S T U
F I I T

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 812 19 Bratislava

Univerzálny virtuálny verifikačný panel logických obvodov

Dokumentácia projektu

(Tím č.6)

Bc. Peter Budinský
Bc. Roman Marcinčín
Bc. Tomáš Palaj
Bc. Juraj Štrba
Bc. Ľudovit Ubreži

Vedúci projektu: Ing. Peter Pištek
Študijný program: Počítačové a komunikačné systémy a siete
Štúdium: inžinierske
Ročník: 1.
Akademický rok: 2009/2010
Semester: zimný
Predmet: Tímový projekt I

Obsah

1 Úvod

Virtuálny verifikačný panel logických obvodov je aplikácia pre osobné počítače, ktorá poskytuje možnosti návrhu a overenia funkčnosti zostrojených logických obvodov. Virtuálny verifikačný panel je nástupcom hardvérových verifikačných panelov, pri ktorých sa navrhnute logické obvody realizovali na fyzickej úrovni. Takéto panely sa využívajú najmä na účely vzdelávania. Vzhľadom na neustály rast podielu moderných informačných technológií aj v oblasti školstva, naskytla sa myšlienka simulácie tohto hardvérového zariadenia na úrovni počítačového programu, čo má nesporné výhody najmä čo sa týka finančnej, ale ja funkčnej stránky. Účelom tohto dokumentu je cez analýzu problémovej oblasti a špecifikáciu požiadaviek dospieť k samotnému návrhu aplikácie, ktorá bude poskytovať funkčný nástroj na simuláciu správania sa navrhnutých logických obvodov

Výrazná časť tohto dokumentu je venovaná analýze oblasti logických funkcií a logických členov. V druhej kapitole sú popísané základné booleovské funkcie a spôsob ich implementácie pomocou logických členov. Rovnako sa tu nachádza prehľad už existujúcich programových riešení tohto typu, ktorý je zameraný najmä na vyzdvihnutie ich kladných stránok.

Tretia kapitola obsahuje špecifikáciu požiadaviek navrhovaného systému. Sú tu spísané požiadavky na funkčnosť ako i prípady použitia nášho programu.

V štvrtej kapitole sa zaoberáme predbežným návrhom programu. Okrem požiadaviek na hardvérové a softvérové vybavenie počítačov, na ktorých bude program prevádzkovaný, sa tu nachádza aj opis použitých technológií spolu so základným architektonickým modelom systému.

2 Analýza problémovej oblasti

2.1 Booleovské funkcie

Pre popis činnosti akéhokoľvek logického systému, v tomto prípade súčiastok virtuálneho verifikačného panela, sú potrebné funkcie, v ktorých hodnoty aj argumenty môžu nadobúdať iba konečný počet hodnôt. Takéto funkcie sú známe ako logické funkcie. Najčastejšie používané sú dvojhodnotové logické funkcie, ktoré sú známe ako **Booleovské funkcie** [2]**.** Pomenovanie dostali podľa známeho írskeho matematika 19. storočia George Boolea.

Booleovská funkcia (B-funkcia)

$$f(x_1, x_2, \ldots, x_n)$$

n premenných $x_1,x_2,...,x_n$ je zobrazenie

$$f : \{0,l\}^n \rightarrow \{0,1\}$$

Definičným oborom funkcie je množina všetkých n-tíc s prvkami 0 a 1 , ktorým zodpovedajú všetky možné usporiadané n-tice hodnôt premenných $x_1,x_2,...,x_3$. Z toho vyplýva, že definičný obor obsahuje práve 2^n rôznych n-tíc. Oborom hodnôt) funkcie je množina {0,l}. Booleovská funkcia priraďuje každej n-tici hodnôt premenných určitú hodnotu 0 alebo 1 . Pre n premenných je definovaných práve $2^{2\wedge n}$ B-funkcií.

2.1.1 Booleovské funkcie jednej premennej

Pre jednu vstupnú premennú x je možné definovať práve $2^{2\wedge 1} = 4$ B-funkcie. Ich prehľad je zobrazený v tabuľke (Tab. 1).

X	f_1	f_2	f_3	f_4
	0	x	$\overline{x}$	1
0	0	0	1	1
1	0	1	0	1

Tab. 1 B-funkcie jednej premennej

Funkcie f_1 a f_4 nadobúdajú konštantné hodnoty 0 resp. 1 nezávisle od premennej x, preto ich nazývame konštantami.

$$f_1 = 0$$
$$f_4 = 1$$

Funkcia f_2 má v každom bode rovnakú hodnotu ako je hodnota premennej x, opakuje jej hodnotu, preto sa nazýva priamou funkciou alebo opakovaním. Označujeme

$$f_2 = x$$

Funkcia f_3, naopak, nadobúda hodnotu opačnú ako je hodnota premennej x. Nazývame ju negáciou a označujeme

$$f_3 = \bar{x}$$

alebo

$$f_3 = !x$$

Funkcia negácie sa často nazýva slovom NOT.

2.1.2 Booleovské funkcie dvoch premenných

Pre dve vstupné premenné x_1, x_2 je možné definovať 16 ($2^{2\wedge2}$) B-funkcií. V nasledujúcej tabuľke (Tab. 2) je zobrazený ich stručný prehľad.

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
		0	.	↚	x_1	↦	x_2	⊕	+	↓	≡	$\bar{x}_2$	←	$\bar{x}_1$	→	↑	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Tab. 2 B-funkcie dvoch premenných

Ako je možné vidieť z tabuľky, funkcie f_1 a f_{16} sú funkcie konštantné, teda nezávislé na vstupných premenných. Platí

$$f_1 = 0$$
$$f_{16} = 1$$

Funkcie f_4, f_6, f_{11} a f_{13} nadobúdajú hodnotu v závislosti od jednej z vstupných premenných x_1 resp. x_2, pričom f_4 a f_6 sú priamou funkciou vstupnej premennej a teda

$$f_4 = x_1$$
$$f_6 = x_2$$

zatiaľ čo f_{11} a f_{13} sú negáciou jednej vstupnej premennej a platí

$$f_{11} = \overline{x}_2$$
$$f_{13} = \overline{x}_1$$

Pre všetky ostatné funkcie platí, že ich hodnota závisí od oboch vstupných premenných.

Funkcia f_2 sa nazýva logický súčin premenných x_1, x_2 alebo AND a nadobúda hodnotu 1 práve vtedy, keď obe vstupné premenné x_1, x_2 majú hodnotu 1. V ostatných prípadoch je hodnota logického súčinu 0. Operáciu logického súčinu označujeme

$$f_2 = x_1.x_2$$

V praxi sa často znak "." často vynecháva.

Funkcia f_8 sa nazýva logický súčet premenných x_1, x_2 alebo OR a nadobúda hodnotu 1 v prípadoch, keď aspoň jedna z premenných x_1, x_2 alebo obidve súčasne majú hodnotu 1. V prípade, keď premenné x_1, x_2 majú obidve hodnotu 0 je aj hodnota operácie logického súčtu 0. Operáciu logického súčtu označujeme

$$f_8 = x_1+x_2$$

Funkciu f_9 je negácia logického súčtu x_1, x_2 alebo tiež Pierceova funkcia a označujeme ju vertikálnou šípkou orientovanou nadol

$$f_9 = x_1 \downarrow x_2$$

Pierceova funkcia nadobúda hodnotu 1 práve vtedy, keď obidve premenné x_1, x_2 majú hodnotu 0. V ostatných prípadoch nadobúda Pierceova funkcia hodnotu 0. Pierceova funkcia

sa často nazýva NOR (NOT-OR), pretože ak je funkciu vyjadrená pomocou operácií logického súčtu a negácie, výsledkom je

$$f_9 = \overline{x_1 + x_2}$$

Funkciu f_{15} sa nazýva negácia logického súčinu x_1, x_2 alebo tiež Shefferova funkcia a označuje sa vertikálnou šípkou orientovanou nahor

$$f_{15} = x_1 \uparrow x_2$$

Shefferova funkcia nadobúda hodnotu 0 práve vtedy, keď obidve premenné x_1, x_2 majú hodnotu 1. V ostatných prípadoch má Shefferova funkcia hodnotu 1. Shefferova funkcia sa často nazýva NAND (NOT-AND), pretože ak je funkcia vyjadrená pomocou operácií logického súčinu a negácie, výsledkom je

$$f_{15} = \overline{x_1 . x_2}$$

Funkcia f_7 je takzvaná neekvivalencia x_1, x_2 alebo tiež súčet x_1, x_2 modulo 2. Často používaným je jej anglický názov XOR (eXclusive OR), čiže exkluzívny súčet x_1, x_2. Táto funkcia nadobúda hodnotu 1 práve vtedy, keď premenné nadobúdajú navzájom rozdielne hodnoty. naopak, ak majú premenné rovnaké hodnoty, nadobudne f_7 hodnotu 0. Neekvivalenciu značíme ako

$$f_7 = x_1 \oplus x_2$$

Funkcia f_{10} je opakom funkcie f_7 a nazýva sa ekvivalencia x_1, x_2. Hodnotu 1 nadobúda v prípade, že premenné x_1, x_2 majú rovnaké hodnoty. Ak sú hodnoty x_1, x_2 rozdielne, ekvivalencia má hodnotu 0. Označenie ekvivalencie je

$$f_{10} = x_1 \equiv x_2$$

Funkcie f_3 a f_5 sa nazývajú inhibícia. Funkcia f_5 nadobúda hodnotu premennej x_2 práve vtedy, keď hodnota premennej x_1 je rovná 0. Ak x_1 má hodnotu 1, funkcia f_5 nadobudne hodnotu 0. Túto funkciu možno slovne popísať ako „x_1 bráni x_2.“ Jednotková hodnota premennej x_1 zabraňuje prenosu hodnoty premennej x_2. Funkciu f_5 označíme

$$f_5 = x_1 \not\rightarrow x_2$$

a platí:

$$f_5 = x_2 \text{ pre } x_1 = 0$$
$$f_5 = 0 \text{ pre } x_1 = 1$$

Pre funkciu f_3 platia tie isté pravidlá ako pre f_5 s tým rozdielom, že poradie premenných je vymenené. Funkciu f_3 označíme

$$f_3 = x_2 \nrightarrow x_1$$

a platí:

$$f_3 = x_1 \text{ pre } x_2 = 0$$
$$f_3 = 0 \text{ pre } x_2 = 1$$

Funkcie f_{12} a f_{14} sa nazývajú implikácia. Funkcia f_{14} nadobúda hodnotu premennej x_2 práve vtedy, keď hodnota premennej x_1 je 1. Ak x_1 má hodnotu 0, funkcia f_{14} nadobudne hodnotu 1. Hovoríme, že „ x_1 implikuje x_2“ a označujeme

$$f_{14} = x_1 \rightarrow x_2$$

Platí:

$$f_{14} = x_2 \text{ pre } x_1 = 1$$
$$f_{14} = 1 \text{ pre } x_1 = 0$$

Pre funkciu f_{12} opäť platia tie isté pravidlá ako pre f_{14} s tým rozdielom, že poradie premenných je vymenené. Funkciu f_{12} označíme

$$f_{12} = x_2 \rightarrow x_1$$

a platí:

$$f_{12} = x_1 \text{ pre } x_2 = 1$$
$$f_{12} = 1 \text{ pre } x_2 = 0$$

2.1.3 Booleovské funkcie troch a viacerých premenných

Nakoľko počet B-funkcií pre n premenných je $2^{2\wedge n}$, popísanie všetkých B-funkcií pre tri a viac sa zdá byť nemožné. Ukázalo sa však, že všetky B-funkcie troch a viacerých premenných je možné popísať pomocou B-funkcií jednej a dvoch premenných pri vhodnom použití pravidiel Booleovskej algebry.

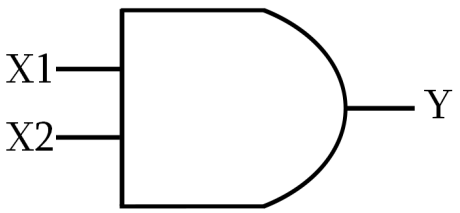
2.2 Logické členy

Logický člen (hradlo) [1] je elementárny prvok pre stavbu logických obvodov, ktorý realizuje požadovanú logickú (booleovskú) funkciu. Logické členy sú v súčasnosti najčastejšie realizované prostredníctvom tranzistorov uzatvorených v jednom puzdre integrovaného obvodu. Sú však známe aj iné možnosti realizácie logických členov, napríklad pomocou elektrónok, relé, rezistorov a diód či dokonca hydraulických alebo pneumatických ventilov. Z hľadiska návrhu a implementácie virtuálneho verifikačného panela však vnútorná štruktúra nie je dôležitá.

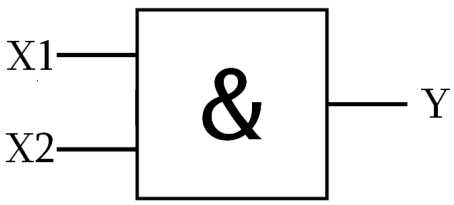
Každý logický člen je navrhnutý a prispôsobený na realizáciu práve jednej logickej funkcie. Na základe logickej funkcie, ktorú daný člen realizuje, dostal každý logický člen svoje pomenovanie. V nasledujúcom texte sú stručne popísané kombinačné logické členy AND, OR, XOR, NAND, NOR, XNOR, AND-NOR a NOT ktoré sú stavebnými prvkami navrhovaného virtuálneho verifikačného panela.

