

Tímový projekt
Prostredie pre návrh digitálnych
systemov

Vedúci projektu: Ing. Peter Pištek

Tím č. 3: Bc. Michal Kebis
Bc. Tomáš Halagan
Bc. Peter Bôžik
Bc. Ivan Bešina
Bc. Štefan Kováč

Máj, 2012

Obsah

1 ÚVOD	4
1.1 ZADANIE	4
1.2 ÚČEL A ROZSAH DOKUMENTU	5
2 ANALÝZA PROBLÉMU	6
2.1 MODELSIM	6
2.1.1 Používateľské grafické rozhranie.....	6
2.1.2 Manažment projektu	7
2.1.3 Textový editor	7
2.1.4 Šablóny zdrojového kódu a sprievodcovia.....	8
2.1.5 Testbench	9
2.1.6 Simulácia	10
2.1.7 Výhody a nevýhody ModelSimu	11
2.2 KLOGIC.....	12
2.2.1 Používateľské grafické rozhranie.....	12
2.2.2 Návrh logického obvodu	12
2.2.3 Simulácia	12
2.2.4 Výhody a nevýhody produktu KLogic	14
2.3 TKGATE.....	15
2.3.1 Používateľské grafické rozhranie.....	15
2.3.2 Návrh logického obvodu	15
2.3.3 Simulácia	16
2.3.4 Výhody a nevýhody produktu TkGate	17
2.4 LOG	18
2.4.1 Používateľské grafické rozhranie.....	18
2.4.2 Návrh logického obvodu	19
2.4.3 Simulácia	19
2.4.4 Výhody a nevýhody produktu LOG	20
2.5 MVSIS	21
2.5.1 Viac hodnotová logická syntéza.....	21
2.6 CUDD	23
2.6.1 Dátové štruktúry CUDD	24
2.6.2 Princíp práce s CUDD	24
2.7 BDS.....	25
2.7.1 BDD.....	25
2.8 DIGITAL SYSTEM DESIGNER	27
2.8.1 Užívateľské rozhranie	27
2.8.2 Modulárnosť	28
2.8.3 Zhodnotenie	29
2.9 DIGI CREATOR	29
2.9.1 Systémové požiadavky.....	29
2.9.2 Spustenie programu Digi Creator	30
2.9.3 Architektúra systému.....	31
2.9.4 Modulárnosť produktu a používané rozhranie	33
2.9.4.1 Rozhrania IPlugin a IPluginHost	33
2.9.4.2 Triedy SavedNode a SavedCon	35
2.9.4.3 Načítanie modulov.....	36
2.9.5 Serializácia	36
2.9.6 Testovanie a používanie produktu Digi Creator.....	38
2.9.6.1 Typ obvodu logický obvod	38
2.9.6.2 Typ obvodu Petriho siete	39
2.9.6.3 Typ obvodu stavové automaty	40
2.9.7 Zhodnotenie produktu Digi Creator	40
2.10 POUŽÍVANÉ SÚBOROVÉ FORMÁTY V DIGITÁLNYCH SYSTÉMOCH.....	41

2.10.1	Súborový formát <i>BLIF</i>	41
2.10.1.1	Logické hradlá	43
2.10.1.2	Nedefinované stavy	43
2.10.1.3	Preklápacie Obvody a zámky	45
2.10.1.4	Knižničné hradlá	45
2.10.2	Súborový formát <i>PLA</i>	47
2.10.3	Súborový formát <i>EQN</i>	48
2.10.4	Súborový formát <i>KISS</i>	49
2.10.5	Súborový formát <i>PNML</i>	50
3	ŠPECIFIKÁCIA POŽIADAVIEK	53
3.1	VŠEOBECNÉ POŽIADAVKY	53
3.2	FUNKCIONÁLNE POŽIADAVKY	54
4	HRUBÝ NÁVRH RIEŠENIA	56
4.1	ARCHITEKTÚRA SYSTÉMU	56
4.2	JADRO	57
4.3	MODULY	58
4.3.1	<i>Textový editor</i>	58
4.3.2	<i>Grafický editor</i>	58
4.4	SIMULÁCIA	59
4.5	POŽIADAVKY NA SYSTÉM	59
5	NÁVRH A IMPLEMENTÁCIA	60
5.1	ARCHITEKTÚRA SYSTÉMU	60
5.2	PREVOD VHDL KÓDU DO GRAFICKÉHO EDITORA	60
5.3	GRAFICKÝ PLUGIN	62
5.3.1	<i>Vykresľovanie hradiel</i>	62
5.3.2	<i>Ukladanie a načítavanie hradiel</i>	63
5.3.3	<i>Riešenie problémov s modulom grafického editora</i>	64
5.3.4	<i>Plugin Blif</i>	64
5.3.5	<i>Načítavanie z Blif formátu</i>	64
5.3.6	<i>Načítavanie do Blif formátu</i>	65
5.4	TEXTOVÝ MODUL	66
5.4	MODUL VERIFIKÁCIE	67
6	TESTOVANIE PRODUKTU	69
7	ZÁVER	71
8	POUŽITÉ SKRATKY	73
9	POUŽITÁ LITERATÚRA	74
10	PRÍLOHY	75
10.1	PRÍLOHA A - POUŽÍVATEĽSKÁ PRÍRUČKA	75

1 Úvod

1.1 Zadanie

Každý zo študentov odboru PKSS sa počas svojho štúdia stretol s viacerými prostrediami pre návrh digitálnych systémov. Jedná sa o rôzne úrovne návrhu od klasických kombinačných logických obvodov, cez použitie Petriho sietí na opis správania systému až po návrh procesorov alebo ASIC obvodov. Základným problémom je nízka podpora programových systémov pri testovaní niektorých spôsobov návrhu a následnej simulácie, poprípade syntézy. Z tohto dôvodu je cieľom vytvoriť prostredie, s ktorým by študenti mohli pracovať na čo najväčšom počte predmetov zameraných na návrh digitálnych systémov a zastrešovalo by všetky metodiky návrhu na rôznych úrovniach. Významné z pohľadu výskumu na našej fakulte je rovnako aj vytvorenie prostredia pre testovanie jednotlivých skúmaných metód (v rámci ústavu, fakulty, SR,...).

Niektoré z hlavných cieľov projektu:

- Základ pre modulárny aplikačný systém umožňujúci prácu s čo najväčším množstvom metodík návrhu.
- Umožniť dodatočné pridávanie nových metodík.
- Podporovať súborové štandardy - BLIF, KISS, SLIF, atd.
- Zahnutý grafický editor pre klasické hradlové obvody, Petriho siete, Konečné stavové automaty, prípadne iné grafické modely používané pri návrhu.
- Podpora simulácie daných grafických modelov (príkladom je PIPE pre Petriho siete alebo LOG pre hradlové obvody)

Je nutné podrobne zanalyzovať súvisiacu problematiku a implementovať prostredie spolu s grafickým editorom, do ktorého by bolo možné pridávať jednotlivé metodiky návrhu a podporované modely. Cieľom Tímového projektu je nadviazanie na minuloročné tímové projekty, vybrať najvhodnejšie časti z oboch projektov a projekt ďalej rozvíjať.

1.2 Účel a rozsah dokumentu

V súčasnosti pri vývoji systémov na čipe je nutné vytvárať prostredia pre ich návrh. Aj keď riešení je v súčasnosti väčšie množstvo, zatiaľ neexistuje postačujúci modulárny systém, ktorý by pokryl všetky požiadavky návrhu. Táto práca bude pojednávať o problematike návrhu časti systému, ktorého základ bol už vytvorený v rámci predmetu Tímový projekt v roku 2010/2011. Cieľom dokumentu, ako aj jeho účelom bude opravenie nedokonalostí predchádzajúceho systému, pričom sa bude orientovať hlavne na vytvorenie modulu pre testovanie návrhu a preklad návrhu medzi grafickou a textovou podobou. Dokument je zložený z troch častí, a to analýza, špecifikácia a návrh.

V analytickej časti dokument pojednáva o rôznych existujúcich systémoch s podobnou tematikou, pričom opisuje funkcionality týchto systémov. Nemožno zabudnúť ani na analýzu projektov z minulých rokov, v ktorých práci sme sa rozhodli pokračovať. Ďalej sa venuje problematike súborových formátov z oblasti logických systémov, ktoré bližšie analyzuje.

V ďalšej časti sa dokument zaoberá špecifikáciou projektu, v ktorej popisuje základné požiadavky na vytváraný systém a následne hrubým návrhom systému na základe špecifikácie požiadaviek.

2 Analýza problému

V našej analýze sme sa venovali predovšetkým produktu Digi Creator, ktorý bol vytvorený tímom č. 2 v akademickom roku 2010/2011 pod vedením Ing. Petra Pišteka, ako aj iným obdobným produktom pre návrh digitálnych systémov.

2.1 Modelsim

ModelSim od spoločnosti Mentor Graphics je veľmi silný a kvalitný nástroj pre tvorbu a testovanie logických obvodov. Obsahuje inteligentné používateľské grafické rozhranie s rozsiahlym počtom nastavení. Najnovšia verzia má označenia 10.0c a je určená pre operačný systém Windows XP, Vista a 7.

ModelSim spája jadro simulátora s ladiacim prostredím pre programovacie jazyky Verilog, VHDL alebo SystemC. Kombináciou návrhu, analýzy a ladenia je tento softvérový produkt vhodný pre návrh ASIC (application-specific integrated circuit) obvodov, ktoré sa vyrábajú pre určitú špecifickú aplikáciu ale aj FPGA (Field programmable gate array) obvody, ktoré predstavujú programovateľnú digitálnu logiku. ModelSim tiež podporuje všetky ASIC a FPGA knižnice a zabezpečuje tak presné načasovanie simulácie.

Jadro simulátora povoľuje transparentné miešanie jazyka VHDL a Verilog v jednom dizajne. Jeho architektúra umožňuje platforme nezávislú kompiláciu s výborným výkonom natívne kompilovaného kódu [1].

2.1.1 Používateľské grafické rozhranie

Inteligentne navrhnuté používateľské grafické rozhranie umožňuje efektívne využitie pracovnej plochy. Intuitívne usporiadanie interaktívnych grafických prvkov (okná, panely nástrojov, ponuky, atď.) uľahčuje zobrazenie a prístup k mnohým výkonným funkciám v ModelSime. Výsledkom je funkčne bohaté grafické rozhranie, ktoré sa pomerne ľahko používa a rýchlo ovláda.

Používateľské grafické rozhranie umožňuje rýchlo identifikovať a ladiť problémy, ktoré vznikli pri návrhu za pomoci dynamicky aktualizovaných okien.

Napríklad výber dizajnu v okne štruktúry automaticky aktualizuje zdrojový kód, signály, procesy a premenné. Takéto prepojenie prostriedkov je veľmi dobré pri ladení prostredia. Pri nájdení problému, je možné problém hneď upraviť, skompilovať a simulovať bez toho, aby sme opustili simulátor.

ModelSim šikovne integruje v sebe užívateľské rozhranie Tcl do svojho HDL simulátora. Tcl je jednoduchý, ale výkonný skriptovací jazyk pre riadenie a rozširovanie aplikácií.

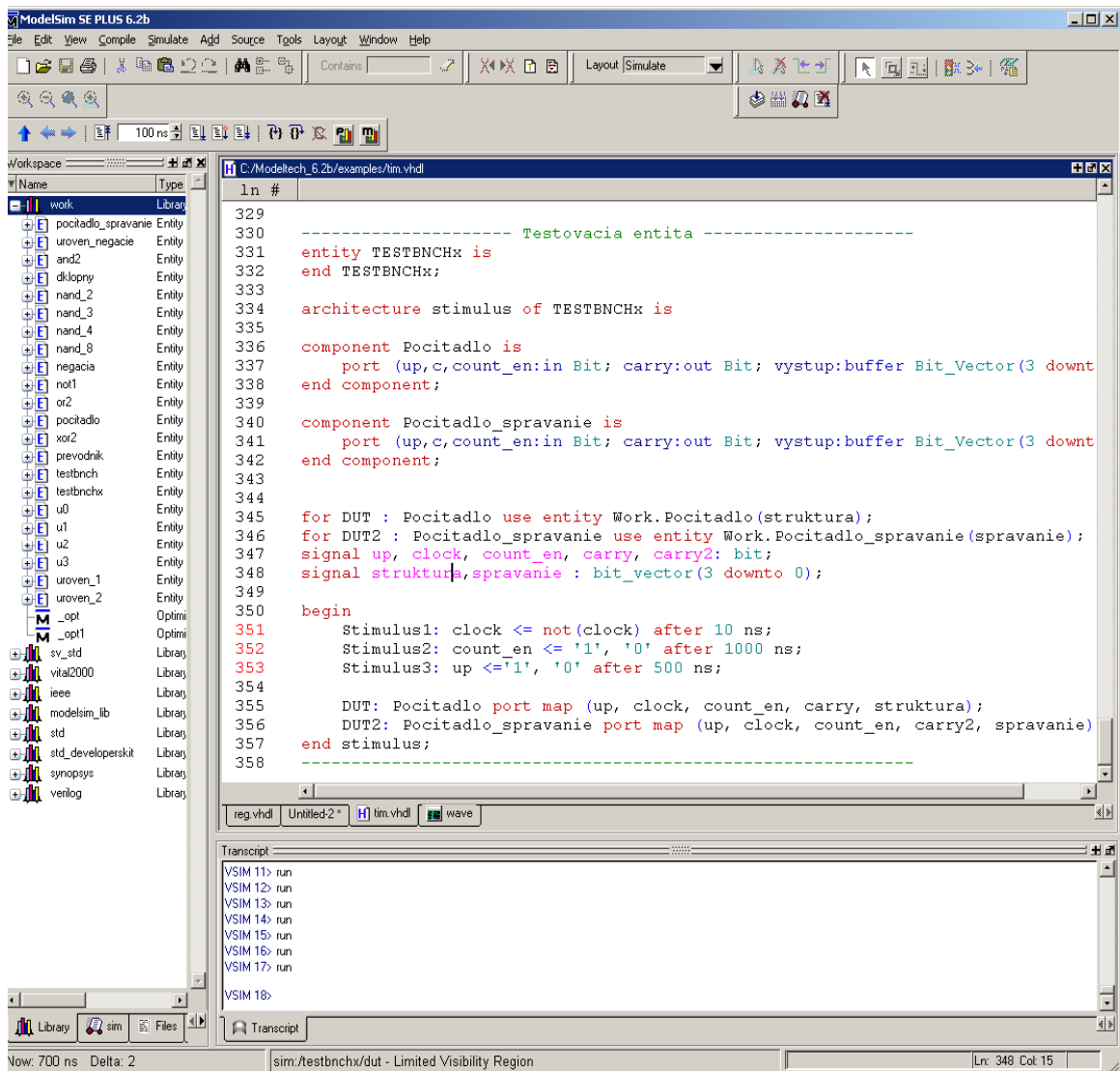
2.1.2 Manažment projektu

Integrovaný manažment projektu výrazne znižuje čas potrebný pre organizáciu súborov a knižníc. Po kompilácii a simulovaní si manažment projektu ukladá jedinečné nastavenia každého projektu, čo nám umožní znovu spustiť simulátor, presne tam kde sme prestali. Prípadne znovu simulovať pomocou vopred nastavených parametrov.

