná triedou **final public class and extends clen**. V tejto triede sa vykonáva logika výpočtu výstupnej hodnoty zo vstupných hodnôt. Ako je vidno, tieto súčiastky rozširujú spoločnú triedu **abstract public class clen extends objekt**, ktorá obsahuje informácie o vstupoch súčiastky. Táto trieda rozširuje triedu **abstract public class objekt implements Serializable**, v ktorej sú informácie o stave a umiestnení súčiastky. Túto triedu rozširujú aj súčiastky ako jednotka, nula, tlačidlo a žiarovka.