2.2.1 Logický člen AND

Logický člen AND sa používa na realizáciu operácie logického súčinu. Schematicky sa logický člen AND zobrazuje dvomi spôsobmi:



Obr. 1a Logický člen AND podľa štandardu IEC



Obr. 1b Logický člen AND podľa štandardu IEEE

Pravdivostná tabuľka pre operáciu logického súčinu realizovanú na dvojvstupovom logickom člene AND má tvar:

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Tab. 3 Pravdivostná tabuľka operácie logického súčinu

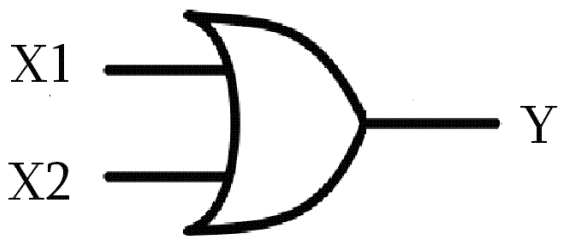
Pre vstupy X1, X2 a pre výstup Y platí:

$$Y = X1.X2$$

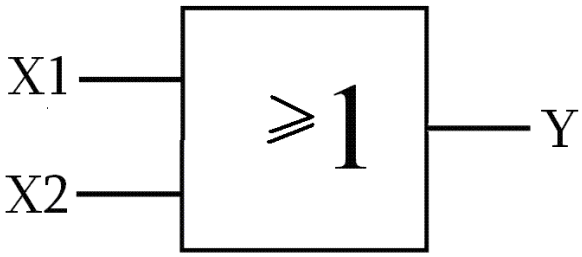
V praxi sa okrem dvojvstupových logických členov AND využívajú aj viacvstupové členy. Typicky známe sú trojvstupové, štvorstupové a osemvstupové členy AND. Výhodou použitia viacvstupových logických členov je možnosť realizovať daný obvod s menšími finančnými ako i priestorovými požiadavkami. Ak logický člen AND nevyužíva všetky svoje vstupy, na nevyužitých vstupoch je automaticky priradená logická hodnota 1, aby bol zaručený správny výstup logického súčinu.

2.2.2 Logický člen OR

Logický člen OR je implementáciou funkcie logického súčtu. Schematická značka logického člena OR vyzerá nasledovne::



Obr. 2a Logický člen OR podľa štandardu IEC



Obr. 2b Logický člen OR podľa štandardu IEEE

Pravdivostná tabuľka pre operáciu logického súčtu realizovanú na dvojvstupovom logickom člene OR má tvar:

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Tab. 4 Pravdivostná tabuľka operácie logického súčtu

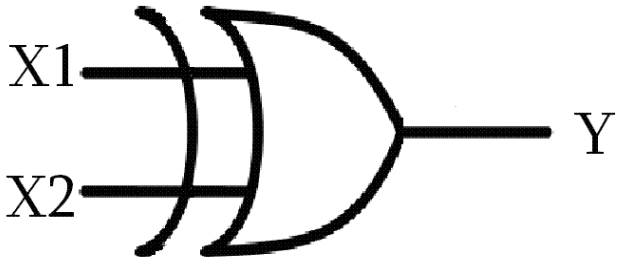
Pre vstupy X1, X2 a pre výstup Y platí:

$$Y = X1+X2$$

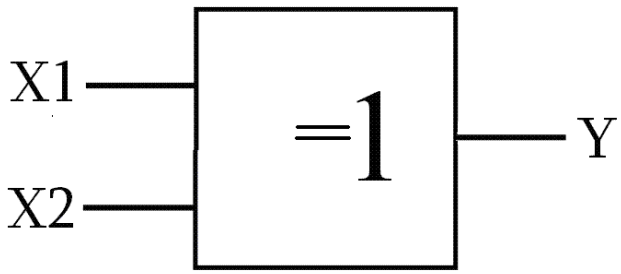
Okrem dvojvstupových logických členov OR sa často využívajú aj viacvstupové členy. Typicky známe sú trojvstupové, štvorstupové a osemvstupové členy OR. Ak nie sú využité všetky vstupy logického člena OR, na nepripojených vstupoch je automaticky priradená logická hodnota 0, aby bol zaručený správny výstup logického súčtu.

2.2.3 Logický člen XOR

Logický člen XOR sa používa na realizáciu funkcie neekvivalencie, často označovanej aj ako funkcia logického súčtu modulo 2. Schematická značka logického člena XOR je odvodená od značky člena OR a vyzerá nasledovne::



Obr. 3a Logický člen XOR podľa štandardu IEC



Obr. 3b Logický člen XOR podľa štandardu IEEE

Pravdivostná tabuľka pre operáciu neekvivalencie realizovanú na dvojvstupovom logickom člene XOR má tvar:

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 5 Pravdivostná tabuľka operácie logického súčtu modulo dva

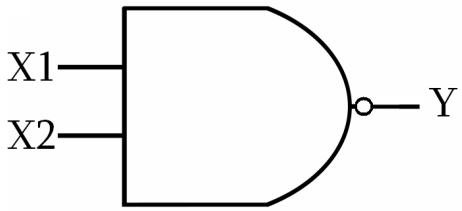
Pre vstupy X1, X2 a pre výstup Y platí:

$$Y = X1 \oplus X2$$

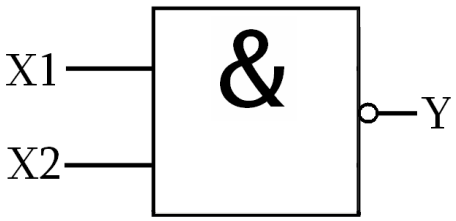
Prakticky sa člen XOR používa na porovnávanie dvoch signálov, pričom indikuje ich rozdielnosť. V prípade, keď je jeden zo vstupov nepripojený, správa sa člen XOR ako keby bola na nepripojený vstup pripojená logická 0. Vzhľadom na funkcionalitu poznáme iba dvojvstupové logické členy XOR.

2.2.4 Logický člen NAND

Logický člen NAND sa používa na realizáciu Shefferovej funkcie. Schematicky sa logický člen NAND zobrazuje dvomi spôsobmi:



Obr. 4a Logický člen NAND podľa štandardu IEC



Obr. 4b Logický člen NAND podľa štandardu IEEE

Pravdivostná tabuľka pre operáciu negovaného logického súčinu realizovanú na dvojvstupovom logickom člene NAND má nasledovný tvar:

X1	X2	Y
0	0	1
0	1	1
1	0	1
1	1	0

Tab. 6 Pravdivostná tabuľka operácie negovaného logického súčinu

Pre výstup Y platí:

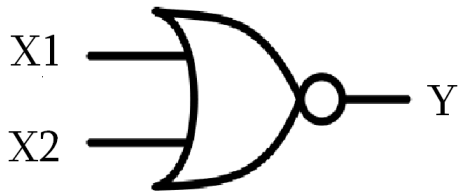
$$Y = \overline{X1.X2}$$

$$Y = X1 \uparrow X2$$

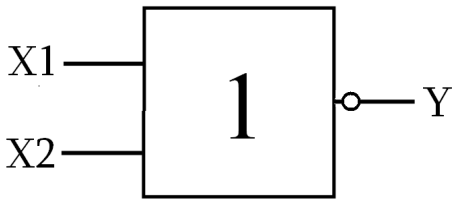
V praxi sa okrem dvojvstupových logických členov NAND využívajú aj trojvstupové, štvorstupové a osemvstupové členy NAND. Ak logický člen NAND nevyužíva všetky svoje vstupy, na nevyužitých vstupoch je automaticky priradená logická hodnota 1, aby bol zaručený správny výstup logického súčinu.

2.2.5 Logický člen NOR

Logický člen NOR sa používa na realizáciu Pierceovej funkcie. Schematicky sa logický člen NOR zobrazuje nasledovne:



Obr. 5a Logický člen NOR podľa štandardu IEC



Obr. 5b Logický člen NOR podľa štandardu IEEE

Pravdivostná tabuľka pre operáciu negovaného logického súčtu realizovanú na dvojvstupovom logickom člene NOR:

X1	X2	Y
0	0	1
0	1	0
1	0	0
1	1	0

Tab. 7 Pravdivostná tabuľka operácie negovaného logického súčtu

Pre výstup Y platí:

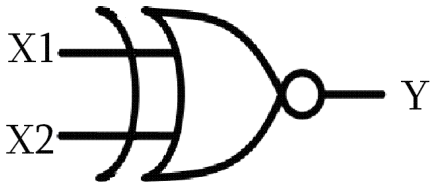
$$Y = \overline{X1 + X2}$$

$$Y = X1 \downarrow X2$$

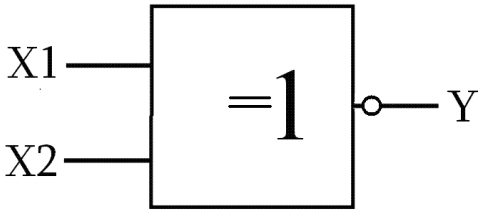
V praxi sa okrem dvojvstupových logických členov NOR využívajú aj trojvstupové, štvorstupové a osemvstupové členy NOR. Ak logický člen NOR nevyužíva všetky svoje vstupy, na nevyužitých vstupoch je automaticky priradená logická hodnota 0, aby bol zaručený správny výstup logického súčtu.

2.2.6 Logický člen XNOR

Logický člen XNOR realizuje funkciu ekvivalencie. Schematicky sa logický člen XNOR zobrazuje dvomi spôsobmi:



Obr. 6a Logický člen XNOR podľa štandardu IEC



Obr. 6b Logický člen XNOR podľa štandardu IEEE

Pravdivostná tabuľka pre operáciu negovaného logického súčtu realizovanú na dvojvstupovom logickom člene NOR:

X1	X2	Y
0	0	1
0	1	0
1	0	0
1	1	0

Tab. 8 Pravdivostná tabuľka operácie ekvivalencie

Pre výstup Y platí:

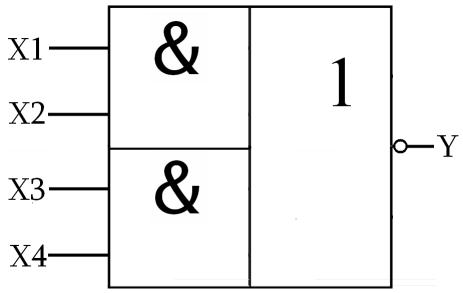
$$Y = X1 \equiv X2$$

V praxi sa člen XNOR používa na porovnávanie dvoch signálov, keď indikuje ich podobnosť. V prípade, keď je jeden zo vstupov nepripojený, správa sa člen XNOR ako keby bola na nepripojený vstup pripojená logická 0. Vzhľadom na funkcionalitu poznáme iba dvojvstupové logické členy XNOR.

2.2.7 Logický člen AND-NOR

Logický člen AND-NOR vznikol kombináciou dvoch logických členov AND a jedného logického člena NOR. Je to vlastne dvojstupňový kombinačný obvod, kde výstupy členov AND sú pripojené na vstup člena NOR. Všetky členy a prepojenia sú integrované v jednom puzdre integrovaného obvodu AND-NOR. Okrem štvorvstupového člena AND-NOR sa najčastejšie používa osemvstupový člen AND-NOR. ten sa skladá zo štyroch dvojvstupových členov AND a jedného štvorvstupového člena NOR alebo z dvoch dvojvstupových členov AND, jedného štvorvstupového člena AND a jedného trojvstupového člena NOR.

Schematicky sa člen AND-NOR zvykne zobrazovať ako kombinácia schematických značiek členov AND a NOR



Obr. 7 Logický člen AND-NOR podľa štandardu IEEE

Pravdivostná tabuľka pre operáciu realizovanú na štvorvstupovom logickom člene AND-NOR má tvar::

X1	X2	X3	X4	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. 9 Pravdivostná tabuľka operácie realizovanej členom AND-NOR

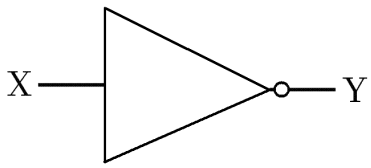
V konečnom dôsledku realizuje člen AND-NOR logickú funkciu

$$\overline{Y = X1.X2 + X3.X4}$$

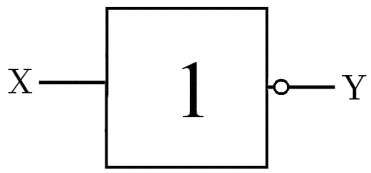
$$Y = X1.X2 \downarrow X3.X4$$

2.2.8 Logický člen NOT

Logický člen NOT, ktorý sa často označuje ako invertor alebo negátor, realizuje funkciu logickej negácie. Na výstupe člena NOT je vždy signál opačnej hodnoty ako na jeho vstupe. Schematicky sa logický člen NOT zobrazuje:



Obr. 8a Logický člen NOT podľa štandardu IEC



Obr. 8b Logický člen NOT podľa štandardu IEEE

Pravdivostná tabuľka pre operáciu logickej negácie realizovanú členom NOT má tvar:

X	Y
0	1
1	0

Tab. 10 Pravdivostná tabuľka operácie logického súčinu

Pre vstup X a výstup Y platí:

$$Y = \overline{X}$$

2.3 Úplný súbor logických členov

Pojem úplný súbor logických členov predstavuje takú množinu elementárnych logických členov, pomocou ktorej môžeme vyjadriť všetky operácie z nejakej funkčne úplnej množiny operácií. Funkčne úplná množina operácií je množina takých operácií, pomocou ktorých môžeme vyjadriť akúkoľvek B-funkciu. V literatúre [1] môžeme nájsť dôkaz, že takouto funkčne úplnou množinou operácií je množina $\{.,+,\bar{}\}$. Z toho vyplýva, že ak pomocou množiny logických členov dokážeme realizovať operácie negácie, logického súčtu a logického súčinu, tak táto množina logických členov je úplná. Pojem minimálna úplná množina logických členov označuje takú množinu, z ktorej nemôžeme odobrať ani jeden logický člen bez toho, aby sa porušila jej úplnosť.

Ako príklad uvádzame množinu logických členov {AND-NOR,NAND} a pokúsme sa dokázať jej úplnosť.