V rozličných oknách môžeme vidieť zoradené samostatne použité komponenty, signály, vstupy a výstupy logického obvodu, čo sprehľadňuje prácu s návrhom a simuláciou.

2.1.3 Textový editor

Textový editor je pomerne jednoduchý a prakticky navrhnutý. Syntax je farebne zvýraznená a odlíšená. Po skompilovaní zdrojového kódu a prípadnej chyby, je označený riadok, v ktorom sa nachádza chyba, čo je veľmi praktické. I keď nápoveda k chybe by mohla byť lepšia, pretože nie vždy vypíše správny dôvod chyby.



Obr. 1 Okno s textovým editorom a Tcl príkazovým riadkom

2.1.4 Šablóny zdrojového kódu a sprievodcovia

VHDL a Verilog šablóny a sprievodcovia umožňujú rýchly vývoj HDL kódu bez nutnosti pamätať si presné syntaxe jazyka. Všetky jazykové konštrukcie sú k dispozícii kliknutím myši. Ľahko použiteľný sprievodca krok za krokom prevedie tvorbu zložitejších HDL blokov.

Sprievodca vie ukázať, ako vytvoriť parametrizované logické bloky, testbench a dizajn objektov. Šablóny zdrojového kódu a sprievodcovia sú prínosom pre začiatočníkov aj pre pokročilých HDL vývojárov.

2.1.5 Testbench

Testbench tvorí samostatnú entitu, ktorá má za cieľ odsimulovať navrhnutý obvod s použitými komponentmi opisom správania sa alebo opisom štruktúry pri generovaní nami zvolených vstupných signálov.

Architektúra testbench obsahuje názov komponentov, ktoré ideme simulovať. Každý komponent má priradené vstupné a výstupné porty. Na vstupné signály si definujeme hodnoty 0 alebo 1 a ich časovanie. podľa ktorých sa bude obvod správať. Nami vytvorený testbench si môžeme ľahko meniť a podľa rôznych vstupov sledovať výstupy simulovaného logického obvodu.

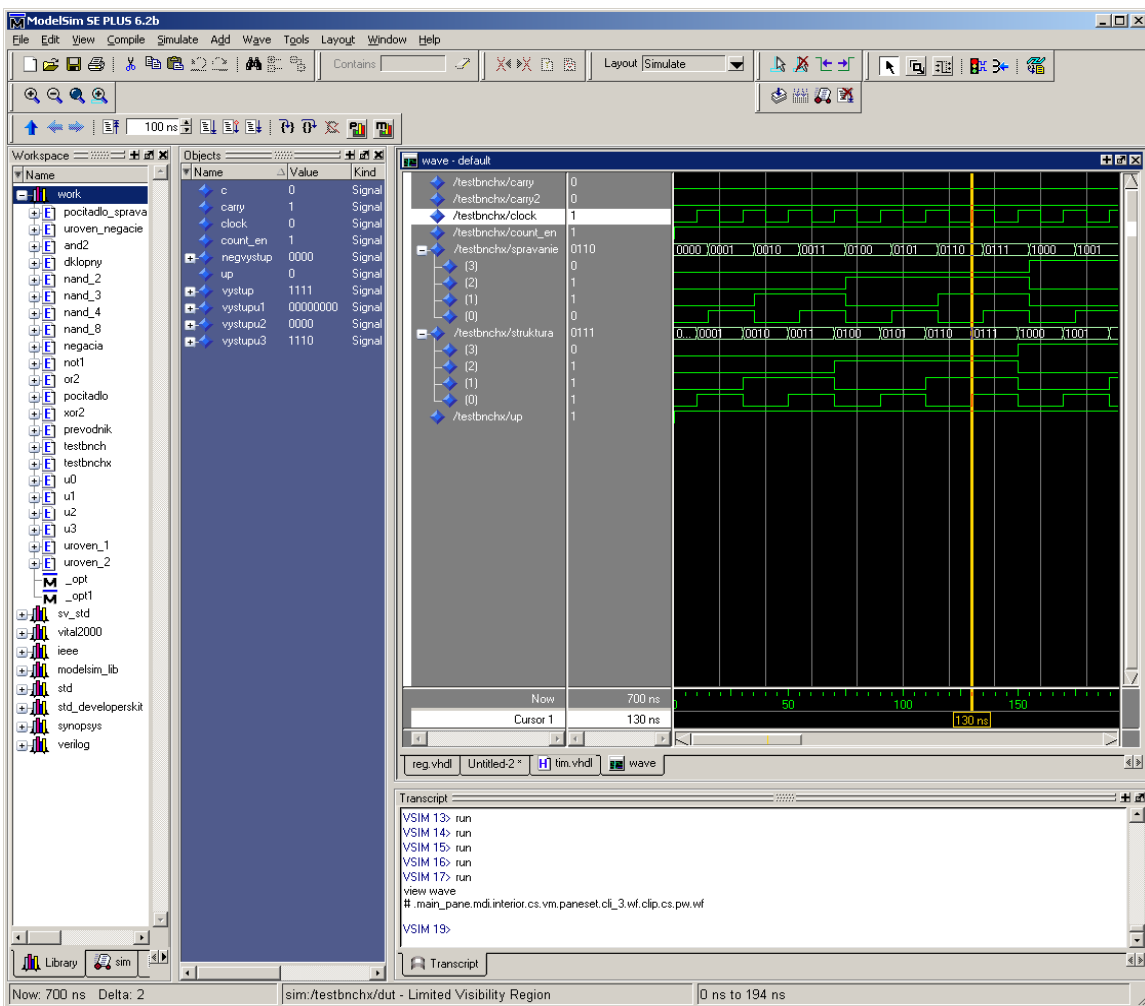
Nasledujúci príklad testovacej entity ukazuje spôsob testovania počítadla od 0 do 15 zhora/nadol.

```
----- Testovacia entita -----
entity TESTBNCHx is
end TESTBNCHx;
architecture stimulus of TESTBNCHx is
component Pocitadlo is
port (up,c,count_en:in Bit; carry:out Bit; vystup:buffer Bit_Vector(3 downto 0));
end component;
component Pocitadlo_spravanie is
port (up,c,count_en:in Bit; carry:out Bit; vystup:buffer Bit_Vector(3 downto 0));
end component;
for DUT : Pocitadlo use entity Work.Pocitadlo(struktura); // testovanie štruktúry
for DUT2 : Pocitadlo_spravanie use entity Work.Pocitadlo_spravanie(spravanie);
// testovanie správania
signal up, clock, count_en, carry, carry2: bit;
signal struktura,spravanie : bit_vector(3 downto 0);
begin
Stimulus1: clock <= not(clock) after 10 ns; // vstupné hodnoty a časovania
Stimulus2: count_en <= '1', '0' after 1000 ns;
Stimulus3: up <= '1', '0' after 500 ns;

DUT: Pocitadlo port map (up, clock, count_en, carry, struktura); //mapovanie portov
DUT2: Pocitadlo_spravanie port map (up, clock, count_en, carry2, spravanie);
end stimulus;
```

2.1.6 Simulácia

Po skompilovaní zdrojového kódu je možné obvod simulovať, pokiaľ máme napísaný a skompilovaný aj testbench. Z okna Objects si presunieme do okna Wave signály, ktoré chceme v našej simulácii sledovať. Simulácia opisom správania a štruktúry prebieha v tomto okne Wave, kde sú prehľadne zobrazené priebehy signálov nami zadaných alebo generovaných vstupoch a tiež vidíme aké sú výstupy obvodu.



Obr. 2 Okno s priebehom signálov (WAVE), objektmi simulácie, komponentmi návrhu a Tcl príkazovým riadkom

2.1.7 Výhody a nevýhody ModelSimu

Výhody ModelSimu sú nasledovné:

- Podpora viacerých programovacích jazykov ako VHDL, SystemC a Verilog.
- Samostatné panely pre určený druh zobrazení alebo editácie (textový editor, Wave, Objects, Process, Transcript, Workspace...).
- Integrovaný manažment projektu, ktorý zjednodušuje a sprehl'adňuje správu dát projektu.
- Šablóny zdrojového kódu.
- Integrovaný sprievodca.
- Podpora ASIC a FPGA knižníc.
- Uložený zdrojový kód v súbore s príponou VHDL je možné otvoriť obyčajným textovým editorom a upraviť ho.
- Technická podpora výrobcu.

Nevýhody ModelSimu sú nasledovné:

- Grafické a používateľské rozhranie sa snaží byť intuitívne, no niektoré funkcie nie sú tam, kde by ich používateľ hľadal. Na jednej strane je výhodné mať množstvo nastavení, no ich veľký počet komplikuje prehľadnosť programu.
- Nestabilita programu. Program niekedy pri kompilovaní zamrzol, prípadne sa neočakávane ukončil.
- Problémy s kompiláciou toho istého projektu na 2 rôznych PC, kedy projekt nešiel skompilovať, prípadne vypisoval nové chyby.
- Problémy s prácou knižnice. V niektorých prípadoch pri zmene projektu bolo potrebné vymazať starú knižnicu a načítať novú, lenže manuálne niekedy knižnica nešla vymazať. V tomto prípade pomohol reštart ModelSimu.
- Chýbajúce niektoré všeobecne známe klávesové skratky, ktoré by uľahčili prácu s textovým editorom (napríklad Ctrl – A a podobne)

2.2 KLogic

KLogic je aplikácia pre vytváranie a simuláciu logických obvodov pod operačným systémom Linux [2].

2.2.1 Používateľské grafické rozhranie

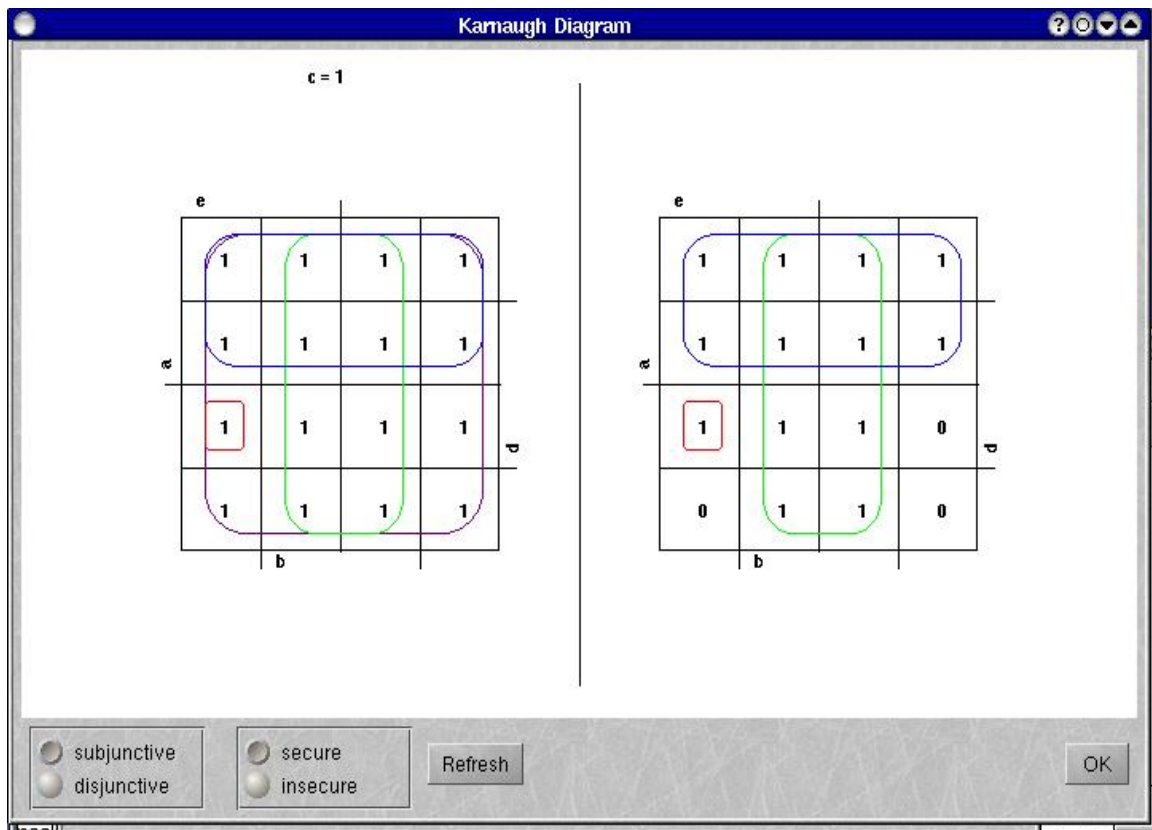
Ovládanie nástroja je jednoduché a intuitívne. Program ponúka sadu jednotlivých základných komponentov ako AND, OR, XOR a preklápacie obvody JK a DK, za účelom vybudovania zložitejších a štruktúrovanejších obvodov. Vyhľadávanie komponentov je intuitívne s možnosťou pomenovania a nastavenia prislúchajúcich parametrov k danému komponentu [3].

2.2.2 Návrh logického obvodu

Budovanie logického obvodu je uskutočnené jednoduchým presúvaním komponentov z panela ponuky na prázdnu plochu s mriežkou. Logickú hodnotu 1 symbolizuje červená farba logickú nulu farba modrá. Čierna farba sa používa pri nedefinovanom stave, väčšinou keď kábel nie je zapojený. Výnimku tvorí komponent, ktorý môže nadobúdať 3 stavy a vtedy je čierna farba platný stav.

2.2.3 Simulácia

Simulácia beží permanentne pri návrhu vlastného obvodu. Krokovanie simulácie umožňuje analyzovať správanie logických hodnôt v rôznych častiach obvodu a v rôznom čase. K dispozícii sú aj Karnaughove grafické metódy analýzy pre účely zjednodušenia funkcií.

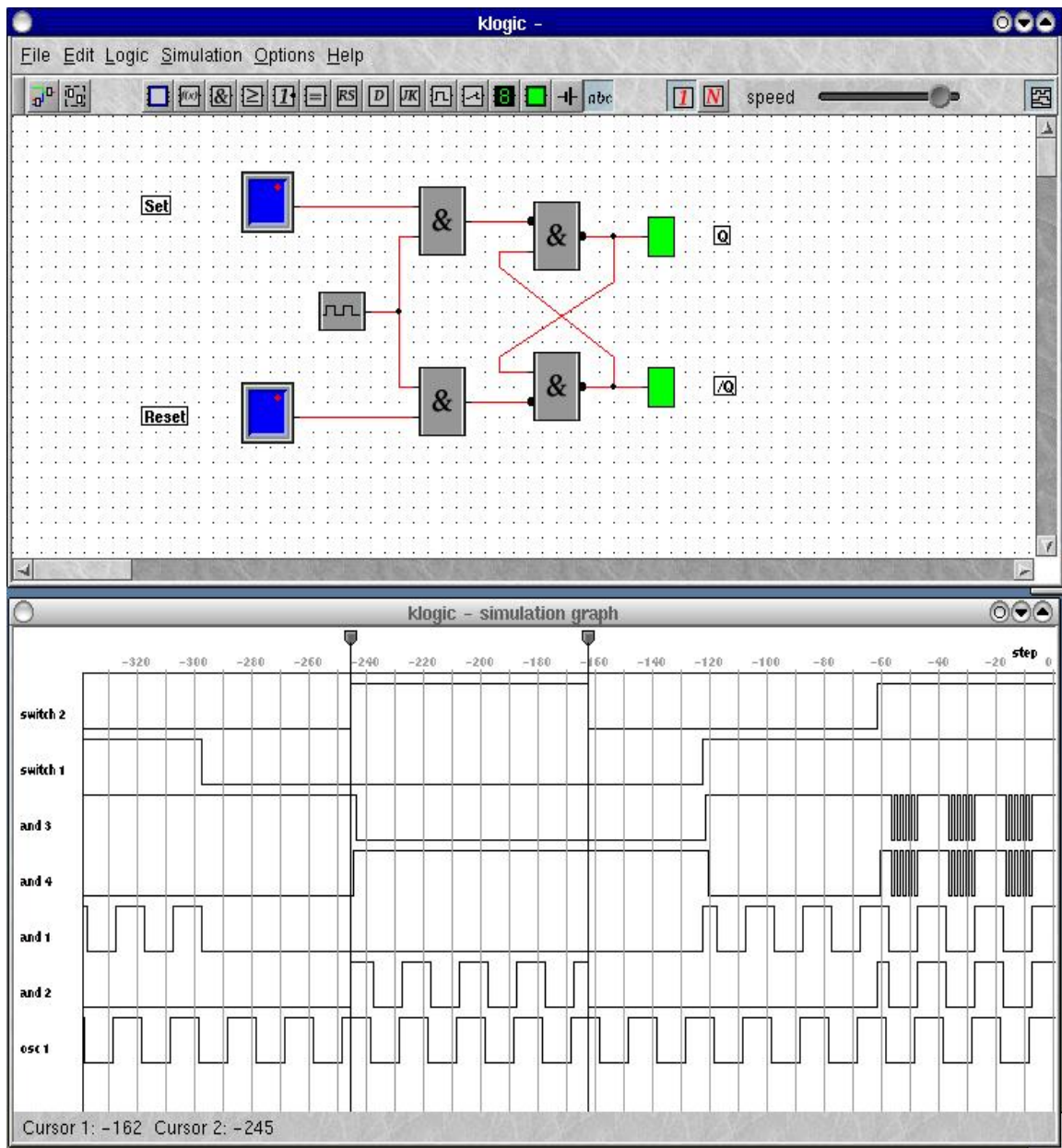


Obr. 3 Karnaughove mapy pre zjednodušenie funkcií

Tok signálov komponentov je možné zobrazit' vo forme grafu. Každé zariadenie v obvode má možnosť nastavenia oneskorenia.

V programe sa dá nastaviť rýchlosť simulácie, môžeme meniť čas medzi každým krokom simulácie.

Projekt je možné uložiť do XML formátu.



Obr. 4 Príklad jednoduchého obvodu s jeho grafom simulácie

2.2.4 Výhody a nevýhody produktu KLogic

Výhody KLogic:

- Jednoduché a intuitívne používateľské rozhranie.
- Jednoduchá tvorba a simulácia vytvoreného obvodu.
- Voľne šíriteľný program.

Nevýhody KLogic

- Obmedzené funkcie programu.
- Nedostatočný počet zložitejších komponentov.
- Verzia len pre Linux.
- Nemožnosť naprogramovať si vlastnú súčiastku.
- Program nie je určený pre logické obvody s rozsiahlou štruktúrou a viacerými hierarchickými úrovňami

2.3 TkGate

TkGate je grafický editor a simulátor logických obvodov napísaný z veľkej časti v jazyku C s užívateľským rozhraním vytvoreným pomocou Tcl/Tk. Program beží na operačnom systéme FreeBSD, Linux, Solaris alebo Mac OS X. Pod operačným systémom Windows sa dá spustiť v emulátore Cygwin [5].

2.3.1 Používateľské grafické rozhranie

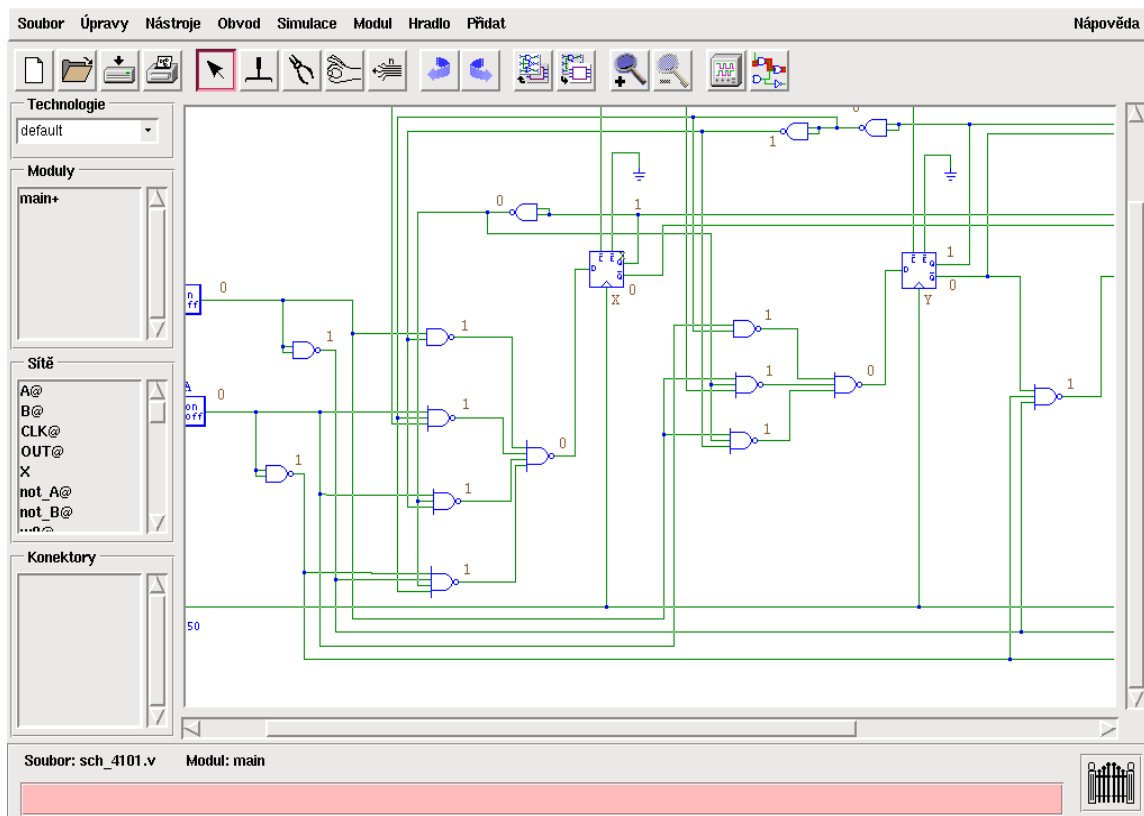
Pomocou menu je možné na plochu vkladať nové komponenty od základných hradiel a spínačov, LED displeje a obvody vyššej integrity ako multiplexory, sčítačky, pamäte a terminály. Používateľské rozhranie je navrhnuté logicky a zrozumiteľne. Jeho používanie je veľmi intuitívne.

2.3.2 Návrh logického obvodu

Logický obvod môže mať hierarchickú štruktúru až po moduly definované užívateľom.

Editovací režim má viacero funkcií, kde sa dajú prerušiť určené vodiče alebo pridať k existujúcim hradlám invertory.

Rozmiestnenie súčiastok na plochu je možné pohodlne meniť štýlom „drag and drop“. Príjemnou vlastnosťou je, že pri týchto úpravách sa vodiče v okolí posunutého hradla dokážu inteligentne usporiadať tak, že sa pridajú potrebné segmenty alebo sa zmení ich zoskupenie v okolí vodičov [4].



Obr. 5 Příklad návrhu logického obvodu

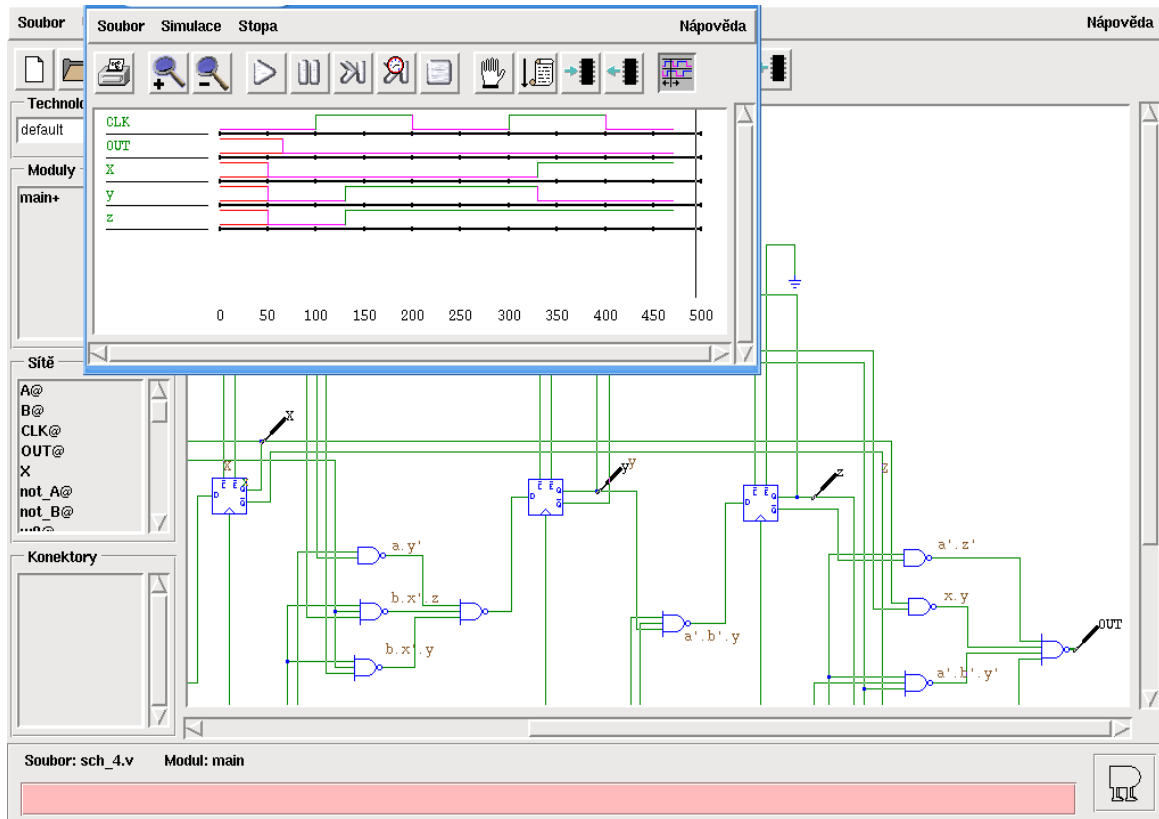
Kliknutím pravého tlačidla na ktorýkoľvek prvok obvodu, je možné upravovať jeho vlastnosti. U hradiel je možné zvoliť výrobnú technológiu (CMOS, TTL) na čom závisí oneskorenie, pridávať vstupné a výstupné konektory, u pamätí sa týmto spôsobom dá nahrať ich obsah z externého súboru, u vodičov je možné prispôbiť ich bitovú šírku. Všetky tieto úpravy je možné aplikovať jednotlivo alebo aj na skupiny prvkov, čo je veľmi praktické a šetrí to čas.

Výhodou programu je aj vlastná tvorba modulov, ktoré nie sú preddefinované. Takto si môžeme navrhnuť ľubovoľný model a hocikedy ho použiť. S takýmito modelmi môžeme vytvárať hierarchickú štruktúru obvodu.

2.3.3 Simulácia

Simulovať je možné kombinačné aj sekvenčné obvody. Simuláciu spustíme výberom príslušnej položky z menu alebo kliknutím na tlačidlo v panelu nástrojov. Objaví sa prázdne okno, ktoré môžeme postupne zaplniť stopami. Každá táto stopa patrí

jednej zo sond, ktoré sme umiestnili na požadované miesta v obvode kliknutím na daný vodič a v priebehu simulácie sa zobrazuje jeho aktuálna logická hodnota.



Obr. 6 Príklad simulácie logického obvodu

Simuláciu je možné nechať bežať nepretržite alebo ju odkrokovávať. Taktiež je možné posunúť simuláciu o pevne stanovený počet jednotiek času alebo hodinových cyklov. Program podporuje aj nastavenie breakpointov, ktoré zastavia simuláciu v tej chvíli kedy premenná je nastavená na určitú hodnotu.