1. $x = x \,.\, x = x \uparrow x = x\uparrow$

2. $x = \overline{x \,.\, x} = x \,.\, x + x \,.\, x$ - AND-NOR

3. $x \,.\, y = \overline{\overline{x \,.\, y}} = (x \uparrow y) \uparrow$

4. $x \,.\, y = \overline{\overline{x}} \,.\, \overline{\overline{y}} = (\overline{x} + \overline{y}) = ((x \,.\, x + x \,.\, x) + (y \,.\, y + y \,.\, y)) - 3x$ AND-NOR

5. $x + y = \overline{\overline{x + y}} = (\overline{x} \,.\, \overline{y}) = (x \uparrow) \uparrow (y\uparrow)$

6. $x+y = \overline{\overline{x} + \overline{y}} = (\overline{x} \,.\, \overline{y}) = ((x \,.\, x + x \,.\, x).(y \,.\, y + y \,.\, y)) + ((x \,.\, x + x \,.\, x).(y \,.\, y + y \,.\, y))$

Dokázali sme, že pomocou logického člena NAND a rovnako aj pomocou logického členu AND-NOR dokážeme vyjadriť úplnú množinu operácií {.,+,¯}. Na základe tohto dôkazu môžeme sformulovať tieto tvrdenia:

1. Logický člen NAND tvorí sám o sebe úplný súbor logických členov.

2. Logický člen AND-NOR tvorí sám o sebe úplný súbor logických členov.

3. Množina logických členov {AND-NOR, NAND} tvorí úplný súbor logických členov

4. Množina logických členov {AND-NOR, NAND} netvorí minimálny úplný súbor logických členov, pretože aj keby sme z množiny vypustili ktorýkoľvek z logických členov, táto množina by stále bola úplná.

2.4 Prehľad existujúcich programových systémov

V súčasnosti možno nájsť niekoľko programov na návrh a simuláciu logických obvodov. Ako príklad môžeme uviesť LOG, Logsim, SimulWorks, Simcir a Logicly. V tejto kapitole

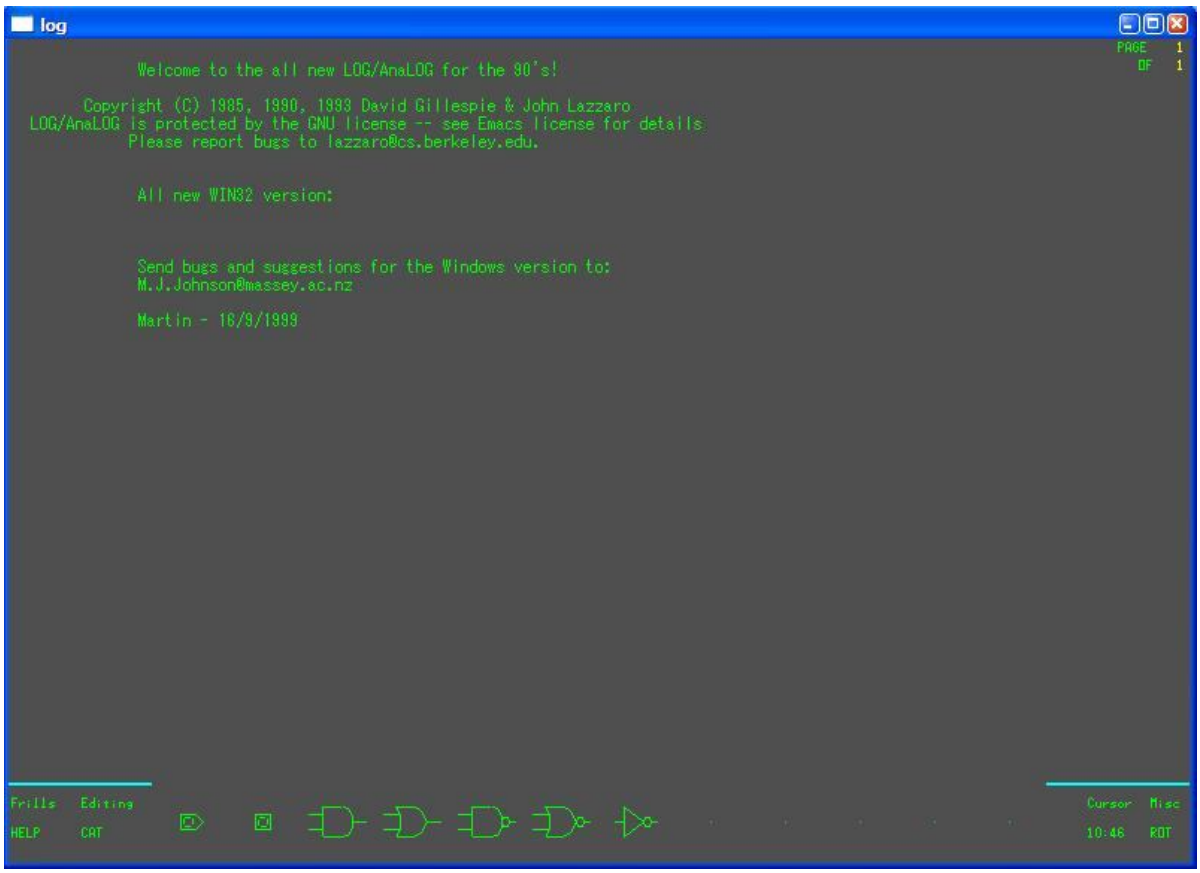
budeme opisovať väčšinu týchto programov. Chceme sa zamerať hlavne na funkcionalitu, preto nebudeme opisovať každý program dopodrobna, keďže by sme len opakovali podobné vlastnosti.

2.4.1 LOG

Tento program, ktorý bol vytvorený v roku 1985, je síce už svojou vizuálnou stránkou zastaraný a rovnako aj intuitívnosť prostredia je dosť zlá, je však doteraz využívaný napríklad vo výučbovom procese. Jeho sila je vo funkcionalite, ktorú neprekonávajú ani niektoré mladšie programy. Slúži na návrh veľkých obvodov a následnú simuláciu a toto všetko je podporované bohatou knižnicou.

Užívateľské prostredie

Po spustení LOG-u môžeme vidieť hlavnú obrazovku. Táto obrazovka nám poskytuje priestor na návrh obvodu a možnosti editácie, ktoré sa nachádzajú v dolnej časti. V pravom rohu hlavnej obrazovky sa nachádzajú informácie o aktuálnej strane a o počte všetkých strán. LOG teda podporuje aj vytváranie niekoľkých obvodov naraz, pričom sa medzi nimi prepíname číslami 1 až 9. Pri ukladaní vidíme veľkú nevýhodu LOG-u, pretože do súboru uložíme iba aktuálnu stránku. Toto je pre užívateľa dosť mätúce, pretože pri ukladaní program neupozorní užívateľa hlásením o uložení iba danej stránky. Máme možnosť sa posúvať v rámci jednej stránky a to smerovými šípkami na klávesnici.



Obr. 9 Prostredie programu LOG

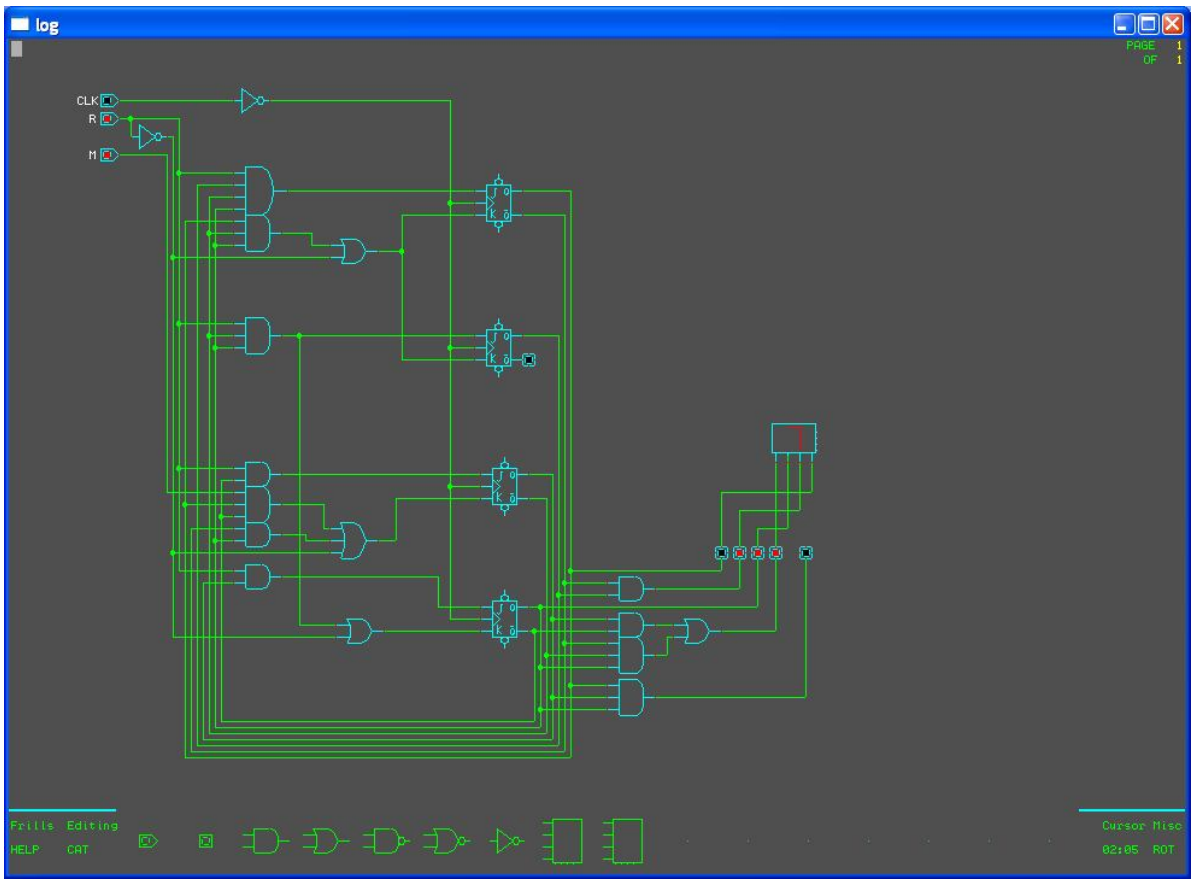
Editačná časť v dolnej časti obrazovky sa skladá z logických členov a záložiek. Ak sa vieme orientovať medzi jednotlivými položkami, práca s ňou je veľmi rýchla a neúnavná. V strede editačnej časti sa nachádzajú logické členy. Sem je možné umiestniť členy a to pomocou drag and drop. Tak isto môžeme následne tieto členy umiestňovať na stranu s navrhovaným obvodom. Kliknutím na člen, či už je v obvode, alebo len v spodnej lište, ho rotujeme o 90° proti smeru hodinových ručičiek. Keď už máme vložené členy na pracovnej ploche, môžeme ich začať spájať. Jedným kliknutím na pracovnej ploche začneme kresliť čiaru, ktorú môžeme ťahať v štyroch smeroch - hore, dole, doprava a doľava. Šikmé prepojenia nie sú dovolené, čo je pozitívne pre prehľadnosť zložitých obvodov.

Editačné záložky sa nachádzajú v pravom a v ľavom dolnom rohu hlavnej obrazovky. Po kliknutí na ne sa buď vyroluje záložka alebo sa prepneme do inej obrazovky, ako je to v prípade záložky CAT.

Funkcionalita

Začíname so záložkou Cursor. V tejto záložke nájdeme položky ako Grid, pomocou ktorého sa okolo kurzora zobrazí kríž. Probe zmení kurzor na sondu logickej hodnoty na prepojení, nad ktorým máme práve kurzor. Glow zobrazí farebne aktuálnu hodnotu na všetkých

prepojeniach. Táto funkcia je veľmi nešťastne umiestnená, pretože s heslom Cursor nemá nič spoločné. Alt vráti posunutý obraz na predošlý, Home vráti obraz do centra kreslenia, Refresh obnovuje obrazovku. V záložke Misc môžeme nájsť funkcie ukladania a načítavania. Ďalej máme možnosť zobrazovať navrhnutý obvod ako výkres, zobrazovať rôzne informácie o vytvorenom obvode a spúšťať a vypínať simuláciu. Ako ďalšia položka záložiek je ROT, MIRX, MIRY a CNFG, medzi ktorými je možné sa prepínať klikaním na záložku. Prepínaním určujeme rotáciu objektov. Táto funkcia je editačná, teda jej umiestnenie by bolo lepšie v záložke Editing. V záložke Editing nájdeme funkcie mazania, kopírovania, otvorenia horizontálneho a vertikálneho bloku, v ktorých je možne používať spojenia medzi obvodmi len v jednej orientácii. Záložka Frills obsahuje možnosti zväčšovania, popisovania objektov, vytvárania boxov a oddeľovania blokov obvodu. Jednou z najdôležitejších záložiek je Cat, kde nachádzame knižnicu logických obvodov. Veľkým problémom LOG-u sú informácie o obvodoch, ktoré sú veľmi krátke a v niektorých prípadoch chýbajú. Samozrejme, nie je problém nájsť tento popis na internete, avšak obohatenie informácií by určite urýchlilo prácu s programom. Ako poslednú záložku spomenieme Help, ktorá nemá vôbec žiadny význam. Po kliknutí na ňu, sa zobrazí výpis: „Prečítaj si Log.doc súbor“, čo opäť zdržuje prácu s programom.



Obr. 10 Okno programu LOG s nakreslenou schémou

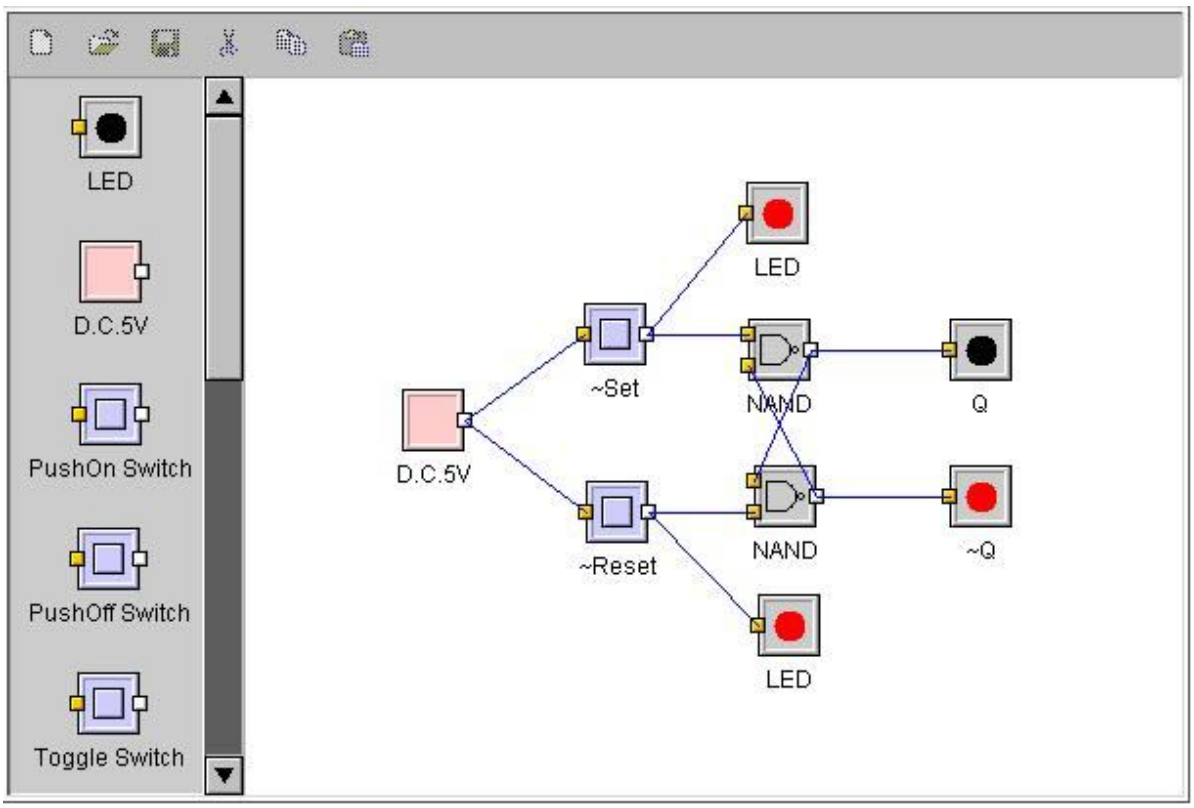
Program LOG je, podľa nášho názoru, veľmi dobrým návrhovým prostredím, ktorým sa chceme inšpirovať. Umožňuje návrh rozsiahlych systémov a aj prehľadnú simuláciu. Aj napriek vyššie spomenutým nedostatkom je tento program akýmsi referenčným modelom v tejto kategórii programov.

2.4.2 Simcir

Tento program je internetovou aplikáciou rovnako slúžiacou na návrh logických obvodov. Simcir jeho autor ešte nedokončil, preto sa v ňom môžeme stretnúť napríklad s nefunkčnými tlačidlami. Na spustenie programu je potrebné mať internetový prehliadač a nainštalované virtuálne prostredie JAVA Runtime Enviroment.

Užívateľské prostredie

Ako môžeme vidieť z obrázku, prostredie je oveľa viac prívetivé ako v LOG-u. Rovnako má pracovnú plochu na kreslenie obvodu. Knižnica s obvodmi je neustále otvorená a je zložená z maximálne dvojvstupových obvodov. Už klasicky známe editačné nástroje sú v hornej lište. Výhodou malej funkcionality tohto programu je, že užívateľ ju celú zvládne po niekoľkých minútach. Nemôžeme však uprieť tomuto programu jeho dobre spracovanú vizuálnu stránku, v ktorej sa vyzná aj úplný začiatočník.



Obr. 11 Prostredie programu Simcir

Funkcionalita

Oproti LOG-u je funkcionalita programu veľmi obmedzená. Simcir napríklad umožňuje vytváranie šikmých čiar, čo pri zložitom obvode veľmi kazí prehľadnosť. Na druhej strane, spojenia logických členov sú stále, preto pri premiestnení člena nestrácame spojenia. Veľmi dobre môžeme hodnotiť ošetrenia chýb. Program neumožňuje vytvárať chybové stavy, ktoré sa v LOG-u vytvoriť dajú. Program umožňuje aj simuláciu na základe vložených logických sond. Knižnica je veľmi jednoduchá a chudobná na logické členy. Táto vlastnosť pri výučbe určite nie je nepriaznivá, nakoľko podnecuje študenta ku kreatívnosti. Ak by sme však potrebovali vytvoriť zložitý obvod, tak obsah knižnice by nám dosť znepríjemňoval návrh.
Z pohľadu intuitívnosti hodnotíme Simcir veľmi pozitívne, ale nízka úroveň funkcionality je nepriaznivá pre zložitejšie využitie.

2.4.3 Espresso

Program Espresso zaraďujeme do analýzy kvôli nevyhnutnosti využívať tento program pri návrhu logických obvodov, a to vo fáze skupinovej minimalizácie logických funkcii. Pri

návrhu obvodu je potrebné vytvoriť logickú funkciu, z ktorej potom môžeme vytvárať obvod. Túto funkciu je potrebné dostať s minimálnym počtom členov. Práca s programom je relatívne zložitá, pretože najskôr potrebujeme vytvoriť vstupný súbor, ktorý musí mať presnú štruktúru. V tomto súbore definujeme množiny vstupov a výstupov a aj výsledný tvar funkcie MDNF alebo MKNF. Potom musíme premiestniť vstupný súbor do priečinka, kde máme espresso.exe a v príkazovom riadku spustiť Espresso s parametrami: mena vstupného súboru a mena výstupného súboru.

```
D:\Skola\4rocnik\7. semester\timovy projekt\existujuce riesenia\Log>espresso -oe
qntott in.txt
J1 = (x&z2&!z3&!z4);

K1 = (!x&z2&!z3&!z4);

J2 = (x&!z1&!z3&z4) | (!x&z1&!z3&!z4);

K2 = (!x&!z1&!z3&z4) | (x&z1&z2&!z3&z4);

J3 = (!x&!z2&z4&!z5) | (!x&!z1&z2&!z4) | (x&!z1&z2&z4) | (x&z1&!z4) | (!x
    &z1&z2&z4) | (x&!z2&!z4);

K3 = (x&!z1&z2&!z4) | (!x&!z1&z2&z4) | (!x&z1&!z4) | (x&z1&z4) | (!x&!z2
    &!z4) | (x&!z2&z4);

J4 = (!x&!z1&z2&z3) | (x&z1&z3) | (x&!z2&z3) | (!x&z1&!z2&!z3);

K4 = (x&!z1&z2&z3) | (!x&z1&z3) | (x&z1&z2&!z3&z4) | (!x&!z2&z3);

J5 = (x&z1&!z2&!z3&z4);

K5 = (!x&z1&!z2&!z3);


D:\Skola\4rocnik\?DF55~1.SEM\TIMOVY~1\EXISTU~1\Log>_
```

Obr. 12 Volanie programu espresso.exe s potrebnými prepínačmi a tvar výstupu

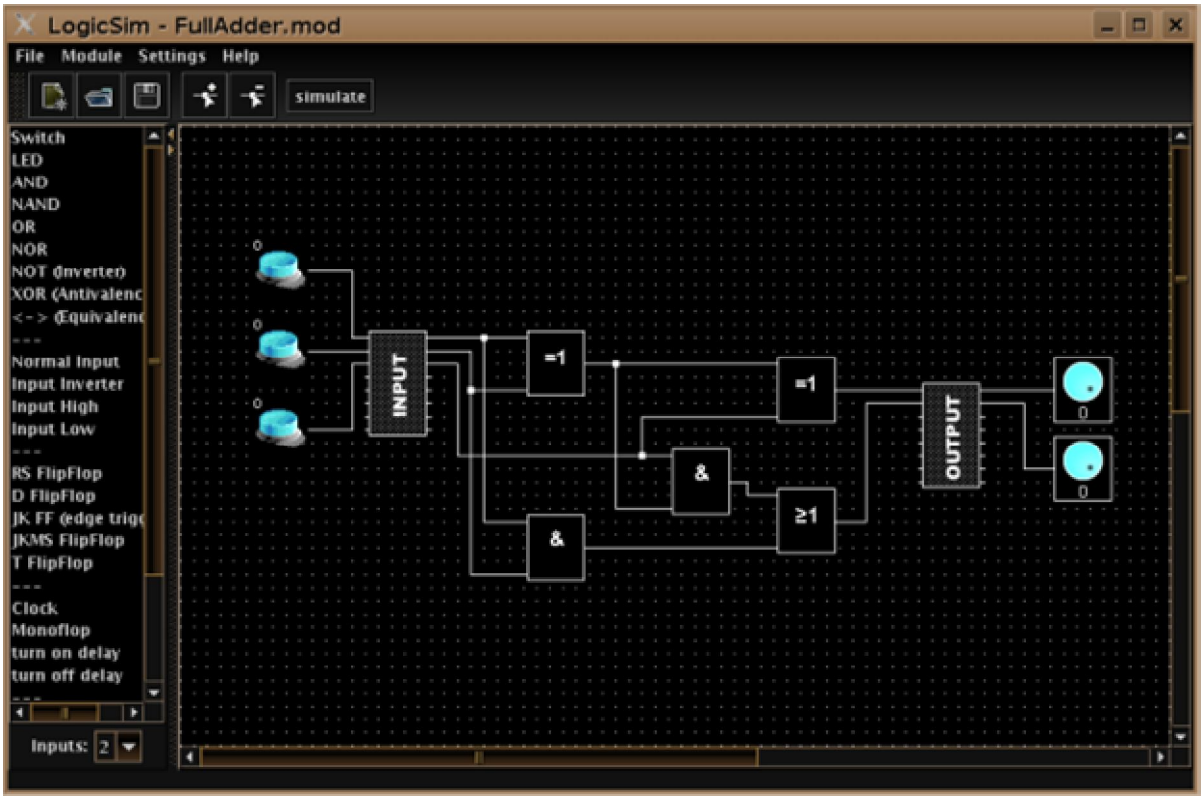
2.4.4 LogicSim

Program LogicSim začal vznikať v roku 1995. Na začiatku bol vytváraný v jazyku Pascal. Poslednou verziou je verzia 2.4 z roku 2009.

Užívateľské prostredie

Program je vytvorený ako klasická oknová aplikácia. Užívateľ má možnosť po spustení vidieť knižnicu, pracovnú plochu a lišty pre rýchly prístup k často využívaným funkciám. Hneď na úvod vidíme problém knižnice, ktorá je síce rozsiahla oproti Simcir-u, ale neprehľadná. Možno by pomohlo zobrazenie obrázku obvodu, prípadne popisu obvodu, na ktorý klikneme.

Na druhej strane celkovo program pôsobí prívetivo a intuitívne. Pri práci s ním užívateľ nemusí nič hľadať, všetko je na správnom mieste.



Obr. 13 Prostredie programu LogicSim

Funkcionalita

Funkcionalita programu je veľmi podobná LOG-u, ale ovládanie jednotlivých funkcií je oveľa príjemnejšie. Je to hlavne tým, že systém umiestnenia je totožný s určitou, dnes zaužívanou, schémou. Ako zaujímavé funkcie môžeme spomenúť zmenu počtu vstupov všetkých logických členov v knižnici či možnosť spájania logických členov do blokov. Slabšou stránkou je užívateľská podpora, ktorá sa skladá iba z dvoch strán textu.

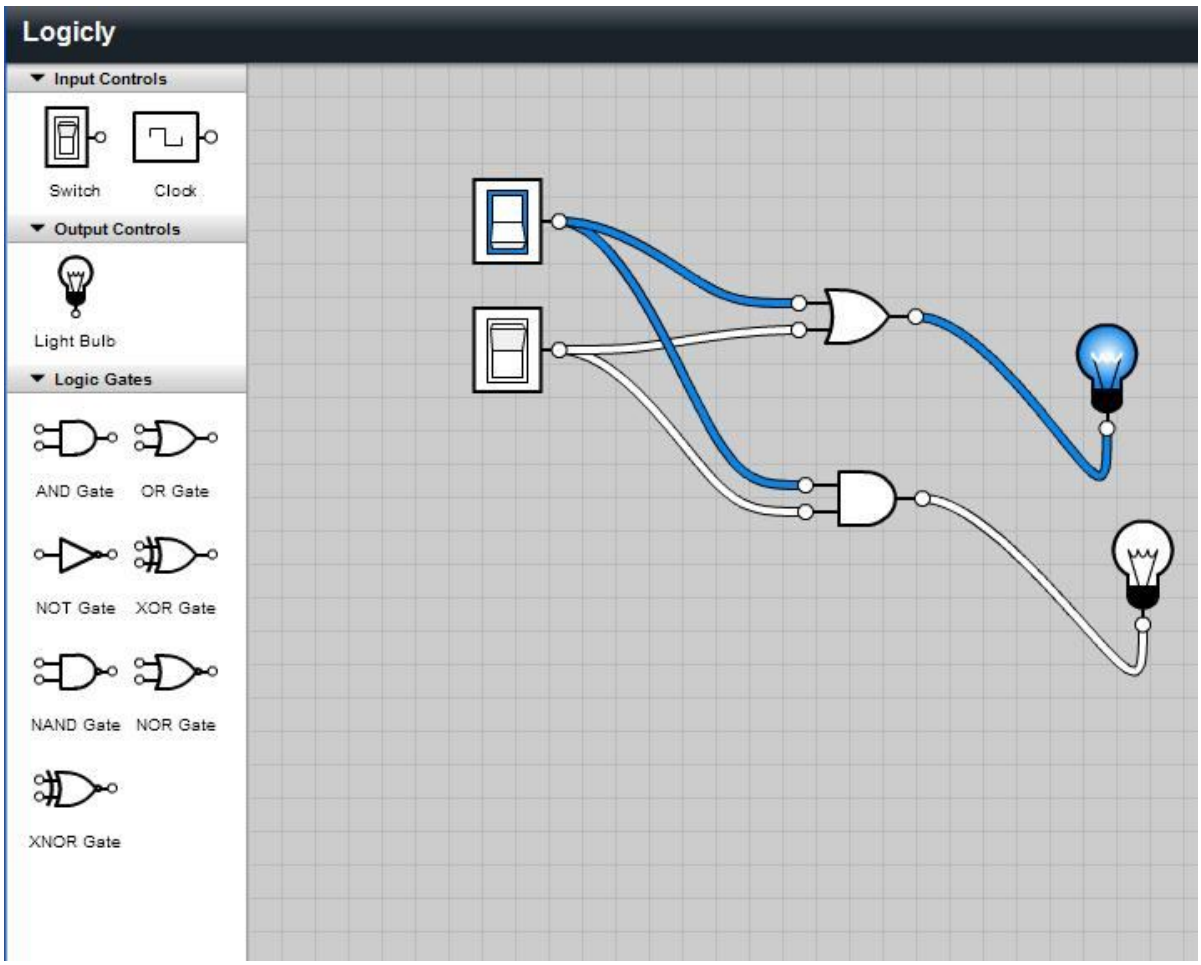
Program neposkytuje žiadne výnimočné funkcie oproti ostatným, avšak poskytuje príjemne prostredie, čo umožňuje efektívnu prácu s ním.