2.3.4 Výhody a nevýhody produktu TkGate

Výhody TkGate sú nasledovné:

- Dobře zvládnuté grafické používateľské rozhranie.
- Jednoduchý a efektívny spôsob návrhu logických obvodov.
- Možnosť navrhovať a uložiť vlastné moduly s vnútornou logickou štruktúrou.

- Možnosť zdefinovať vlastnú technológiu hradíel.
- Možnosť spustiť a nastaviť simuláciu pomocou skriptu.
- Dobrá analýza citlivých častí obvodu.
- Možnosť tlače schém a výstupov simulácie.
- Možnosť uloženia schém a simulácií do proprietárneho súborového formátu .ps
- Viacjazyčné rozhranie.
- Ovládanie cez GUI alebo pomocou skriptu.
- Ukladanie súboru do Verilog formátu.
- Voľne šíriteľný program
- Veľmi dobrá podpora klávesových skratiek

Nevýhody produktu TkGate sú nasledovné:

- Program sa nedá spustiť natívne pod operačným systémom Windows.
- Preklápacie obvody nie sú k dispozícii ako preddefinované komponenty a je nutné si ich manuálne vytvoriť ako modul.
- Nie je k dispozícii grafické rozlíšenie vodičov s hodnotou 0 alebo 1.

2.4 LOG

LOG je starší program na návrh a simuláciu logických obvodov. Jeho tvorba siaha do začiatkov 90- tých rokov. Tým je podmienený aj jeho jednoduchý dizajn.

2.4.1 Používateľské grafické rozhranie

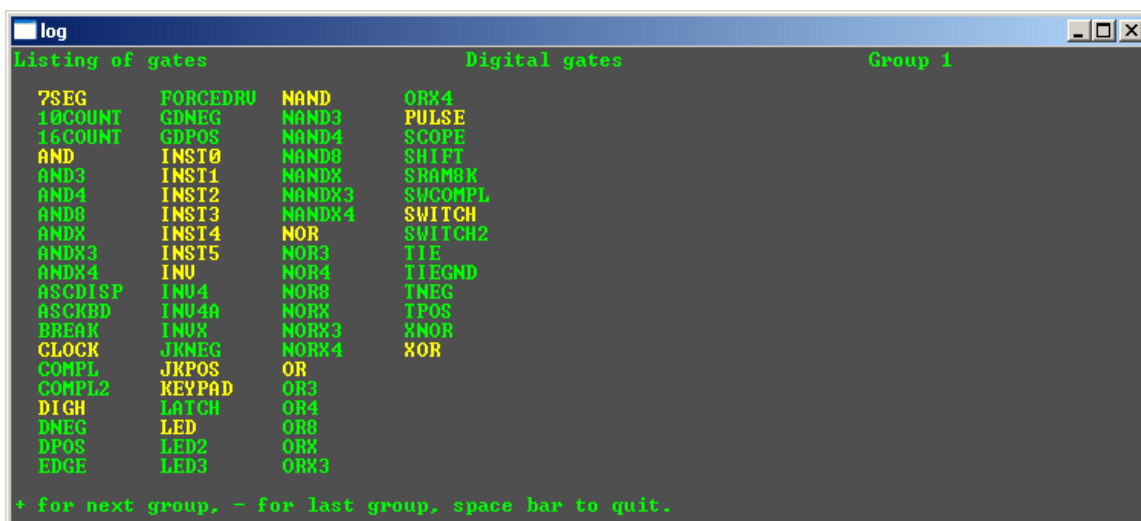
Používateľské rozhranie nie je moc prívetivé aj keď základné dvoj vstupové hradlá ako AND, OR, NAND, NOR, INVERTOR a súčiastka vstupu a výstupu sú k dispozícii hneď v dolnej časti obrazovky. No v prípade použitia iných súčiastok je

nutné vstúpiť do knižnice a nájsť požadovanú súčiastku, čo nie je vždy jednoduché, pretože knižnica je naozaj rozsiahla.

2.4.2 Návrh logického obvodu

Návrh logického obvodu sa robí vkladaním súčiastok z dolnej lišty a spájaním pomocou vodičov. Nepraktické je hlavne nemožnosť posúvať súčiastky zo zachovaním vodičov. V prípade posunu súčiastky, vodič zostane na pôvodnom mieste bez zapojenia. Editácia už zhotoveného obvodu je taktiež problematická.

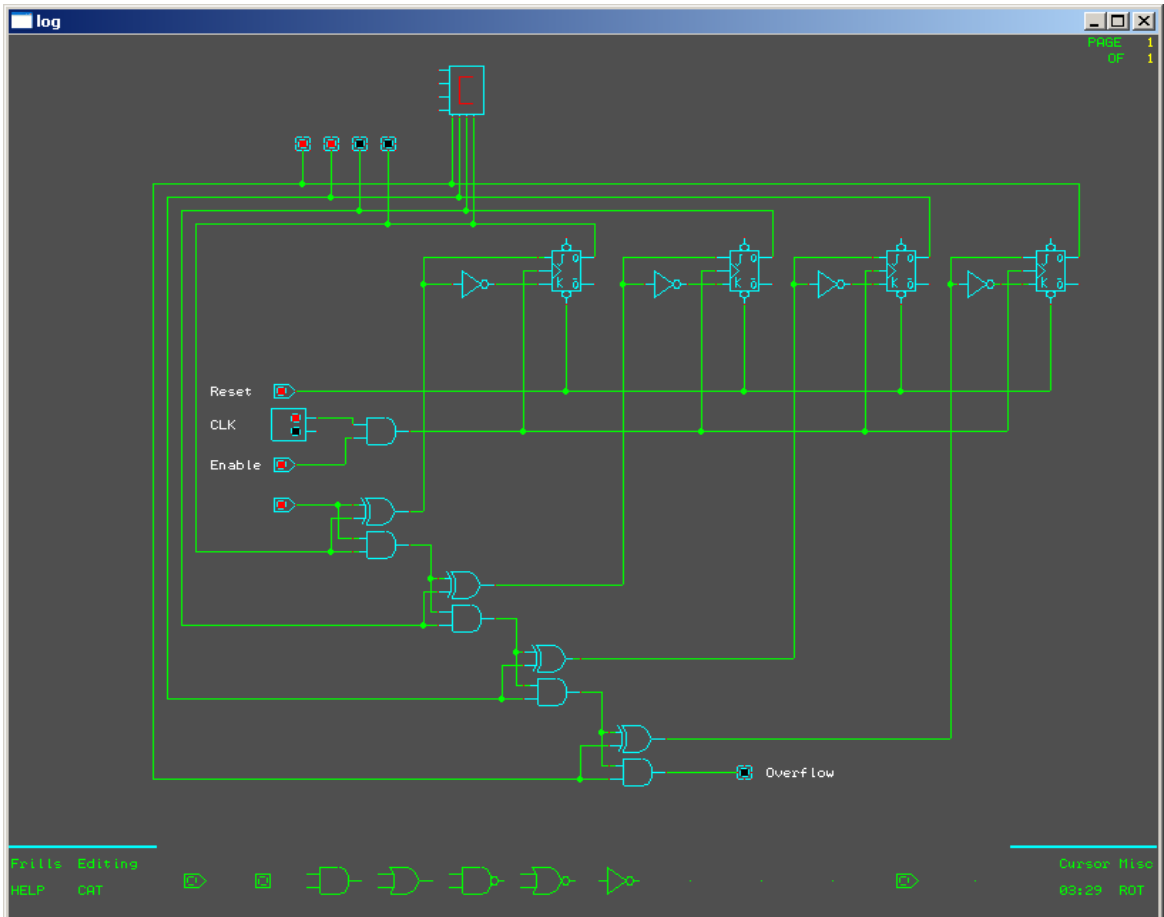
Knižnica obsahuje množstvo súčiastok ako viac vstupové hradlá, multiplexory, displeje, časovače, generátory a iné, čo považujem za pozitívne.



Obr. 7 Výber prvkov z knižnice skupiny 1 z 8

2.4.3 Simulácia

Simulácia sa uskutočňuje neustále aj pri tvorbe obvodu, takže hneď pri zapojení súčiastky vidíme výsledok simulácie.



Obr. 8 Počítadlo navrhnuté a simulované v LOGu

2.4.4 Výhody a nevýhody produktu LOG

Výhody produktu LOG sú nasledovné:

- Nemusí sa inštalovať.
- Program zaberá len niečo okolo 1 MB.
- Kolízne situácie obvodu sa automaticky vyznačia červenou farbou
- Na jednoduché obvody je celkom vhodný

Nevýhody produktu LOG sú nasledovné:

- Neintuitívne grafické rozhranie.
- Zlá editácia logického obvodu.

- Pri spustení programu je permanentne vytážené jedno fyzické jadro procesora takmer na 100 %.
- Slabá podpora klávesových skratiek
- Obmedzený počet vstupov logického obvodu
- Zložitá konfigurácia súčiastok
- Nemožnosť exportu obvodu do obrázkového formátu
- Podpora len vlastného súborového formátu .lgf.

2.5 MVSIS

MVSIS je experimentálny softvér pre optimalizáciu a syntézu sekvenčných obvodov s viac hodnotovou logikou. Jedná sa o sieť uzlov, kde každý uzol predstavuje viac hodnotovú reláciu. Nástroj MVSIS je nástupcom pôvodného systému SIS, ktorý sa už v súčasnosti nepoužíva, kvôli staršej technológii.

Nástroj MVSIS bol vyvinutí tak ako SIS v Berkeley v Spojených štátoch amerických. Skupina vývojárov MVSIS študuje logickú syntézu a verifikáciu návrhu technológie VLSI (Very-large-scale integration), čo je jedna z generácií výroby polovodičových čipov s veľkou integráciou polovodičových prvkov na jednom čipe. Hlavný dôraz MVSIS je kladený na nové optimalizačné algoritmy, ktoré zlepšia kvalitu obvodov, ktoré sú generované pomocou automatických nástrojov na syntézu a zároveň budú škálovateľné aj pre praktické využitie.

Väčšina algoritmov vyvinuté touto skupinou vývojárov sú zahrnuté v open-source programe MVSIS aj keď počiatočným cieľom MVSIS bola logická minimalizácia pre siete s viac hodnotovou logikou, ale postupom času sa vyvinula v plnohodnotný nástroj pre syntézu a overovanie všeobecne [6].

2.5.1 Viac hodnotová logická syntéza

Viac hodnotová logická syntéza môže mať mnoho aplikácií:

- Logická syntéza pre viac hodnotové hardvérové zariadenia s prúdovým módom.

- Počiatočná manipulácia s popisom hardvéru predtým, ako je zakódovaný do binárnej podoby a spracovaný štandardným programom logickej syntézy.
- Softvér prirodzene podporuje vyhodnotenie viac hodnotových premenných v jednom cykle. Pre vložené aplikácie špecifikované zo synchronných jazykov s konečným stavom, môžu byť logické syntézy aplikované na optimalizáciu riadenia toku.
- Asynchrónna syntéza oneskorení necitlivých obvodov (DI).
- Zber dát, kde cieľom je jednoduchý popis, ktorý zahŕňa obsah niektorých údajov.

Logika MVSIS siete je vlastne rozšírenie tradičnej boolean siete, ktorá zahŕňa viac hodnotové uzly, každý s vlastným rozsahom. MVSIS obsahuje všetky technológie nezávisle transformované zo SIS pre syntézu kombinačných logických obvodov z binárnej do viac hodnotovej logiky.

Na minimalizáciu uzla sa používa BDD (binary decision diagram), čiže binárny rozhodovací diagram, ktorý zvládne aj väčšie dvojúrovňové funkcie. Algoritmy MVSIS sú použité na konverziu z viac hodnotovej logiky do binárnej logiky a opačne. Taktiež vedia odhadnúť optimalizáciu vo viachodnotovej oblasti, ešte predtým ako nastane konverzia do binárnej logiky.

Viac hodnotový obvod sa do programu MVSIS nahráva ako súborový formát BLIF-MV. Takýto súbor môže byť vygenerovaný programom Verilog. Binárne obvody sú ukladané v BLIF formáte a môžu byť taktiež nahraté do MVSIS. Po načítaní súboru do MVSIS, je návrh siete prekonvertovaný do viachodnotovej reprezentácie. MVSIS obsahuje príkazy na čítanie, zápis do alebo zo súboru a zobrazovanie súboru. Príkazy na čítanie rozoznávajú rozdielnosť formátov BLIF a BLIF-MV. Každá výstupná hodnota vo MV (multi-value) funkcii sa nazýva i-set.

MVSIS pozná tieto skupiny príkazov: základné, čítacie, zapisujúce, tlačové, časovacie, mapujúce, menujúce, overujúce, vykonávajúce kombinačnú syntézu, vykonávajúce sekvenčnú syntézu.

```

C:\Documents and Settings\Termi\Desktop\FIIT\Timovy projekt\mvsis-3.0-win32\mvsis50720.exe
STG-based sequential synthesis commands:
binary      check      check_nd    complement
complete    dcmn      determinize dot
minimize     moore     prefix      print_support
product      progressive psa        read_automaton
remove_dc    show      support      test
trim         volume    write_automaton

Sequential synthesis commands:
extract_seq_dc io_encode  latch_expose latch_split
net_product    solve     stg_extract

Synthesis from L language commands:
dtest         dual       formula     synth

Various commands:
lhotize       assert     default     dize
frames        free       miter       reorder
sis           split_outputs window

Verification commands:
fraig_verify  reach     sat_verify  verify

Writing commands:
write_bench   write_blif write_blif_mv write_blif_mvs
write_gate    write_pla  write_pla_mv

mvsis 01>

```

Obr. 9 Konzolové prostredie nástroja MVSIS po zadaní príkazu help

2.6 CUDD

CUDD (Colorado University Decision Diagram) je balík funkcií na manipuláciu s rozhodovacími diagramami (Binary Decision Diagram - BDD, Algebraic Decision Diagram - ADD, Zero-suppressed Binary Decision Diagram - ZBDD) a na ich transformáciu medzi sebou. BDD a ZBDD slúžia na reprezentáciu spínacích funkcií, zatiaľ čo ADD dokáže opísať funkciu z množiny $\{0,1\}^n$ do ľubovoľnej množiny [7].

Balík CUDD je možné použiť tromi rôznymi spôsobmi [7]:

- 'Black box' - vytvorenie aplikácie na prácu s rozhodovacími diagramami s využitím iba existujúcich funkcií z balíka. V tomto prípade nie je nutné riešiť preusporiadanie premenných aplikáciou, ale CUDD to vykoná sám.
- 'Clear box' - doprogramovanie vlastných funkcií, ktoré priamo rekurzívne manipulujú s rozhodovacím diagramom, za účelom zvýšenia optimalizácie.
- Prostredníctvom rozhrania - CUDD je distribuované spolu s C++ rozhraním, ktoré umožňujú rýchle prototypovanie. Tento prístup má tú výhodu, že nie je nutné starať sa o manažment pamäte a preťaženie operátorov.

- Pre CUDD existuje viacero rozšírení ako napr. PerlDD (Perl5 rozhranie), DDcal (pre vykresľovanie grafov), alebo DDDMP (pre kompaktné uloženie diagramu) [8].

2.6.1 Dátové štruktúry CUDD

1. Uzly

Rozhodovacie diagramy sú tvorené uzlami, ktoré sú realizované v CUDD ako štruktúra DdNode. DdNode nesie základné informácie o uzle ako index uzla, počet odkazov na uzol, hodnota uzla a ukazovatele na susedné uzly.

2. Manažér

Štruktúra DdManager, pomocou hašovacej tabuľky uzlov, zabezpečuje jedinečnosť každého uzla a zaisťuje teda kanonickosť rozhodovacieho diagramu. V jednej aplikácii je možné mať naraz viacej aktívnych manažérov.

3. Vyrovnávacia pamäť (cache)

Slúži hlavne na dočasné uchovávanie výsledkov pri rekurzívnych manipuláciach diagramu.

2.6.2 Princíp práce s CUDD

Štruktúra programu je rovnaká ako v C/C++, čiže najprv je nutné zadať závislosti na rôznych knižniciach, potom makrá, konštanty, globálne premenné a rôzne funkcie na prácu s rozhodovacími diagramami, ďalej len RD. Predtým ako začne aplikácia manipulovať s RD je nutné inicializovať manažéra. Potom treba vytvoriť a opísať RD. Keď je vytvorený, môže aplikácia začať použiť rôzne funkcie z balíka na manipuláciu s RD. Po dokončení zadaných operácií nad RD ukončíme manažéra a program ukončíme.

CUDD balík poskytuje viacero funkcií na uloženie RD do súboru. Ukladanie do formátu BLIF je obmedzené len na BDD. BDD, ADD a ZBDD je možné uložiť vo formáte vhodnom na vykreslenie RD pomocou programu dot od Eleftheriosa Koutsofiosa a Stephena C. Northa [9]. BDD a ADD sa dajú takisto uložiť do formátu vhodného na vykreslenie v programe daVinci vyvinutého Univerzitou Brémy [9].

2.7 BDS

Je systém určený na optimalizáciu logických obvodov, pochádzajúci z University of Massachusetts, využívajúci techniku BDD (binary decision diagram – binárnych rozhodovacích diagramov). Štruktúra správne organizovaného binárneho rozhodovacieho diagramu odhaľuje dôležité informácie o jeho funkčnej dekompozícii. V BDD sa dajú nájsť rôzne štruktúry, tiež označované ako dominátory, ktoré vedú k efektívnej dekompozícii AND, OR, XOR a multiplexor [10]. Systém je veľmi efektívny, ale zvládne len obvody, pre ktoré sa dá vytvoriť BDD diagram. Systém je implementovaný v programovacom jazyku C. Vstupom je obvod zapísaný v súborovom formáte BLIF, ktorý sa následne transformuje na boolovskú sieť, rozloží ju do množiny uzlov, zjednoduší pomocou techník BDD a nakoniec dekomponuje.

Existuje aj novšia, vylepšená implementácia pod názvom BDS-PGA. Táto vylepšená verzia pôvodného BDS optimalizuje navrhované obvody pre použitie v FPGA obvodoch.

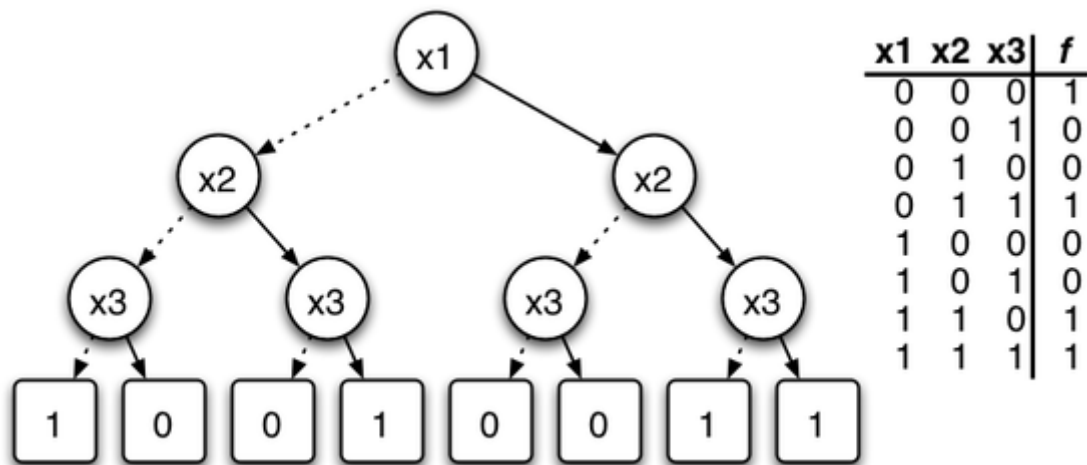
2.7.1 BDD

Binárny rozhodovací diagram je dátová štruktúra využívaná na reprezentáciu Boolovej funkcie, ktorá sa znázorňuje v tvare rozhodovacieho stromu, kde sa každá hrana označí číslom 0 alebo 1. Po prepise funkcie do diagramu sa aplikujú nasledovné operácie:

- Spojenie izomorfných podstromov
- Eliminácia redundantných testov

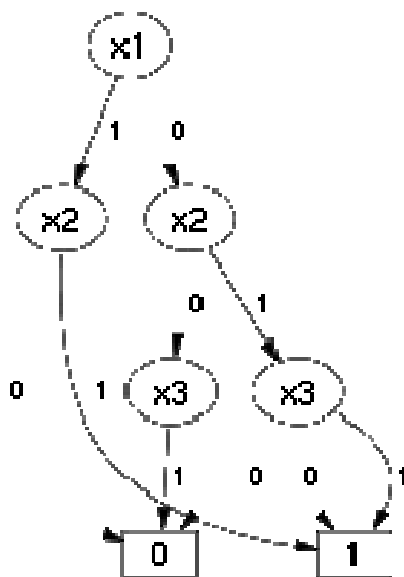
Po týchto úpravách môžeme povedať, že diagram je usporiadaný a redukovaný. Redukované usporiadané binárne rozhodovacie diagramy majú vlastnosť, že sú unikátne pre konkrétnu funkciu.

Príklad: Majme funkciu $f(x_1, x_2, x_3) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_1 \cdot x_2 + x_2 \cdot x_3$. Na nasledujúcom obrázku je vidieť binárny rozhodovací diagram pre túto funkciu.



Obr. 10 Binárny rozhodovací diagram $f(x_1, x_2, x_3) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_1 \cdot x_2 + x_2 \cdot x_3$

Tento diagram nebol zatiaľ upravený. Ak aplikujeme úpravy uvedené vyššie, dostaneme diagram ako na ďalšom obrázku:



Obr. 11 Diagram po úpravách [14]

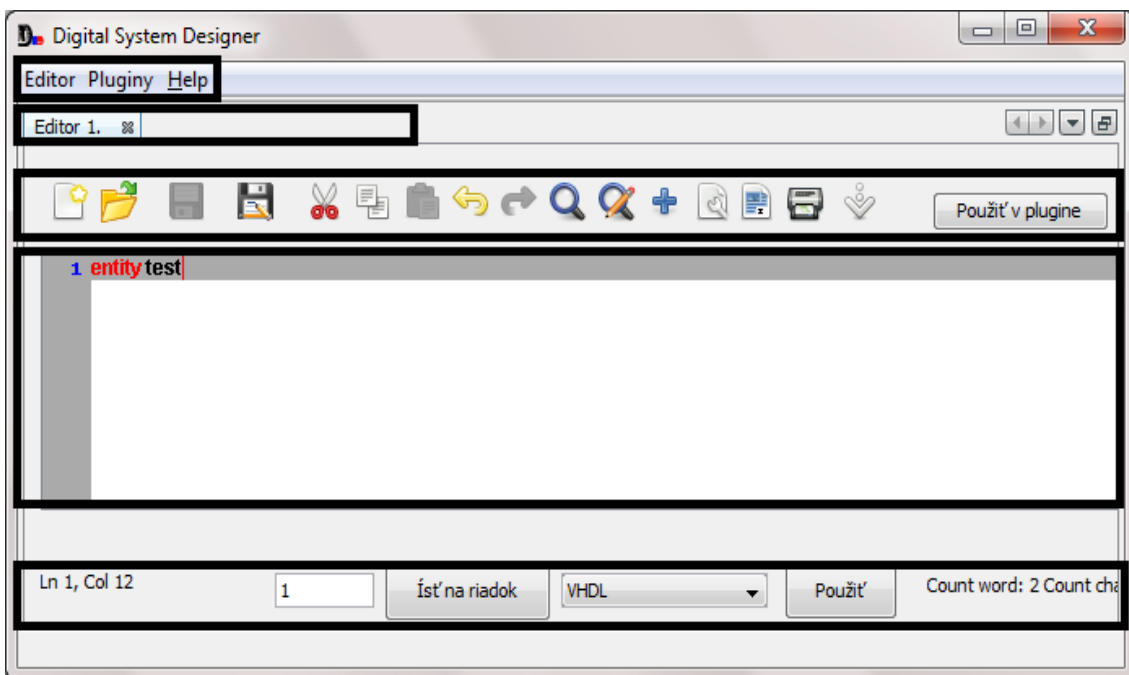
Na tento diagram boli aplikované najmä úpravy eliminácie redundantných uzlov, keďže niektoré uzly dávali rovnakú hodnotu na výstupe a boli teda zbytočné.

2.8 Digital System Designer

Digital system designer je jednoduchý modulárny program na návrh digitálnych systémov, ktorý vznikol ako produkt členov tímu č.10 na predmete Tímový projekt. Systém je kompletne naprogramovaný v Java a je multiplatformový, funguje na operačných systémoch Windows, Linux a MacOS.

2.8.1 Užívateľské rozhranie

Po spustení programu sa otvorí hlavné okno, ktoré je rozdelené na niekoľko častí:



Obr. 12 Hlavné okno Digital System Designer

Hlavné menu je veľmi jednoduché a obsahuje len niekoľko položiek rozdelených do 3 kategórií: Editor, Pluginy a Help. V kategórii Editor je možnosť vytvoriť nový súbor a zavrieť program. Pod Pluginmi sa nachádza možnosť zobrazíť všetky inštalované pluginy, odkiaľ sa dajú ovládať, pridávať nové, spúšťať alebo odoberať. V nápovede sa nachádzajú len informácie o vývojovom tíme.

Lišta so záložkami – obsahuje záložky otvorených okien v programe a je možné sa medzi nimi prepínať.

Lišta s nástrojmi je zrejme najdôležitejším ovládacím prvkom. Obsahuje všetky dostupné nástroje, ktoré by mohol používateľ potrebovať. Dá sa tu nájsť napríklad otváranie a ukladanie súborov, kopírovanie, vkladanie a kopírovanie textu, možnosť kroku späť a dopredu, tlač a ďalšie.

Textový editor zaberá najviac priestoru v hlavnom okne. Ide o klasický textový editor so zvýrazňovaním syntaxe a číslovaním riadkov.

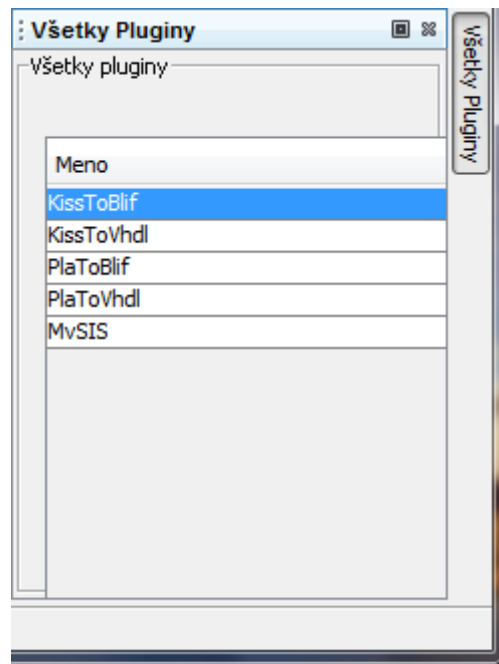
Ovládanie editora v spodnej časti hlavného okna slúži len na rýchly presun na určitú pozíciu a voľba programovacieho jazyka na zvýrazňovanie syntaxe v editore.

2.8.2 Modulárnosť

Modulárnosť je v danom programe zabezpečená využívaním niekoľkých externých programov na realizáciu prevodov medzi súborovými formátmi:

- Kiss to blif – prevádza súbory z formátu KISS na formát BLIF
- Kiss to vhdl – prevádza súbory z formátu KISS na formát VHDL
- Pla to blif – prevádza súbory z formátu PLA na formát BLIF
- Pla to vhdl – prevádza súbory z formátu PLA na formát VHDL
- MVSIS – modul na prácu s programom MVSIS

Všetky tieto moduly sú realizované ako externé programy v jazyku Java a po spustení v programe sa volajú pomocou príkazového riadku so zadanými parametrami. Potrebné parametre sú pre každý modul definované v XML súbore. Samozrejme sa dajú v prípade potreby pridať (dorobiť) ďalšie.



Obr. 13 Okno Všetky Pluginy

2.8.3 Zhodnotenie

Medzi hlavné výhody programu by som určite zaradil textový editor so zvýrazňovaním syntaxe a externé pluginy. Práve tieto funkcie by sme sa chceli pokúsiť prebrať do nášho programu, ktorý budeme implementovať a mierne vylepšiť.