2.4.5 Logicly

Je jedným z on-line programov slúžiacich na návrh obvodov. Pre spustenie programu je nevyhnutné mať nainštalovaný Adobe Flash Player.

Užívateľské prostredie

Hneď pri prvom pohľade na aplikáciu zisťujeme, že ide o aplikáciu, ktorá neposkytuje rozsiahle možnosti. Všetky funkčné prvky sa nachádzajú na hlavnej obrazovke. Knižnica obsahuje sedem členov, ktoré sa ťahajú na pracovnú plochu. Prvky žiarovka, hodiny a vypínač slúžia na simuláciu a kontrolu obvodu.



Obr. 14 Prostredie programu Logicly

Funkcionalita

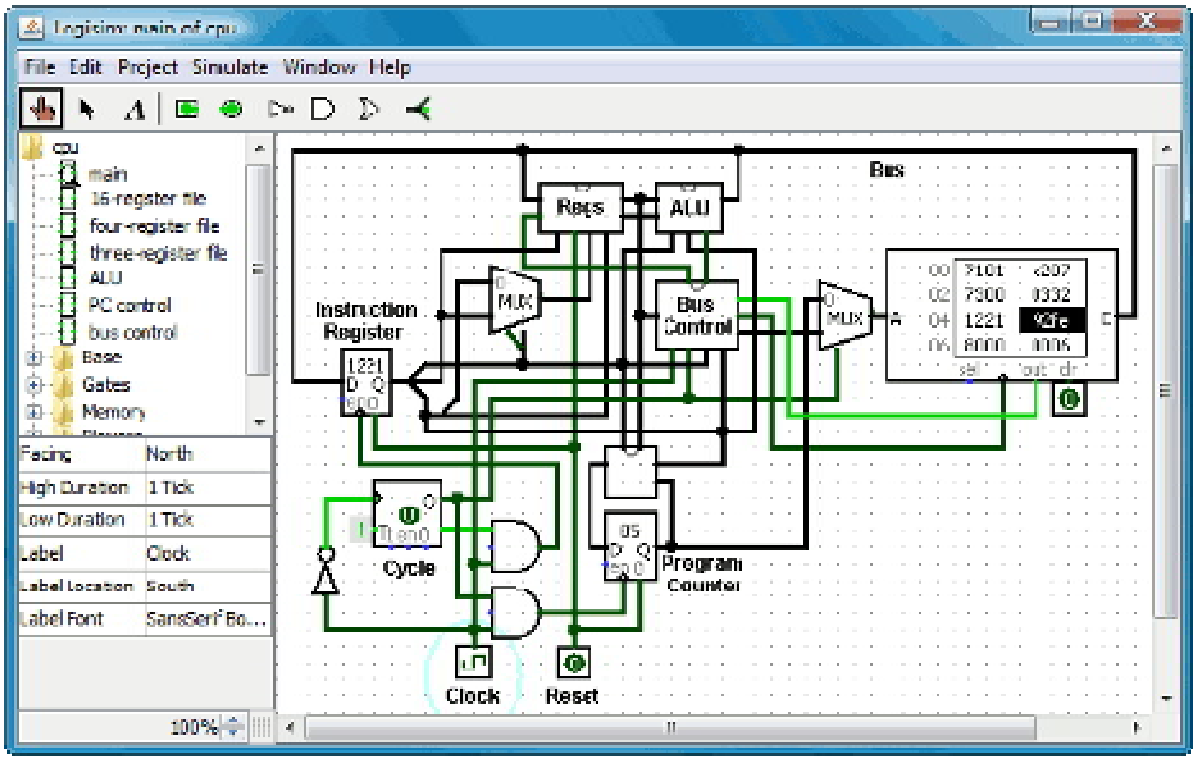
Funkcie programu sú veľmi obmedzené. Chýbajú aj základne funkcie ako ukladanie a načítavanie vytvorenej schémy zo súboru. Tým pádom program slúži iba na návrh jednoduchého obvodu bez možnosti opätovného použitia. Pri vytvorení sekvenčného obvodu program vykazuje chybné výstupy. Ako priaznivé však hodnotím grafické spracovanie. Ak by sa funkcionalita obohatila, mohol by tento programu byť veľmi užitočný.

2.4.6 Logisim

Je jedným z edukačných programov slúžiacich na návrh logických obvodov. Aktuálna verzia je 2.3.1 z roku 2009. Podmienkou spustenia je prítomnosť virtuálneho prostredia JAVA Runtime Enviroment.

Užívateľské prostredie

Na úvodnej obrazovke máme možnosť vidieť najdôležitejšie časti programu - pracovnú plochu, knižnicu, nástrojovú lištu a možnosti zväčšovania. Toto prostredie veľmi pripomína program LogicSim. Tento program je však oveľa prepracovanejší, hlavne knižnica poskytuje oveľa viac členov a ku každému členu je aj veľmi efektívny opis. Takéto rozloženie obrazovky je veľmi prehľadné a ako vidíme z predchádzajúcich programov, aj často používané.



Obr. 15 Prostredie programu Logisim

Funkcionalita

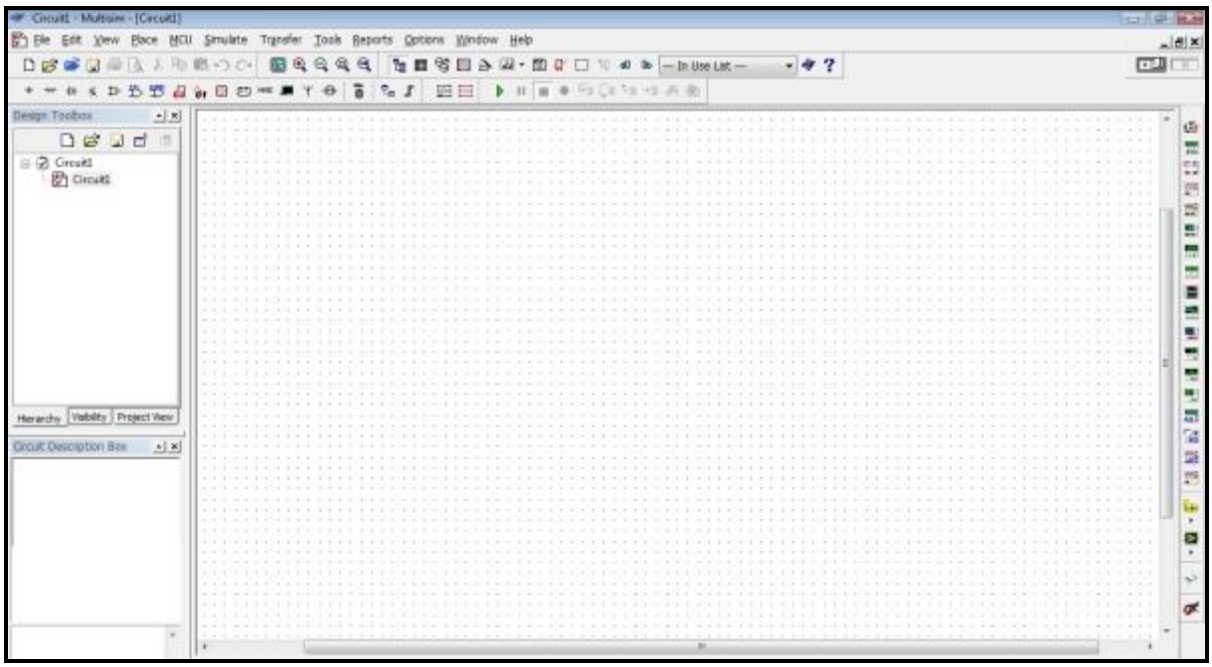
Funkcionalita návrhu a simulácie je podobná LogicSim-u, preto nie je potrebné ju detailnejšie opisovať. Zaujímavé funkcie programu však môžeme nájsť v hlavnej ponuke. Pod položkou Preferences môžme nájsť výber grafického akcelerátora či výber štandardu na vykresľovanie logických členov. Ďalej môžeme aktualizovať knižnicu programu a aj dopĺňať JAVA knižnice. Najzaujímavejšou funkciou tohto programu je analýza vytvoreného projektu. Túto možnosť nájdeme v hlavnom menu pod záložkou Projekt. Po spustení analýzy vyskočí nové okno a máme možnosť získať vstupy, výstupy, tabuľku prepojení a funkcie obvodu. V simulácii môžme meniť časové a frekvenčné premenné, čo robí simuláciu reálnou.

2.4.7 Multisim

Multisim je ďalším programom, ktorý, okrem iného, slúži na vytváranie logických obvodov. V minulosti sa tento program nazýval Electronics Workbench. Poslednou verziou je verzia 10.1.1 z roku 2009.

Užívateľské prostredie

Prostredie programu Multisim 10.1.1 sa od svojich starších verzií takmer vôbec nelíši. Tvoria ho štandardné prvky ako menu, lišta nástrojov, ktorých je mimochodom neúrekom a vykresľovacia plocha. Prostredie nevyniká nijakými zvláštnymi vlastnosťami, ktoré by sme mohli použiť v našom projekte. Za zmienku stojí prijemná štvorcová sieť kresliacej plochy, ktorá uľahčuje kreslenie.