Slabšie stránky analyzovaného programu sú podľa môjho názoru najmä neprehľadné grafické používateľské rozhranie a menu.

2.9 Digi Creator

Táto kapitola bude zameraná na dôkladný opis produktu Digi Creator [11], jeho používanie, možnosti pre ďalšie rozširovanie a zlepšovanie nájdených nedostatkov.

2.9.1 Systémové požiadavky

Produkt Digi Creator kladie na systém len minimálne nároky. Spustenie programu je podmienené operačným systémom Microsoft Windows XP a jeho minimálnymi systémovými požiadavkami, ktoré sú nasledovné:

- Procesor Intel/AMD 300 MHz

- 128 MB RAM
- Monitor podporujúci rozlíšenie 800x600
- Klávesnica a myš
- 100MB voľného miesta na pevnom disku

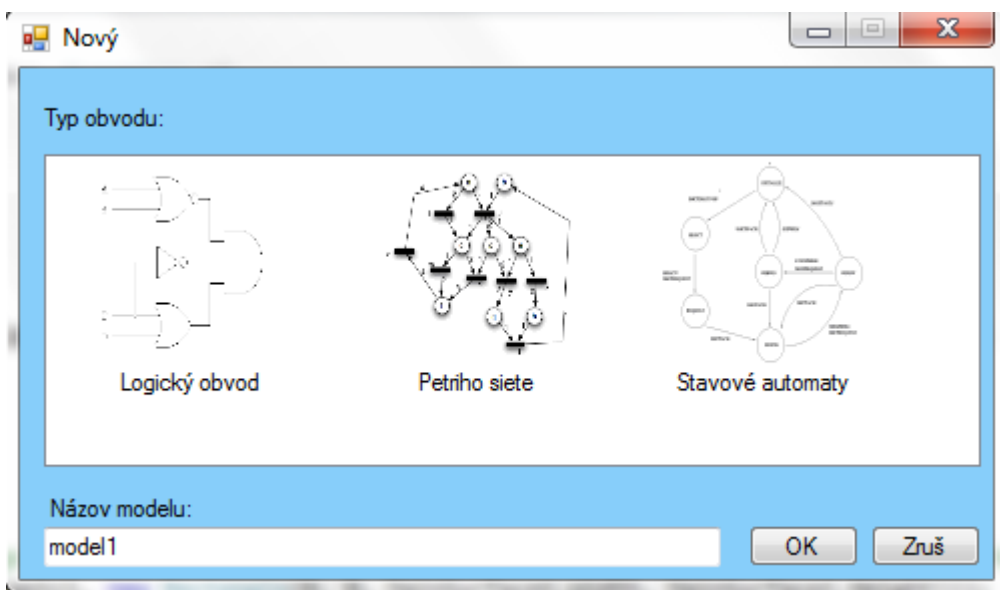
Autori neuvádzajú, či produkt Digi Creator bol testovaný i pod iným operačným systémom z nižšej rady operačných systémov Microsoft Windows alebo pod iným konkurenčným operačným systémom.

Softvérové nároky sú nasledovné:

- Operačný systém Windows XP, Windows Vista, Windows 7
- Platforma .NET 2.0

2.9.2 Spustenie programu Digi Creator

Po skompilovaní a spustení programu Digi Creator má užívateľ v úvodnej zobrazenej ponuke (obr. 14) na výber typ obvodu. Výber pozostáva z troch typov obvodov a to Logický obvod, Petriho siete a Stavové automaty.



Obr. 14 Úvodná ponuka

Po výbere je ešte potrebné zadať názov modelu a kliknutím na tlačidlo „ok“ sa užívateľ dostane do okna hlavnej aplikácie.

2.9.3 Architektúra systému

Ako spomínajú autori v svojej práci, produkt Digi Creator bol vytváraný s úmyslom vytvoriť prostredie, s ktorým by mohli študenti pracovať na čo najväčšom počte predmetov zameraných na návrh digitálnych systémov, a takisto prostredie zastrešujúce všetky metodiky návrhu na všetkých úrovniach.

Hlavné ciele projektu boli nasledovné:

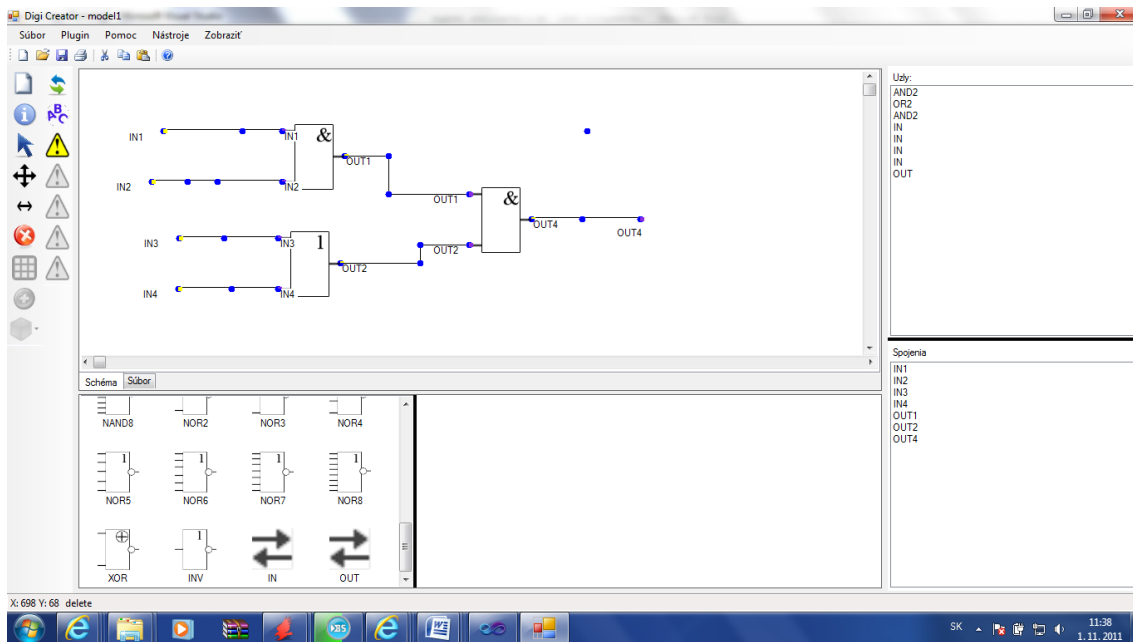
- Základ pre modulárny aplikačný systém umožňujúci prácu s čo najväčším množstvom metodík návrhu.
- Umožniť dodatočné pridávanie nových metodík.
- Podporovať súborové štandardy - BLIF, KISS, SLIF, atď.
- Zahrnutý grafický editor pre klasické hradlové obvody, Petriho siete, Konečné stavové automaty, prípadne iné grafické modely používané pri návrhu.
- Podpora simulácie daných grafických modelov (príkladom je PIPE pre Petriho siete alebo LOG pre hradlové obvody)

Digi Creator bol vyvíjaný v implementačnom prostredí Microsoft Visual C#. Produkt sa skladá z modulov vykonávajúcich požadované operácie a hlavnej aplikácie, ktorá všetky tieto moduly zastrešuje.

Hlavná aplikácia (obr. 15) poskytuje nasledovné:

- Kresliacu plochu, ktorá je umiestnená v strede hlavnej aplikácie
- Panel nástrojov umiestnený v ľavom boku hlavnej aplikácie
- Ponuka aktuálnych dostupných kresliacich prvkov umiestnená v ľavom dolnom rohu

- Okno s popisom jednotlivého prvku nachádzajúceho sa v schéme, ktoré je umiestnené pravom dolnom rohu
- Výpis všetkých uzlov v schéme umiestnený napravo v hornej polovici okna
- Výpis všetkých spojení v schéme umiestnený naľavo v dolnej polovici okna



Obr. 15 – hlavná aplikácia

V súčasnosti Digi Creator obsahuje nasledovné moduly:

- BLIF: Pridáva podporu pre súborový formát blif
- Simplifier: Pridáva podporu pre zjednodušovanie logických funkcií
- ToVHDLsaver: Uloženie obvodu do súboru podľa jeho logickej funkcie
- PNML: Pridáva podporu pre súborový formát pnml
- Stavové automaty: Pridáva podporu pre stavové automaty – podpora pre súborový formát KISS
- Konverzia medzi príbuznými súborovými formátmi

Autori uvádzajú, že posledné dva moduly využívajú príkazy programu SIS sis_bds.exe, ktoré sa mu zadajú presmerovaním vstupov.

2.9.4 Modulárnosť produktu a používané rozhranie

Modulárnosť produktu zabezpečuje ďalšie jednoduché pridávanie potrebných rozširovacích modulov, funkcií a komponentov. V prípade Digi Creatoru je táto modulárnosť zachovaná nasledujúcim spôsobom:

- Hlavná aplikácia, ktorá tvorí samostatnú entitu v projekte
- Prídavné moduly, ktoré podobne ako hlavná aplikácia, tvoria samostatné entity v projekte
- Rozhranie zabezpečujúce komunikáciu modulov a hlavnej aplikácie

Menný priestor *PluginInterface* v sebe zoskupuje entity ako triedy, objekty a funkcie, ktoré priamo súvisia s vytváraným funkčným rozhraním pre komunikáciu hlavnej aplikácie a modulov. Sú to tieto nasledovné:

- Rozhranie *IPlugin*
- Rozhranie *IPluginHost*
- Trieda *SavedNode*
- Trieda *SavedCon*

2.9.4.1 Rozhrania *IPlugin* a *IPluginHost*

V produkte Digi Creator sú vytvorené dve rozhrania *IPlugin* a *IPluginHost*, ktoré zaisťujú komunikáciu medzi hlavnou aplikáciou a modulom. Na to aby hlavná aplikácia mala schopnosť rozoznávať rozširujúce moduly je potrebné, aby takýto modul mal definované určité vlastnosti a práve Na to slúži rozhranie *IPlugin*. V každom novom module je potrebné definovať tieto vlastnosti. Rozhranie *IPlugin* obsahuje nasledujúce vlastnosti:

```
public interface IPlugin
{
    // definícia hlavnej aplikácie
    IPluginHost Host { get; set; }
    // nazov pluginu
    string PlugName { get; }
    // opis pluginu
    string Description { get; }
    // autor
    string Author { get; }
```

```

// verzia
string Version { get; }
// typ (Logic, Petri alebo FSM)
string Type { get; }
// zaklad pluginu
System.Windows.Forms.UserControl MainInterface { get; }
// co sa ma vykonat pri nacistani
void Initialize();
// co sa ma vykonat pri ukonceni
void Dispose();
}

```

Konečné prepojenie hlavnej aplikácie a modulu poskytuje rozhranie *IPluginHost*. Toto rozhranie zahŕňa prototypy metód, pomocou ktorých je možné získavať údaje z hlavnej aplikácie, alebo určité potrebné funkcie. *IPluginHost* obsahuje nasledovné:

```

// zakladne metody, cez ktore sa komunikuje s hlavnou aplikaciou
public interface IPluginHost
{
    // vrati konkretny objekt vyhľadany podľa mena
    Object ReturnObject(string Name);
    // spoji všetky porty s rovnakým názvom
    void ConnectAll();
    // vytvorí uzol s danými parametrami
    void CreateNode(int x, int y, int ID, string Name, string
Type, string[] portsIN, string portsOUT);
    // spoji uzly n1 a n2
    void ConnectNodes(string n1, string n2);
    // pretazena metoda, s pridaným argumentom pre nastavenie
nazvu
    // spojenia
    void ConnectNodes(string n1, string n2, string name);
    // uloži aktuálny obvod do dočasného suboru
    bool UlozUzly();
    // vymaže sa všetko (ako keby sa vytvoril nový projekt)
    void Cleanup();
    // vrati názov modelu
    string GetModelName();
    // nastavi názov modelu
    void SetModelName(string Name);
    // vrati výstupné funkcie
    ArrayList GetFunctions();
    // nastavi sirku vstupu a výstupu pre stavové automaty
    void SetStateInputs(int first, int second);
    //vrati zoznam aktuálnych funkcií
    ArrayList GetFunctionsContent();
    //nastavi zoznam zjednodušenu funkciu
    void SetFunctionsContent(ArrayList setter);
    //vrati hodnotu prepínača changed
    bool GetChanged();
    //nastavi prepínač changed
    void SetChanged(bool setter);
}

```

Novovytvorený modul musí implementovať tieto dve rozhrania pre správnu funkčnosť. Vytvorenie nového zásuvného modulu sa docieľa jednoducho, a to pripojením dynamickej knižnice *PluginInterface.dll*. Táto knižnica je výsledkom kompilácie projektu Digi Creator.

2.9.4.2 Triedy *SavedNode* a *SavedCon*

Okrem rozhraní, ktoré poskytujú vlastnosti, metódy a funkcie pre komunikáciu s hlavnou aplikáciou sú v Menný priestor *PluginInterface* obsiahnuté i triedy, ktoré v sebe definujú základné typy objektov pre uzly a prepojenia. Tieto objekty sú využívané ako v moduloch tak i v hlavnej aplikácii.

Trieda *SavedNode* obsahuje nasledovné:

```
[Serializable]
public class SavedNode
{
    // nazov uzla
    public string Name = "";
    // typ uzla podľa daného standardu, na ktorom
    // sa dohodneme, napr AND2, AND3, PLACE, atd.
    public string Type = "NA";
    // id, pre prípad potreby
    public int id;
    // pole vstupov, su tam ulozene nazvy vstupov
    public ArrayList ConIN = new ArrayList();
    // pole vystupov - nazvy
    public ArrayList ConOut = new ArrayList();
    // suradnice uzla
    public int X = -1;
    public int Y = -1;
}
```

Trieda *SavedCon*

```
[Serializable]
public class SavedCon
{
    // nazov prepojenia
    public string Name = "";
    //Nazov zaciatočného uzla
    public string StartNode = "";
    // nazov končového uzla
    public string EndNode = "";
}
```

2.9.4.3 Načítanie modulov

Po spustení programu sú načítané relevantné moduly, ktoré sú určené pre prácu s daným digitálnym systémom. Tieto moduly sú reprezentované vo forme dynamických knižníc. Schopnosť správne načítať potrebné moduly je zabezpečená vlastnosťou *Type*, v ktorej je uchovaný typ digitálneho systému a môže nadobúdať nasledovné atribúty:

- Petri – typ digitálneho systému Petriho sietí
- Logic – typ digitálneho systému logických obvodov
- FSM – typ digitálneho systému stavových automatov

Moduly sú načítavané z adresára *./Plugins* a to vždy pri vytváraní nového projektu. Počas načítavania sa kontroluje prípona *.dll* knižnice a taktiež či obsahuje všetky náležitosti definované v rozhraní *IPlugin*.