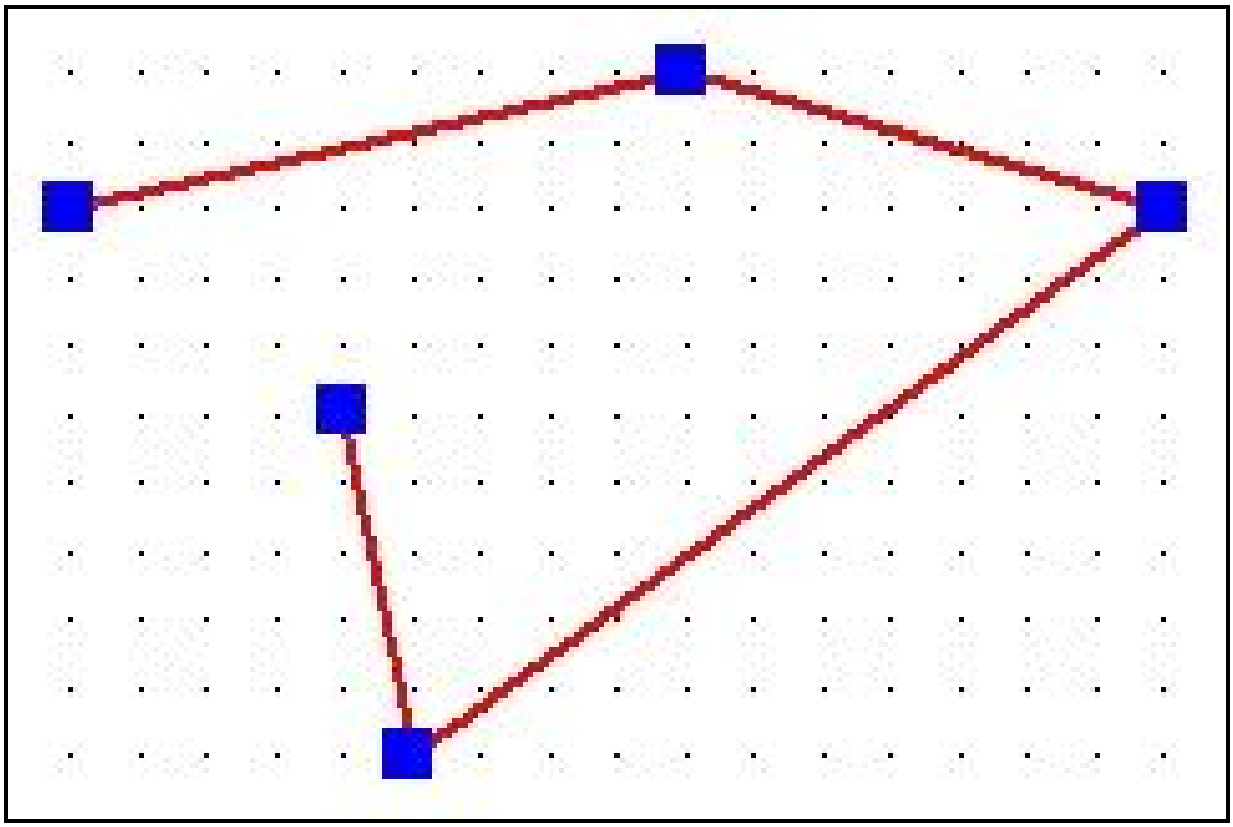


Obr. 16 Prostredie programu Multisim

Funkcionalita

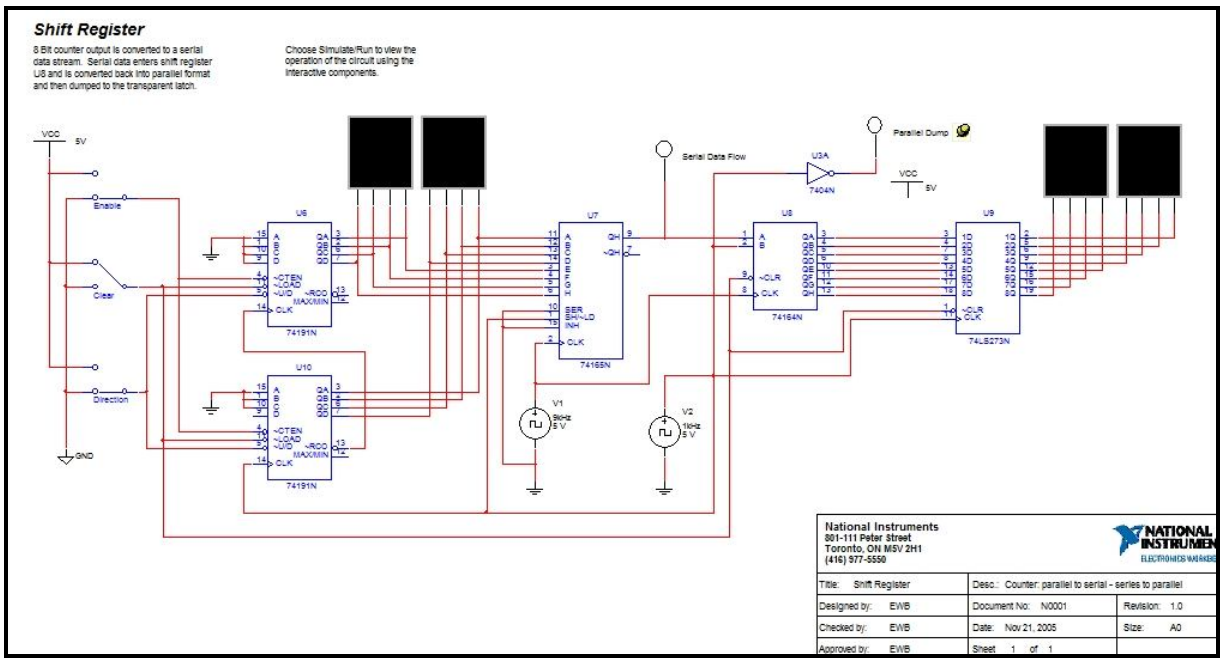
Program Multisim 10.1.1 bol podrobený podrobnej analýze z hľadiska možnosti využitia niektorých jeho častí pre účely použitia v našom programe. Počas analyzovania programu Multisim 10.1.1 sme sa dopracovali k týmto poznatkom.

Program Multisim nám umožňuje kresliť aj iné typy čiar alebo spojov, ako len rovné (napr. Arc, Bezier, Elliptical, Line). Umožňuje nám aj vloženia ľubovoľného obrázka (napr. nejakej schémy) do nákresu. Čiary sa kreslia pomocou zalomenia a potom následného pokračovania v smere pohybu myši. S nakreslenou čiarou je možné neskôr spätne manipulovať. Je možné meniť jej zlomové body.



Obr. 17 Detail kreslenia čiar v programe Multisim

Program ponúka aj vzorové ukážky nejakých vybraných schém, čo sa začínajúcemu používateľovi v tomto programe celkom zíde.



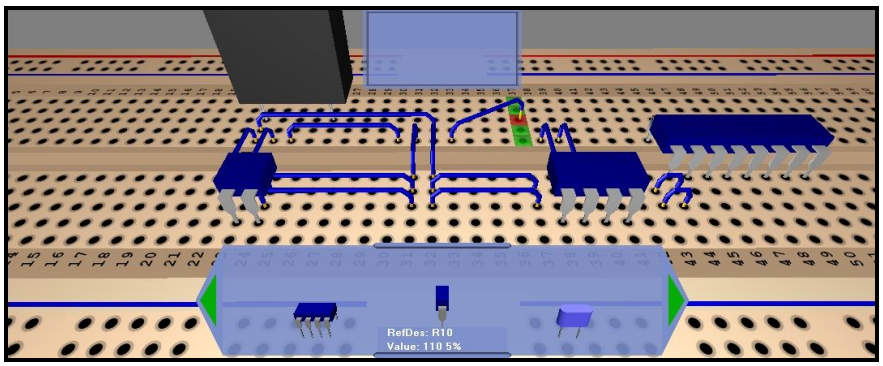
Obr. 18 Preddefinovaná schéma zapojenia v programe Multisim

Za ďalšie veľké plus považujem možnosť overenia konektivity a overenie nastavených pravidiel (v tomto prípade sa jedná o pravidlá z oblasti elektrotechniky). Veľké možnosti ponúka aj knižnica komponentov. Okrem výberu databázy komponentov, kde existuje aj možnosť importovať si vlastná databáza komponentov, ponúka knižnica aj výber skupiny prvkov (komponentov). Táto možnosť je aj pekne graficky spracovaná. Po výbere skupiny sa zobrazí zoznam komponentov patriacich do tejto skupiny. Po vybraní ľubovoľného komponentu sa zobrazí jeho grafická podoba, vypíše funkcia a popis modelu. Program Multisim ponúka aj možnosť využiť pri kreslení schémy nejaký zložitejší prvok, ktorý sa bude skladať z množiny vopred definovaných elektrotechnických súčiastok.

Quantity	Description	RefDes	Package
8	RESISTOR, 1kΩ 5%	R8, R13, R17, R26, R29, R35, R40, R45	IPC-7351\Chip-R0805
4	CAPACITOR, 10nF	C5, C11, C17, C23	IPC-2221A/2222\CAPPA3600-3000X1500
4	RESISTOR, 20kΩ 5%	R14, R59, R60, R61	IPC-7351\Chip-R0805
4	POTENTIOMETER,	R15, R16, R24, R30	Generic\LIN_POT
4	OPAMP, NE5534AD	U1, U2, U3, U7	IPC-7351\SO-8
4	CAPACITOR, 47pF	C3, C12, C18, C24	IPC-2221A/2222\CAPPA3600-3000X1500
3	OPAMP, LM1458M	U5, U6, U12	IPC-7351\SON-8(M08A)
3	DIODE, BAV99	D1, D2, D5	IPC-7351\SOT-23_3
3	OPAMP, MC33078D	U8, U9, U10	IPC-7351\CASE751
2	RESISTOR, 30kΩ 5%	R20, R21	IPC-7351\Chip-R0805
2	RESISTOR, 8.2kΩ 5%	R1, R2	IPC-7351\Chip-R0805
2	RESISTOR, 13kΩ 5%	R22, R23	IPC-7351\Chip-R0805
1	CONNECTORS, HDR1X3	J1	Generic\HDR1X3

Obr. 19 Zoznam súčiastok, z ktorých je možné vyskladať si vlastný logický člen v programe Multisim

Za najväčšie plus tohto programu považujem možné 3D zobrazenie nakreslenej schémy, prípadne aj kreslenie v 3D prostredí. Výber prvkov je však dosť obmedzený, pretože nie všetky komponenty v databáze sú v 3D prevedení. Pri použití kreslenia v 3D prostredí sa zväčša kreslí na jednej pracovnej doske. Je však možné naukladať si aj viac pracovných dosiek vedľa seba.

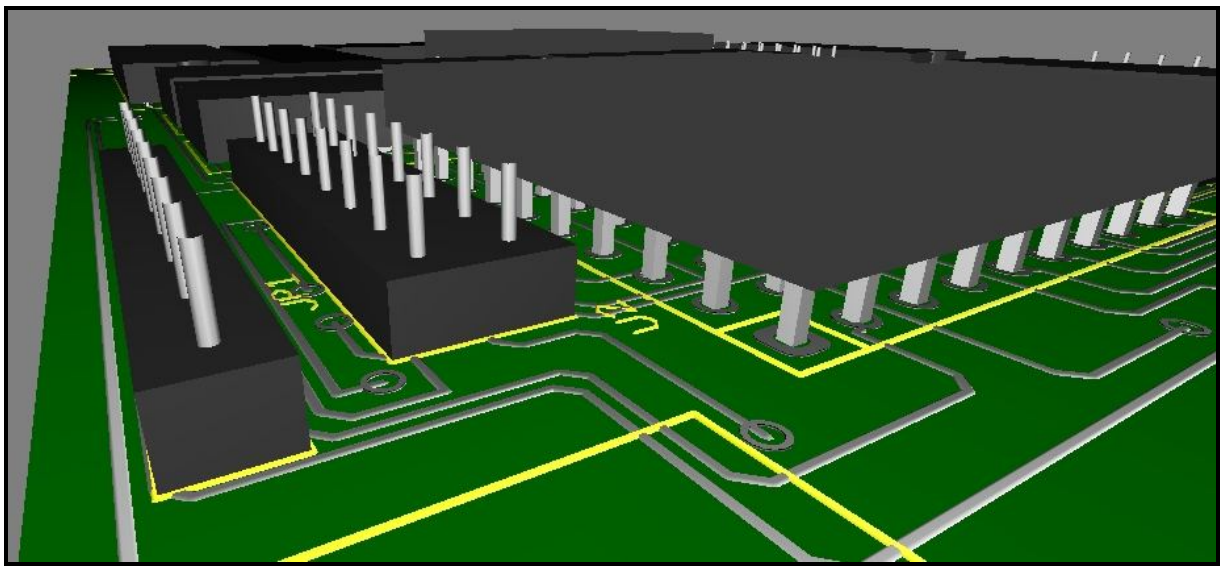


Obr. 20 3D zobrazenie v programe Multisim

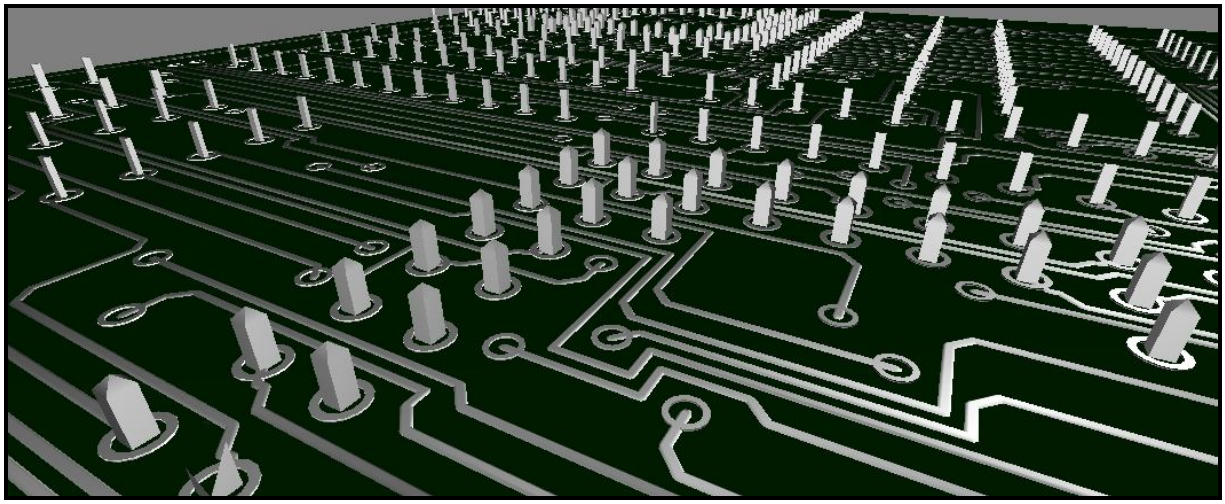
Tento program môžeme zhodnotiť ako naozaj vydarený. Poskytuje takmer neobmedzené možnosti čo sa týka výberu logických členov a veľkou výhodou je aj pridaná funkcionalita a prepracované grafické prostredie.

2.4.8 Ultiboard

Program Ultiboard sa štandardne dodáva k programu Multisim 10.1.1. Jeho terajšia verzia je Ultiboard 10.1.1. Je to program zameraný výlučne na kreslenie schém v 3D prostredí a následné vyhotovenie dosiek plošných spojov. Program je dosť rozsiahly a v základnej podstate jeho funkcionalita je zhruba taká istá ako 3D nadstavba programu Multisim 10.1.1. Jeden zo zásadných rozdielov je v grafickom spracovaní 3D prostredia.



Obr. 21 Ukážka 3D zobrazenia v programe Utilboard



Obr. 22 Program Utilboard slúži aj na návrh dosiek plošných spojov

2.5 Vyhodnotenie analýzy existujúcich riešení

V tejto časti sa budeme venovať výsledkom, ktoré sme získali na základe analýzy ôsmych editačno-stimulačných programov.

Jedným z dôležitých prínosov tejto analýzy je poukázať na rôzne spôsoby rozmiestnenia prvkov a vplyv rozmiestnenia na prácu s týmito nástrojmi. Ako vidíme z analýzy programu Log, rozmiestnenie takýmto spôsobom by určite užívateľa odradilo už pri prvom pohľade. Preto je dôležité držať sa istej zaužívanej formy vzhľadu, ako to vidíme v programoch LogicSim a Logisim. Dobrým riešením je rozdelenie obrazovky na dve časti. Ľavá časť bude predstavovať knižnicu logických členov a zároveň prístup k pridanej funkcionalite, kým v pravej časti bude umiestnená pracovná plocha. Do hornej časti okna programu zakomponujeme hlavne menu zo všetkými funkciami programu.