2.9.5 Serializácia

Vo vytvorenom produkte Digi Creator je použitá serializácia. Jedná sa o proces konvertovania dátových štruktúr alebo objektov do formátu, ktorý tvorí jeden celok. Z tohto celku je potom následne možné získať jednotlivé serializované dáta.

Autori touto serializáciou vyriešili problém prevodu nakresleného obvodu v grafickej podobe do súboru. Z konkrétneho modulu je zavolaná funkcia *UlozUzly()*, ktorá vytvára dva súbory a to:

- *obvod_tmp.z5* – tento súbor obsahuje všetky uzly, ktoré sa v danej schéme nachádzajú. Na ukladanie uzlov sa využíva vyššie spomínaná trieda *SavedNode*.
- *arc_tmp.z5* – tento súbor obsahuje všetky spojenia, ktoré sa v danej schéme nachádzajú. Na ukladanie spojení sa využíva vyššie spomínaná trieda *SavedCon*.

Ak nastane situácia kedy potrebuje modul použiť objekt z jedného alebo druhého súboru, deje sa to pomocou deserializácie.

Funkcia na serializáciu je nasledovná:

```
public bool UlozUzly()
{
    Stream stream = File.Open("obvod_tmp.z5", FileMode.Create);
    BinaryFormatter bf = new BinaryFormatter();
    foreach (NodeCtrl node in Nodes)
    {
```

```
        SavedNode _node = new SavedNode();
        _node.ConIN = node.ConIN;
        _node.ConOut = node.ConOut;
        //tu sa naplnaju vlastnosti objektu
        bF.Serialize(stream, _node);
    }
    stream.Close();
    // toto iste sa vykonava aj pre vsetky
}
```

V prípade deserializácie ide o opačný postup – čítanie zo súboru.

2.9.6 Testovanie a používanie produktu Digi Creator

Táto kapitola bude predovšetkým pojednávať o bežnom užívateľskom používaní produktu, o zistených nedostatkoch pri jeho používaní, náhodné testy, ako aj celkové zhodnotenie intuitívnosti, komplexnosti, ďalších možných vplyvov na užívateľa a samozrejme prípadne návrhy na zlepšenie produktu. Testovanie a používanie bude vždy prebiehať jednotlivo pre konkrétny typ obvodu digitálneho systému a následne potom bude zhodnotený produkt ako celok.

2.9.6.1 Typ obvodu logický obvod

Po zvolení typu obvodu logický obvod je následne spustená aplikácia a užívateľ môže začať pracovať na kreslení schémy logického obvodu, poprípade takúto schému otvoriť z BLIF súboru.

Kreslenie je intuitívne a užívateľ má hneď bez zbytočného hľadania prístupné najčastejšie používané logické hradlá a to hradlá typu INVERTOR, AND, NAND, OR, NOR, IN, OUT a ich variácie podľa počtu vstupov. Vstup IN a výstup OUT je na schéme prezentovaný ako malý bod, čo je podľa mňa neprehľadné, nepraktické a pri zložitejších logických obvodoch i nepoužiteľné. Výhodnejšie by bolo použiť väčšiu grafickú reprezentáciu týchto hradiel.

Ďalšiu neprehľadnosť a nepraktickosť spôsobuje nejednoznačnosť pomenovávania uzlov v schéme. Logickým hradlám sa po pridaní do schémy automaticky vytvorí pomenovanie. Toto pomenovanie je však pre každý uzol rovnaké a práve preto je obrovský problém orientovať sa vo výpise uzlov. Zmena názvu je takisto komplikovaná a dokonca i nemožná. Zmena pri premenovaní uzla sa v zozname uzlov neprejaví ! Toto významné obmedzenie je potrebné ošetriť napríklad spôsobom premenovávania uzlov po dvojklik na konkrétny uzol v okne, kde sú všetky uzly zobrazené.

Pri kreslení absentuje ľubovoľné kreslenie čiar – signálov a nadväzovanie signálov na ľubovoľnom mieste – bode. V schéme je možnosť vedenia signálov a spájanie len z bodu IN a z bodu OUT.

Ďalšia nedokonalosť, ktorá sa prejavila až po určitom čase, spočíva vo vytvorení spojenia v tom istom bode. Takéto spojenie je logicky nesprávne a nemalo by byť umožnené užívateľovi vykonať takúto možnosť.

Pri vytváraní spojov sa objekty v schéme niekedy správajú nepredvídateľne. Ide o to, že na to aby sa vytvoril spoj je potrebné kliknúť presne na určité miesto – bod spoju. Ak užívateľ klikne trochu vedľa, objekt sa samovoľne presunie na miesto kliknutia. To môže pôsobiť rušivo hlavne kvôli tomu, že body spájania sú malé a užívateľ nie vždy presne odhadne miesto kliknutia.

Ukladanie logického obvodu do súborového formátu BLIF prebieha bez komplikácií. Súbor má správny formát podľa normy BLIF.

Otváranie súborového formátu jednoduchého logického obvodu, ktorý obsahuje jedno logické hradlo je úplne bezproblémové. Problémy začínajú narastať pri zložitejších obvodoch s obsahom väčšieho počtu logických hradiel. Uzly sú v schéme častokrát rozhádzané, signály vedené z uzlov sú prekryvané a celková schéma pôsobí neprehľadne a miestami pri veľkých schémach i nepoužiteľne. Riešením by mohlo byť pridanie funkcie, ktorá by zaistila neprekryvanie sa signálov a správne rozmiestnenie uzlov v schéme.

Ďalšia chybička krásy sa neskôr ukázala ako chyba, ktorú je potrebné v projekte riešiť. Rýchlym dvojklikom na bod spoju sa objaví ďalší bod, s ktorým však nemožno manipulovať. Po viacnásobnom kliknutí sa po čase objaví v projekte nezachytená výnimka, ktorá nedovoľuje v programe ďalej pokračovať.

Vhodným vylepšením, ktoré by pomohlo k lepšej ovládateľnosti prvkov v schéme by bolo označenie viacerých uzlov súčasne a ich následne presunutie. V súčasnosti je možné presúvať len jeden uzol na obrazovke.

2.9.6.2 Typ obvodu Petriho siete

Podobne ako pri logických obvodoch je po zvolení typu obvodu Petriho siete možné začať kresliť, poprípade načítať schému z PNML súboru.

Užívateľovi sú ponúknuté všetky relevantné uzly potrebné pre vytvorenie plnohodnotnej grafovej štruktúry.

Na rozdiel od typu logických obvodov je hneď po pridaní uzla do schémy vyžadovaný názov uzla. Práve to pokladám za veľmi užitočnú funkcionality, ktorá chýbala pri kreslení a navrhovaní logických obvodov. Takto je možné hneď zo začiatku budovať prehľadnosť vytváratej schémy.

Kreslenie je intuitívne, jednoduché, prehľadné i pri väčších grafových štruktúrach. Manipulácia s uzlami v schéme je poznateľne jednoduchšia ako pri logických obvodoch, takisto ako aj vytváranie spojení medzi uzlami.

Ukladanie do súborového formátu PNML funguje bezchybne a súbor má po kontrole správny štandardný formát súborového formátu PNML.

Pri otváraní súboru PNML sú však menovky uzlov neprehľadne rozhadzvané. Veľmi často sa stáva, že sa jedna cez druhú prekrývajú a tým výrazne rušivým vplyvom pôsobia na užívateľa. V tomto prípade by bolo vhodné implementovať funkciu, ktorá by toto rozloženie menoviek usporiadala a tým zamedzila nečitateľnosti.

2.9.6.3 Typ obvodu stavové automaty

Pre opis obvodu stavové automaty slúži súborový formát KISS. Kreslenie a navrhovanie schémy je podobné ako pri predchádzajúcich dvoch typoch obvodov. Otváranie a ukladanie schémy prebieha bez problémov. Na rozdiel od predchádzajúcich dvoch typov obvodov, tento typ obvodu dokáže pri načítaní zo súboru pekne a prehľadne rozmiestniť všetky objekty na obrazovke.

2.9.7 Zhodnotenie produktu Digi Creator

Digi Creator je softvérový produkt, ktorý má za úlohu zjednodušiť, spríjemniť, sprehľadniť a hlavne zjednotiť prácu študentov pri navrhovaní digitálnych systémov. Myšlienka tohto programu je veľmi pekná, no je potrebné ju ďalej rozvíjať, implementované súčasti zdokonaľovať, ako aj pridávať a rozširovať o ďalšie potrebné funkcionality.

2.10 Používané súborové formáty v digitálnych systémoch

Táto kapitola bude pojednávať o rôznych súborových systémoch, ktoré sa používajú v digitálnych systémoch a je inšpirovaná zo zdrojov [12] a [13].

2.10.1 Súborový formát BLIF

Táto kapitola a podkapitoly sú inšpirované zo zdroja. Cieľom súborového formátu BLIF je opísať logický obvod v textovej forme. Každý uzol má pripojenú dvoj - úrovňovú, jednovýstupovú logickú funkciu. Každá spätná väzba musí obsahovať aspoň jeden zámok. Každá sieť (alebo signál) má iba jeden vodič, a to buď signál, alebo bránu, ktorej vedenie signálu môže byť pomenované bez dvojnásobnosti.

V uhlových zátvorkách sú <neterminály> a v guľatých sú (voliteľné konštrukcie).

Model je "flatened" hierarchický obvod. Súbor BLIF môže obsahovať mnoho modelov a odkazy na uvedené modely v iných BLIF súboroch.

Modely sú deklarované nasledovne:

- *.model* <decl-model-name> - string s menom modelu.
- *.inputs* <decl-input-list> - zoznam vstupných terminálov pre deklarovaný model, viacero riadkov je tu povolených. V prípade ak je toto prvý, alebo jediný model, tieto signály sú identifikované ako primárne vstupy. Viacero *.inputs* riadkov je spojených do jedného zoznamu.
- *.outputs* <decl-output-list> - zoznam výstupných terminálov pre deklarovaný model, viacero riadkov je povolených. V prípade ak je toto prvý, alebo jediný model, tieto signály sú identifikované ako primárne výstupy. Viacero *.outputs* riadkov je spojených do jedného zoznamu.
- *.clock* <decl-clock-list> - zoznam hodinových signálov deklarovaných v modeli. Viacero *.clock* riadkov je spojených do jedného zoznamu hod. signálov
- <command> - môže byť: <logic-gate>, <generic-latch>, <library-gate>, <model-reference>, <subfile-reference>, <fsm-description>, <clock-constraint>, <delay-constraint>

- .end – koniec súboru.

Deklarované meno každého nového modulu sa musí líšiť a byť jedinečné pre každý novovzniknutý model. V prípade ak .inputs (.outputs) nie sú špecifikované, môžu byť odvodené od signálov, ktoré nie sú výstupy (vstupy) iných logických blokov.

Deklarácie .inputs, .outputs, .clock, .end sú nepovinné. Deklarovanie mena modulu takisto nie je povinné. V tomto prípade sa použije ako meno modelu názov BLIF súboru.

V prípade ak nie je určený .clock, ide o čisto kombinačný obvod. Tieto .clock signály sú brané len v prvom modeli, rovnako aj príkazy <clock-constraint> a <delay-constraint>.

Označenie komentára vyzerá nasledovne:

označenie komentára# - tento znak sa logicky nesmie použiť v názvoch signálov.

Ak je „\” na konci riadku, príkaz pokračuje až na nasledovnom riadku

Príklad súborového formátu BLIF môže vyzerat' nasledovne:

```
.model simple
```

```
.inputs a b
```

```
.outputs c
```

```
.names a b c # .names popísaný neskôr
```

```
11 1
```

```
.end
```

```
# nepomenovaný model
```

```
.names a b \
```

```
c # ‘\’ na demonštráciu
```

```
11 1
```

2.10.1.1 Logické hradlá

<logic-gate> - priraduje signálu logickú funkciu v modeli.

Definícia logického hradla je nasledovná:

- *.names* <input-1> <input-2> ... <input-n> <output>
- <pravdivostná tabuľka>- formálne udáva n-vstupový, 1-výstupový PLA opis logickej funkcie daného hradla – {0,1,-} je použitý v n-bit +sirokej vstupnej rovine, {0,1} v 1-bit širokej výstupnej rovine. On = 1, Off = 0.
- <výstup> - reťazec udávajúci názov logického hradla
- <vstup-1..n> - vstupy hradla

Súborový formát BLIF uchováva časti obvodu ako súčet súčinov, a práve preto sú aj obvody vykresľované pomocou hradiel AND, OR a INVENTOR. Súbor BLIF musí mať na výstupe jednotkovú funkciu a tak hradlá ako NAND a NOR sa rozkladajú na podporované hradlá.