Ak si už teda vieme predstaviť ako rozmiestniť prvky na obrazovke, môžeme sa začať venovať samotným funkciám. Ako prvú problematiku vidíme spôsob kreslenia prepojení. V programe Log je myšlienka zvislých a vodorovných čiar určite dobrá, no niekedy veľmi obmedzujúca. Pre náš produkt by bolo zaujímavé, aby podporoval rôzne typy čiar ako to vidíme v programe Multism. Avšak ako základné nastavenie určite použijeme spôsob kreslenia z LOG-u. Ak už teda máme predstavu o vytváraní spojení, musíme sa venovať aj ich editácii. Túto možnosť je určite najlepšie umiestniť do hlavnom menu, kde bude možné vyberať spomedzi rôznych editačných nástrojov, či už na editáciu spojení alebo logických

členov. Následne sa môžeme venovať používaniu knižnice. Táto funkcionalita je v podstate v každom analyzovanom programe totožná a preto je zrejmé, že nemá zmysel vymýšľať iný spôsob. Z knižnice bude možné prvky pomocou technológie „ťahaj a pusti“ (drag and drop) presúvať na pracovnú plochu. Problém však nastáva v umiestnení knižnice. Potrebujeme totiž veľké množstvo prvkov z popismi a najlepšie aj obrázkami dostať na časť pracovnej plochy. Zaujímavým, a asi najprehľadnejším, je riešenie v programe Logisim, v ktorom je vytvorená stromová štruktúra, čo zabezpečuje, aby užívateľ nemusel hľadať do nekonečna správny logický člen. Pri hľadaní prvku a po kliknutí naň je v nižšej časti zobrazený obrázok aj popis prvku. Zaujímavou možnosťou je možnosť prepínať sa medzi štandardmi a tým pádom meniť spôsob vykreslenia prvkov v knižnici. Samozrejme, ak už máme vytvorenú schému, program musí dokázať ukladať a načítať iný súbor. Ďalšou funkciou, ktorá nás upútala, bola analýza vytvoreného obvodu. Túto funkcie poskytuje program Logisim, ktorý dokázal vytvárať pravdivostné tabuľky a logické výrazy z vytvoreného obvodu.

Ak sa však začneme venovať funkcionalite, ktorá nám chýbala vidíme že žiadny z programov nepodporoval súčasne možnosť vytvorenia logických výrazov z pravdivostnej tabuľky a zároveň kreslenie schémy. V tomto vidíme veľkú nevýhodu, pretože je potrebné používať viac nástrojov. Ďalej by bolo zaujímavé pre výučbový proces, ak by sa dala obmedziť knižnica. Tato možnosť by mohla byť využitá napríklad pri testovaní študentov. Tak isto by bolo zaujímavé ak by program dokázal komunikovať v okolitej sieti a zdieľať informácie medzi spustenými programami (novo vytvorené členy, nakreslené schémy...).

Z vypracovanej analýzy sme získali dostatočný rozhľad v problematike. Na základe zistení sme schopní presne formulovať požiadavky na nami navrhovaný softvér.

3 Špecifikácia požiadaviek

3.1 Funkcionálne požiadavky

Nami navrhovaná aplikácia bude poskytovať možnosť vytvárania logických obvodov pomocou logických členov, ktoré bude možne vyberať z panela a vkladať na pracovnú plochu programu. Tieto logické členy bude možné prepájať a takto z nich vytvárať požadovaný obvod. Funkčnosť zapojenia bude vyhodnocovaná v reálnom čase, avšak verifikáciu bude možné aj vypnúť, čo môže byť v niektorých špeciálnych situáciách užitočné.

Počas vytvárania obvodu bude možné obvod kedykoľvek uložiť do súboru a následne ho z neho opäť načítať naspäť. Takto vytvorený obvod bude tiež možné pridať medzi externé logické členy, čo bude mať za následok ponúknutie tohto externého logického člena v ponuke medzi ostatnými logickými členmi a bude kedykoľvek pripravený pre použitie v ďalšom logickom obvode. Táto funkcia tak predstavuje jednoduchý spôsob definovania vlastných logických členov, či modulov rozsiahlejších logických systémov. Program tiež poskytne možnosť vytvoriť vlastný logický člen zadaním počtu požadovaných vstupov, výstupov a prechodovej funkcie. Prechodovú funkciu bude možné zadať buď vo forme pravdivostnej tabuľky alebo logického výrazu.

Ďalšou funkciou programu je možnosť výpisu logického výrazu reprezentujúceho nakreslenú schému, čo nájde svoje uplatnenie najmä v procese testovania alebo spätného overenia správnosti nakreslenej schémy.

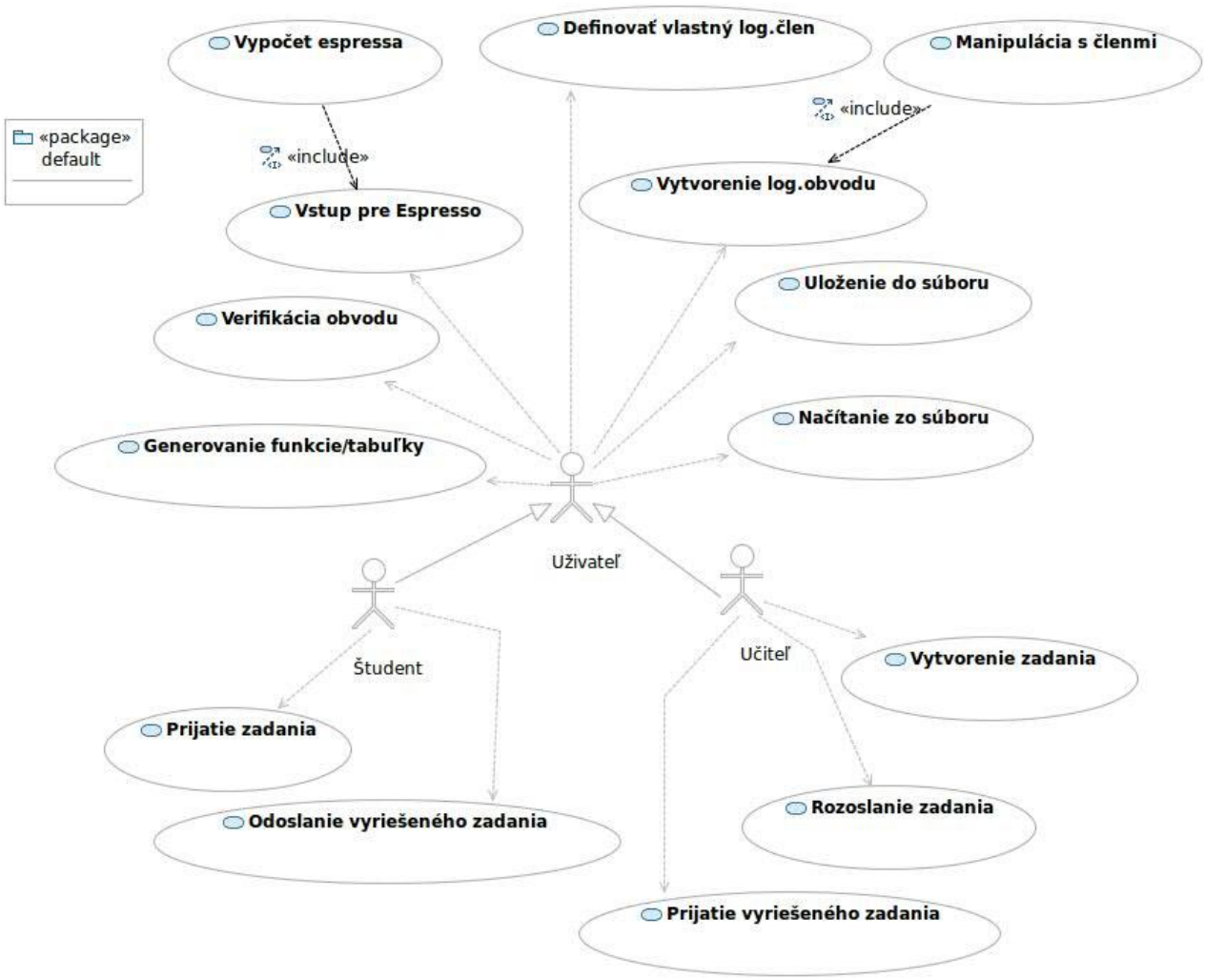
Pomocou prepojenia k programu Espresso bude možné z prostredia programu zadávať parametre pre výpočet skupinovej minimalizácie v tomto programe a následné zobrazenie výsledku opäť v prostredí verifikačného panela.

Hlavnou funkcionalitou, na ktorú sa chceme zamerať, je podpora testovania študentov z problematiky logických obvodov. Z toho dôvodu bude možné v tejto aplikácii vytvárať zadania, ktoré budú následne rozoslané presne špecifikovanej skupine užívateľov (študentov).

Pri vytváraní zadania bude možné definovať rôzne obmedzenia pre potreby konkrétneho testu, napríklad čas, po ktorom sa zadanie odošle nazad učiteľovi na kontrolu, či možnosť definovať, ktoré logické členy bude môcť študent použiť a ktoré mu budú naopak zakázané. Študent po prijatí takéhoto zadania bude môcť riešiť danäe zadanie a po doriešení ho odoslať nazad učiteľovi na vyhodnotenie.

3.2 Opis prípadov použitia systému

Na nasledujúcom obrázku (Obr. 23) je znázornený diagram prípadov použitia navrhovaného systému. Jednotlivé prípady použitia sú popísané ďalej v texte.



Obr. 23 Diagram prípadov použitia systému

Číslo Use Case	Názov Use Case	Hlavný aktér
UC1.1	Vstup pre Espresso	Užívateľ
UC1.2	Výpočet Espressa	Užívateľ
UC2.1	Definovať vlastný logický člen	Užívateľ
UC2.2	Manipulácia s členmi	Užívateľ
UC3	Vytvorenie logického obvodu	Užívateľ
UC4	Uloženie do súboru	Užívateľ
UC5	Načítanie zo súboru	Užívateľ
UC6	Generovanie funkcie / tabuľky	Užívateľ
UC7	Verifikácia obvodu	Užívateľ
UC8	Prijatie zadania	Študent
UC9	Odoslanie zadania	Študent
UC10	Vytvorenie zadania	Učiteľ
UC11	Rozoslanie zadania	Učiteľ
UC12	Prijatie vyriešeného zadania	Učiteľ

Tab. 11 Identifikácia prípadov použitia

V nasledujúcom texte sú stručne popísané jednotlivé prípady použitia nami navrhovaného systému.

Názov: **Vstup pre Espresso**
Číslo Use Case: **UC1.1**
Cieľ: Vytvorenie vstupných údajov pre Espresso
Aktér: Užívateľ
Kontext: Pri potrebe užívateľa získať skupinovú minimalizáciu logických funkcií môže tento využiť funkciu minimalizácie. Pre vytvorenie skupinovej minimalizácie je nutné zadať počet vstupov, počet výstupov a pravdivostnú tabuľku ako vstup pre program Espresso.

Názov: **Výpočet Espressa**
Číslo Use Case: **UC1.2**
Cieľ: Získanie skupinovej minimalizácie logických funkcií
Aktér: Užívateľ

Kontext: Ak už užívateľ nadefinoval vstupné údaje pre Espresso, potom výpočet Espressa zabezpečí získanie logických výrazov v minimálnej forme, ktoré môže užívateľ využívať pri návrhu.

Názov: **Definovať vlastný logický člen**
Číslo Use Case: **UC2.1**
Cieľ: Doplnenie knižnice novým členom
Aktér: Užívateľ
Kontext: Užívateľ má pravo upravovať knižnicu logických členov a to vytváraním nových členov. Užívateľ vyberie funkciu definovať vlastný logický člen a zadá funkciu, ktorú ma člen vykonávať. Po úspešnom vytvorení logického člena je tento člen pridaný do knižnice.

Názov: **Manipulácia s členmi**
Číslo Use Case: **UC2.2**
Cieľ: Doplnenie obvodu logickými členmi
Aktér: Užívateľ
Kontext: Manipulácia s členmi zahŕňa kroky výberu logického člena z knižnice, jeho umiestnenia na pracovnú plochu, presúvanie a prípadné zrušenie daného člena.

Názov: **Vytvorenie logického obvodu**
Číslo Use Case: **UC3**
Cieľ: Vytvorenie logického obvodu užívateľom
Aktér: Užívateľ
Kontext: Užívateľ pomocou čiar poprepája jednotlivé logické členy umiestnené na pracovnej ploche a tak vytvorí logický obovod.

Názov: **Uloženie do súboru**
Číslo Use Case: **UC4**
Cieľ: Uloženie logického obvodu do súboru užívateľom
Aktér: Užívateľ
Kontext: Užívateľ chce uložiť logický obvod do súboru. Kedykoľvek počas vytvárania nového logického obvodu bude možné tento obvod uložiť do súboru jednoduchým stlačením tlačidla Uložiť. Vytvorený logický obvod sa uloží na pevný disk a bude ho možné kedykoľvek opätovne načítať.

Názov: **Načítanie zo súboru**

Číslo Use Case: **UC5**

Cieľ: Načítanie logického obvodu zo súboru užívateľom

Aktér: Užívateľ

Kontext: Užívateľ bude chcieť načítať logický obvod zo súboru. Najskôr si otvorí panel s ponukou a vyberie si súbor obsahujúci požadovaná schému. Po kliknutí na tlačidlo Načítať sa schéma zobrazí na praconej ploche programu.

Názov: **Generovanie funkcie / tabuľky**

Číslo Use Case: **UC6**

Cieľ: Generovanie funkcie / tabuľky užívateľom

Aktér: Užívateľ

Kontext: Užívateľ chce generovať funkciu / tabuľku z logického obvodu. V prvom rade si užívateľ vyberie logický obvod, z ktorého chce vygenerovať pravdepodobnostnú tabuľku alebo funkciu. Následne stlačí tlačidlo generuj a vybraná možnosť, buď funkcia alebo pravdepodobnostná tabuľku sa vygeneruje a vypíše do oznamovacej časti programu.

Názov: **Verifikácia obvodu**

Číslo Use Case: **UC7**

Cieľ: Verifikácia logického obvodu užívateľom

Aktér: Užívateľ

Kontext: Užívateľ bude chcieť verifikovať logický obvod. Najprv si vyberie obvod, ktorý bude chcieť verifikovať, či už to bude predvolený logický obvod, alebo práve ním vytvorený. Následne užívateľ spustí simuláciu, ktorá logický obvod verifikuje a tým sa užívateľ presvedčí o funkčnosti konkrétneho riešenia.

Názov: **Prijatie zadania**

Číslo Use Case: **UC8**

Cieľ: Prijatie zadania študentom

Aktér: Študent

Kontext: Študent chce prijať zadanie. Študentovi sa zobrazí oznam o možnosti prijatia zadania. Po potvrdení sa zadanie zobrazí na hlavnej ploche a študent ho môže následne riešiť.

Názov: **Odoslanie zadania**

Číslo Use Case: **UC9**

Cieľ: Odoslanie zadania študentom

Aktér: Študent

Kontext: Študent bude chcieť odoslať vyhotovené zadania naspať učiteľovi. Po vypracovaní zadania študent stlačí tlačidlo odoslať a vypracovaná úloha sa po zadaní mena študenta odošle učiteľovi, ktorý ju zadal. Pri odoslaní úlohy sa odošle spolu s úlohou aj čas za koľko študent vypracoval úlohu.

Názov: **Vytvorenie zadania**

Číslo Use Case: **UC10**

Cieľ: Vytvorenie zadania učiteľom

Aktér: Učiteľ

Kontext: Učiteľ by chcel vytvoriť zadanie. Najprv si učiteľ vytvorí zadanie. Či už to bude nakreslený logický obvod s chýbajúcimi logickými členmi alebo funkcia, na ktorú budú musieť študenti vytvoriť logický obvod. Učiteľ bude mať možnosť obmedziť dĺžku vypracovanie zadania, s tým že po určitom čase, nastavenom učiteľom, sa zadanie odošle spať, alebo bude mať napríklad možnosť obmedziť študentom použitie logických členov.

Názov: **Rozoslanie zadania**

Číslo Use Case: **UC11**

Cieľ: Rozoslanie zadania učiteľom

Aktér: Učiteľ

Kontext: Učiteľ chce rozoslať zadania. Učiteľ bude mať v ponuke na výber rôzne skupiny študentom, ktorý budú usporiadaný podľa zaradenia do príslušnej skupiny. Učiteľ si vyberie skupinu alebo skupiny, ktorým chce zadanie poslať a stlačí tlačidlo rozoslať. Zadanie sa rozošle študentom vo vybraných skupinách.

Názov: **Prijatie vyriešeného zadania**

Číslo Use Case: **UC12**

Cieľ: Prijatie vyriešeného zadania učiteľom

Aktér: Učiteľ

Kontext: Učiteľ bude chcieť prijať vyriešené zadania. Zadania sa učiteľovi vrátia buď po uplynutí stanoveného času, alebo po odoslaní študentom. Učiteľ si ich potom nájde v záložke

prijaté zadania, kde bude vidieť od koho prišlo zadanie a za aký čas ho študent vypracoval, rovnako ako aj schému nakreslenú študentom.

3.3 Nefunkcionálne požiadavky

Medzi ďalšie požiadavky, ktoré nemajú zásadný vplyv na funkcionalitu patria:

- prívetivé a prehľadné grafické používateľské rozhranie
- multiplatformové vyhotovenie umožňujúce používanie programu na pracových staniciach s rozdielnymi operačnými systémami
- schopnosť programu farebne rozlišovať prepojenia podľa nimi vedenej logickej hodnoty

4 Hrubý návrh riešenia

4.1 Výber programovacieho jazyka

Pre riešenie tohto projektu sa ponúkali rôzne alternatívy programovacích jazykov, ako napríklad Java, C++, C#, Delphi. Po dôkladnom zvážení všetkých možností týchto programovacích jazykov, hrubej špecifikácií funkcionality výslednej aplikácie a jej grafického návrhu sme ako vhodný programovací jazyk pre implementáciu nášho projektu vybrali objektovo-orientovaný programovací jazyk Java.

Jazyk Java bol vybraný ako najvhodnejší hneď z niekoľkých dôvodov:

Prenositeľnosť

Program vytvorený v jazyku Java je možné spustiť na akomkoľvek stolnom či prenosnom počítači prakticky bez softvérových či hardvérových obmedzení. Jedinou podmienkou pre spustenie programu vytvoreného na platforme Java je mať nainštalovaný Java Runtime Enviroment (JRE). JRE je virtuálne prostredie, v ktorom sú spúšťané Java aplikácie. V súčasnosti je JRE dostupné pre všetky známe operačné systémy a hardvérové platformy, od stolných a prenosných počítačov, cez tablety, až po vreckové počítače a mobilné telefóny.

Prívetivosť prostredia

Samozrejmou požiadavkou na aplikáciu typu virtuálny verifikačný panel je komunikácia s používateľom prostredníctvom grafického používateľského rozhrania (GUI – Graphical User Interface) a práve vo vzhľade GUI a v jednoduchosti jeho ovládania je platforma Java najvhodnejšia.

Jednoduchosť jazyka

Programovací jazyk Java patrí medzi vyššie programovacie jazyky. Programovanie v jazyku Java je relatívne jednoduché aj pre nie veľmi skúseného programátora.

Jedným zo subjektívnych dôvodov výberu tohto programovacieho jazyka, boli aj niekoľkonásobné skúsenosti vývojárskej časti tímu s implementáciou iných projektov v tomto programovacom jazyku.

4.2 Prehľad použitých technológií

4.2.1 Swing

V Jave existuje niekoľko knižníc na vytváranie GUI. Swing je jedna z najpoužívanejších knižníc na toto určených hlavne pre jednoduchosť a prehľadnosť. Pomocou tejto knižnice bude realizované grafické rozhranie tejto aplikácie.

4.2.2 Drag and drop

Drag and drop technológia bude realizovaná pomocou triedy TransferHandler a jeho vnorenej triedy TransferSupport. Táto trieda v spojení so svojimi metódami poskytuje všetky potrebné mechanizmy na úspešnú implementáciu drag and drop technológie.

Filozofia použitia tejto technológie bude vo výbere akéhokoľvek logického členu z panelu ponuky a pomocou myšky premiestnenie ho na pracovnú plochu verifikačného panelu. Táto realizácia sa dá rozdeliť do niekoľkých bodov:

1. Výber logického člena z ponuky, stlačenie ľavého tlačidla myši a začiatok ťahania tohto člena. Táto procedúra bude mať za následok aktivovanie ťahania.
2. Keď sa aktivuje ťahanie objektu, zabalia sa dáta pre export a zadeklaruje sa, aká akcia sa bude vykonávať, z dostupných Copy, Move alebo Link. V našom prípade pôjde o akciu Move (Presunutie).
3. Počas ťahania objektu Swing postupne prepočítava pozíciu a riadi prekresľovanie scény.
4. Pri prejdení nad okno verifikačného panela prebehne kontrola či daný cieľový komponent je schopný akceptovať prípadne pustenie objektu. Zistí - na základe spätnej väzby - či je schopný prijať ťahaný objekt a zobraziť ho v požadovanej forme.
5. Pri pustení tlačidla myši cieľový komponent vykoná požadovanú akciu a zobrazí ťahaný objekt v požadovanej forme.

4.2.3 Serializácia

Ukladanie už hotových schém za účelom uchovania hotovej práce bude realizovane pomocou rozhrania Serializable, ktorý umožňuje, či už do pamäte RAM alebo na pevný disk, ukladať objekty vo forme údajového toku v takom stave, v akom sa aktuálne nachádzajú aj s ich premennými a čo je najdôležitejšie, je schopný rekonštruovať tieto serializované objekty do pôvodného stavu.

Spôsob implementácie ukladania vytvoreného riešenia bude realizovaný v hlavnej triede, ktorá bude zabezpečovať pracovnú plochu, kde sa budú jednotlivé komponenty spájať a umiestňovať. Táto trieda bude mať zdedené rozhranie Serializable a implementované metódy writeObject a readObject pre serializáciu a deserializáciu.
Hlavnou myšlienkou bude, že táto trieda bude uchovávať zoznam objektov ktoré sa budú aktívne nachádzať na ploche a s ktorými sa pracuje, pričom každý z objektov bude mať v sebe definovanú polohu v tomto priestore.

4.3 Architektúra systému

Výsledná aplikácia bude realizovaná v troch verziách a to v jednej pre potreby študenta, druhej pre potreby učiteľa a tretej pre užívateľa bez podpory testovania. Z tohto dôvodu je jej architektúra navrhnutá do troch verzií a ich jednotlivých modulov.

- Študent
- Učiteľ
- Užívateľ

4.3.1 Užívateľ

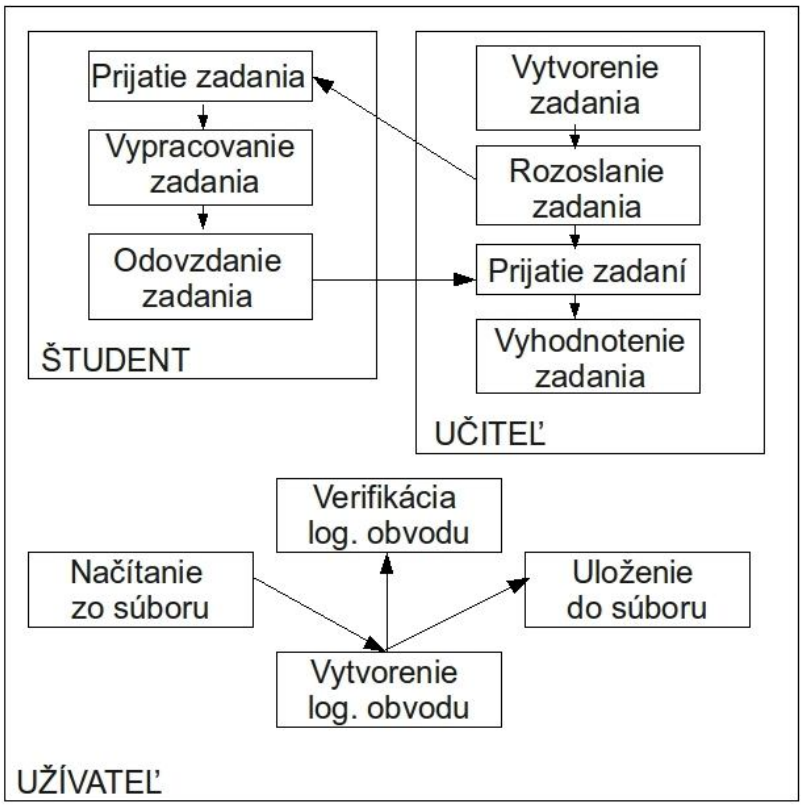
Tento modul obsahuje implementáciu vytvárania logických obvodov, ich verifikovanie, prípadné ukladanie do súboru alebo načítavanie z neho. Tieto moduly budú prístupné aj pre verziu Učiteľ a študent.

4.3.2 Študent

Táto verzia obsahuje okrem zdedených modulov z verzie Užívateľ aj ďalšie prídavné moduly ako je prijatie zadania, vypracovanie zadania (čo je vlastne špeciálny prípad modulu Vytvorenia log. obvodu verzie Užívateľ) a odovzdanie zadania učiteľovi na vyhodnotenie, pre potreby testovania.

4.3.3 Učiteľ

Táto verzia obsahuje okrem zdedených modulov z verzii Užívateľ ďalšie prídavné moduly ako sú vytvorenie zadania, rozoslanie zadania, prijatie zadaní (rozdiel medzi týmto modulom a modulom vo verzii Študent je v tom že Učiteľ očakáva prijatie viac ako jedného zadania) a vyhodnotenie zadania, pre potreby testovania.



Obr.24 Návrh architektúry systému

4.4 Systémové požiadavky

4.4.1 Požiadavky na hardvérové vybavenie

Aplikácia bude mať nízke hardvérové požiadavky a bude spustiteľná na štandardne vybavenom počítači. Minimálne požiadavky sú:

CPU: 1,0 GHz

RAM: 512 MB

Aspoň 20 MB voľného miesta na pevnom disku

Grafická karta

Ukazovacie zariadenie (myš)

4.4.2 Požiadavky na softvérové vybavenie

Nakoľko aplikácia bude implementovaná v programovacom jazyku Java, je potrebné pre úspešné spustenie aplikácie mať nainštalované JDK (Java Developer's Kit) alebo JRE (Java Runtime Enviroment) pre Javu 1.5 a vyššie.

5 Použité zdroje

[1] Frištacký, N., Kolenička, J., Kolesár, M.: *Logické systémy – kombinačné obvody.* Bratislava: Edičné stredisko SVŠT, 1986.

[2] Bača, J.: *Logické obvody*. Bratislava: Alfa, 1977.

[3] Hlavatý, J., Kolesár, M.: *Logické obvody – Kombinačné siete – príklady.* Bratislava: Edičné stredisko SVŠT, 1973.

[4] Palaj T.: *Virtuálny verifikačný panel s členmi AND-NOR a NAND*, Bakalárska práca. Bratislava: FIIT STU, 2009.

[5] LOG: The Chipmunk Logic Simulator (User's Guide) [online]. Dostupný z www: http://caltechcstr.library.caltech.edu/377/ (2009-11-10)

[6] IEEE Standard Graphic Symbols for Logic Functions (ANSI/IEEE Std 91a-1991) [online]. Dostupný z www: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=27895&isnumber=1145 (2009-11-11)

[7] Multisim [online]. Dostupný z www: http://www.ni.com/multisim/ (2009-11-11)

[8] Logisim [online]. Dostupný z www: http://ozark.hendrix.edu/~burch/logisim/ (2009-11-10)

[9] Logicly. Dostupné na Internete: http://joshblog.net/projects/logic-gate-simulator/Logicly.html (2009-11-10)

[10] LogicSim. Dostupné na Internete: http://www.tetzl.de/logicsim_applet.html (2009-11-10)

[11] Simcir. Dostupné na Internete: http://www.d-project.com/simcir/simcir.htm (2009-11-10)

[12] Introduction to DnD [online]. Dostupný z www: http://java.sun.com/docs/books/tutorial/uiswing/dnd/intro.html (2009-11-10)

[13] Interface Serializable [online]. Dostupný z www: http://java.sun.com/j2se/1.4.2/docs/api/java/io/Serializable.html (2009-11-10)

[14] Implementing Serializable [online]. Dostupný z www: http://www.javapractices.com/topic/TopicAction.do?Id=45 (2009-11-10)