Príklad jednoduchého logického hradla s tabuľkou pravdivosti je nasledovný:

```
.names v3 v6 j u78 v13.15
```

```
1--0 1
```

```
-1-1 1
```

```
0-11 1
```

2.10.1.2 Nedefinované stavy

Nedefinované stavy (don't cares) tvoria samostatnú sieť v modeli špecifikovanú na konci modelu a to špecifikované nasledovne:

- *.excd* – označuje, že nasledujúci príkaz *.names* bude patriť k sieti vonkajších don't cares
- *.names* <vstup-1> <vstup-2> ... <vstup-n> <výstup>

- *<pravdivostná tabuľka>*
- *<výstup>* - hl. Výstup s podmienkami don't cares
- *<vstup-1> <vstup-2> ... <vstup-n>* - mená hlavných vstupov pre ktoré sú vyjadrené podmienky Don't cares.
- *<pravdivostná tabuľka>* - označuje n-vstupový a 1-výstupový PLA opis logických funkcií vyhovujúcich podmienkam Don't Cares na výstupe. [PPNDS, 2010/11]

Príklad obvodu s vonkajšími „don't cares“ je nasledovný:

```
.model a
```

```
.inputs x y
```

```
.outputs j
```

```
.subckt b x=x y=y j=j
```

```
.exdc
```

```
.names x j
```

```
1 1
```

```
.end
```

```
.model b
```

```
.inputs x y
```

```
.outputs j
```

```
.names x y j
```

```
11 1
```

```
.end
```

2.10.1.3 Preklápacie Obvody a zámky

Označenie <generic-latch > slúži na vytvorenie oneskorenia v obvode. Môže byť využitý na vytvorenie ľubovoľného zámku, alebo preklápacieho obvodu a jeho špecifikácia vyzerá nasledovne:

- <vstup> – dátový vstup do zámku
- <výstup> – dátový výstup zo zámku
- <typ> – typ hrany
- <ovládanie> – hod. signál pre zámok
- <inic-hod> – zač. stav zámku (2=don't cares, 3=neznámy)

Výsledný zápis môže vyzeráť napríklad takto:

```
. latch <vstup> <výstup> [<typ> <ovládanie>] [<inic-hod>]
```

2.10.1.4 Knižničné hradlá

Označenie <library-gate> vytvorí inštanciu technologicky závislého logického hradla a priradí ho k uzlu, ktorý reprezentuje výstup logického hradla. Logická funkcia hradla, jeho známe technologicky závislé oneskorenie, vedenie a ostatné sú zadefinované príkazom <library-gate > a reprezentované nasledovne:

- .gate <name> <formal-actual-list> - dvojúrovňová reprezentácia ľubovoľného jednovýstupového hradla z knižnice.
- .mlatch <name> <formal-actual-list> <control> - odkazuje na zámok v knižnici.
- .<name> - meno inštancie .gate, alebo .mlatch. musia byť prítomné v danej knižnici. Name zodpovedá danému hradlu alebo zámku v knižnici. Názvy „nand2“, „inv“, a „jk_rising_edge“ v nasledujúcich príkladoch sú opisné názvy hradiel v knižnici.
- .inputs v1 v2
- .outputs j
- .gate nand2 A=v1 B=v2 O=x # given: formals of this gate are A, B, O
- .gate inv A=x O=j # given: formals of this gate are A & O

.end

.end isté, len v technologicky nezávislom tvare:

To .inputs v1 v2

.outputs j

.names v1 v2 x

0- 1

-0 1

.names x j

0 1

Podobne:

To .inputs v1 v2

.outputs j

.names v1 v2 x

0- 1

-0 1

.names x j

0 1

A v nezávislom tvare:

.inputs j kbar

.outputs out

.clock clk

.latch temp q re clk 1 # zámok

.names j k q temp # the .names vstup pre .latch

-01 1

1-0 1

.names q out

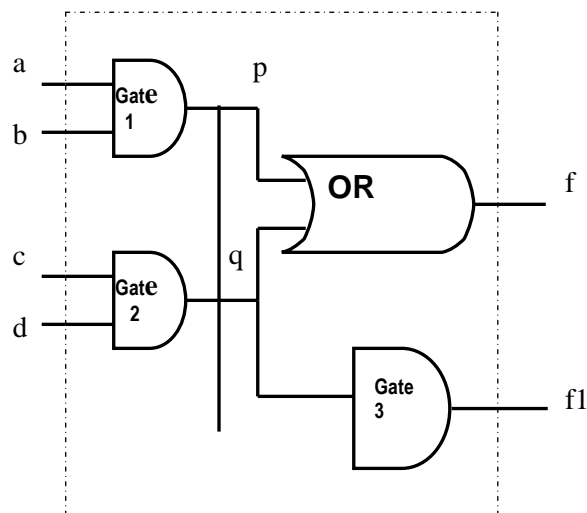
```
0 1
.names kbar k
0 1
.end
```

<formal-actual-list> namapovanie formálnych parametrov knižničného hradla na aktuálne signály v danom modeli:

```
Formát: formal1 = aktual1
        formal2 = aktual2
...
.
```

2.10.2 Súborový formát PLA

Pre lepšie pochopenie súborového formátu PLA (Programmable Logic Array) využijeme nasledujúci logický obvod zobrazený na obr. 16



Obr. 16 Logický obvod PLA

PLA formát obsahuje nasledovné:

- Počet vstupov, notácia - *.i*
- Počet výstupov, notácia - *.o*
- Mená vstupov spojení, notácia - *.ilb*

- Mená výstupov spojení, notácia - *.ob*
- Pravdivostná tabuľka, notácia - *.p* a počet riadkov pravdivostnej tabuľky
- Koniec súboru, notácia - *.e*

Pravdivostná tabuľka pre daný obvod je takáto:

Tabuľka 1 Pravdivostná tabuľka

A	B	C	D	F	F1
1	1	-	-	1	0
-	-	1	1	1	0
1	1	1	1	1	1

V našom prípade bude PLA súbor vyzerat' takto:

```

.i 4
.o 2
.ilb a b c d
.ob f f1

11-- 10
--11 10
1111 11
.e

```

2.10.3 Súborový formát EQN

Pre lepšie pochopenie súborového formátu EQN (Equation format) budeme používať rovnaký logický obvod ako pri súborovom formáte PLA.

Formát EQN patrí medzi najjednoduchšie súborové formáty pre popis logických obvodov. Obsahuje nasledovné:

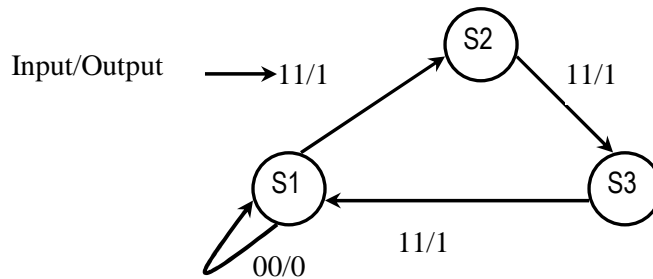
- Počet vstupov, notácia – *INORDER*
- Počet výstupov, notácia – *OUTORDER*
- Špecifikácia pomocných signálov, notácia – napríklad: $[p] = a*b$; Pomocné signály sú uvádzané v hranatých zátvorkách.
- Špecifikácia výstupov, notácia – napríklad $f1 = [p]*[q]$; Vstupy sú špecifikované pomocnými signálmi.

V našom prípade bude EQN súbor vyzerat' nasledovne:

INORDER = a b c d;
OUTORDER = f f1;
[p] = a * b;
[q] = c * d;
f = [p] + [q]
f1 = [p] * [q];

2.10.4 Súborový formát KISS

Pre lepšie pochopenie súborového formátu KISS využijeme nasledujúci logický obvod zobrazený na obr. 17



Obr. 17 Logický obvod KISS

Formát KISS sa používa na minimalizáciu stavových diagramov a obsahuje nasledovné:

- Počet vstupov, notácia - *.i*
- Počet výstupov, notácia - *.o*
- Počet stavov, notácia - *.s*
- Počet riadkov v stavovej tabuľke - *.p* a počet riadkov
- Koniec súboru, notácia - *.e*

Tabuľka 2 Stavová tabuľka

Input	PS	NS	Output
00	S1	S1	0
11	S1	S2	1
11	S2	S3	1
11	S3	S1	1

PS-Present State - súčasný stav
 NS-Next State - nasledujúci stav

V našom prípade bude KISS súbor vyzerat' nasledovne:

.i 2
.o 1
.s 3
.p 4

00 s1 s1 0
11 s1 s2 1
11 s2 s3 1
11 s3 s1 1
.e

2.10.5 Súborový formát PNML

Súborový formát PNML patrí medzi štandardizovaný formát ISO/IEC 15909 a je založený na formáte XML (Extensible Mark-up Language). Pôvodne bol tento formát určený pre JAVU, no vysoký dopyt spôsobil jeho štandardizáciu v práve spomínanom XML formáte.

Dôležitý aspekt súborového formátu PNML spočíva v otvorenosti. Prejavuje sa to tak, že medzi všeobecnými a presne špecifikovanými črtami Petriho sietí sa rozlišuje. PNML formát navyše poskytuje zdieľanie špecifických vlastností – viaceré objekty môžu zdieľať určitú vlastnosť.

Každá konkrétna Petriho sieť sa skladá z kolekcie predefinovaných prvkov. Existuje možnosť vytvorenia nových typov, ak je potrebné opísať pokročilejšiu stavbu siete. Tieto nové typy sa vytvárajú prostredníctvom DTD alebo XML schém.

Usporiadanie PNML súborov je nasledovné:

- Priechody
- Miesta
- Oblúky

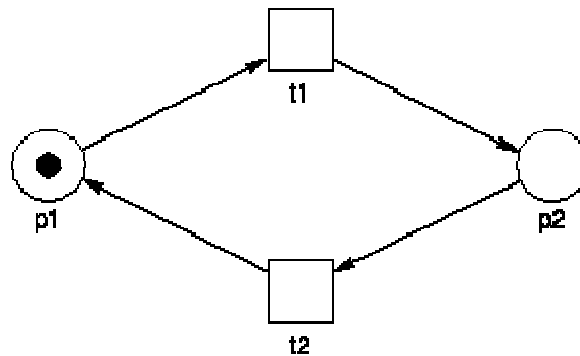
Jednotlivé značky môžu byť v prípade potreby vybavené ďalšími značkami.

Syntax súborového formátu PNML navyše ponúka nasledovné:

- Definovanie prvku ako uzol
- Umiestnenie prvku

Tieto možnosti je možné využiť v grafickom rozhraní pre presné definovanie prvku ako aj jeho umiestnenia.

Príklad Petriho siete a jeho súborový formát PNML môže vyzerat' nasledovne:



Obr. 18 Príklad Petriho siete

```

<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml">
  <net id="n1" type="ptNet">
    <name>
      <value>1-safe circle</value>
    </name>
    <place id="p1">
      <name>
        <graphics><offset page="1" x="0" y="22" /></graphics>
        <value>p1</value>
      </name>
      <initialMarking>
        <graphics><offset page="1" x="-20" y="10" /></graphics>
        <value>1</value>
      </initialMarking>
      <graphics><position page="1" x="50" y="150" /></graphics>
    </place>
    <place id="p2">
      <name>
        <graphics><offset page="1" x="-1" y="20" /></graphics>
        <value>p2</value>
      </name>
      <initialMarking>
        <graphics><offset page="1" x="24" y="5" /></graphics>
        <value>0</value>
      </initialMarking>
      <graphics><position page="1" x="250" y="150" /></graphics>
    </place>
    <transition id="t1">
      <name>
        <graphics><offset page="1" x="0" y="26" /></graphics>
        <value>t1</value>
      </name>
      <graphics><position page="1" x="150" y="50" /></graphics>
    </transition>
    <transition id="t2">
      <name>
        <graphics><offset page="1" x="-1" y="22" /></graphics>
        <value>t2</value>
      </name>
      <graphics><position page="1" x="150" y="250" /></graphics>
    </transition>
    <arc id="a1" source="p1" target="t1">
      <inscription>

```

```

    <graphics><offset page="1" x="20" y="0" /></graphics>
    <value>1</value>
  </inscription>
  <graphics><position page="1" x="100" y="100" /></graphics>
</arc>
<arc id="a2" source="t1" target="p2">
  <inscription>
    <graphics><offset page="1" x="20" y="0" /></graphics>
    <value>1</value>
  </inscription>
  <graphics><position page="1" x="200" y="100" /></graphics>
</arc>
<arc id="a3" source="p2" target="t2">
  <inscription>
    <graphics><offset page="1" x="20" y="0" /></graphics>
    <value>1</value>
  </inscription>
  <graphics><position page="1" x="200" y="200" /></graphics>
</arc>
<arc id="a4" source="t2" target="p1">
  <inscription>
    <graphics><offset page="1" x="20" y="0" /></graphics>
    <value>1</value>
  </inscription>
  <graphics><position page="1" x="100" y="200" /></graphics>
</arc>
</net>
</pnml>

```

3 Špecifikácia požiadaviek

Témou nášho tímového projektu je vytvorenie základu pre modulárny aplikačný systém umožňujúci prácu s čo najväčším množstvom metódik návrhu. Mali sme k dispozícii 2 projekty s rovnakou témou z minulého roka. Preto sme sa rozhodli z nich vybrať niektoré súčasti, ktoré sme považovali za najlepšie a najužitočnejšie a implementovať ich do nášho projektu.

Na základe analýzy predchádzajúcich projektov a spoločnej dohody sme sa zhodli na požiadavkách pre systém, ktorý budeme implementovať v rámci predmetu tímový projekt.

3.1 Všeobecné požiadavky

Niektoré požiadavky priamo vyplývajú zo zadania témy projektu. Patria sem nasledovné požiadavky:

- Modularita – Požiadavka na systém je, aby bol modulárny. To znamená, že by sa mal skladať z menších programových celkov, tzv. modulov. Každý tento celok bude realizovať určitú funkcionality, ktorá sa dá zmeniť výmenou jedného modulu a všetko ostatné je možné nechať v takom stave, ako to bolo a program bude pracovať naďalej. Pre zabezpečenie bezproblémovej komunikácie medzi modulmi je teda potrebné vymyslieť konkrétny univerzálny jazyk, ktorý bude štandardizovať formát vstupu a výstupu modulov.
- Rozšíriteľnosť – je požiadavka priamo závislá na predchádzajúcej. Implementáciou modulárneho systému je možné zabezpečiť jednoduchú rozšíriteľnosť o ďalšie funkcie v budúcnosti. Stačí vytvoriť nový modul, ktorý bude dodržiavať definované štandardné vstupy a výstupy a je možné ho vložiť do systému namiesto nepotrebného modulu.
- Grafické rozhranie – Systém by mal zahŕňať grafický editor obvodov, na čo je potrebné aj grafické používateľské rozhranie. Toto rozhranie by malo byť prehľadné a poskytovať používateľovi všetky možnosti systému. Požiadavky na grafický editor budú definované ďalej v špecifikácii.

- Overiteľnosť – V zadaní bola definovaná požiadavka na možnosť simulácie obvodu. Preto bude systém obsahovať aj takúto možnosť a požiadavky budú definované neskôr.

3.2 Funkcionálne požiadavky

Po definovaní základných všeobecných požiadaviek na systém treba definovať funkcionálne požiadavky, ktoré sa budeme snažiť implementovať. Zo zadania projektu vyplýva potreba vytvoriť modulárny program pre návrh digitálnych systémov. Pri vytváraní budeme brať do úvahy a čiastočne prevezmeme práce predchádzajúcich tímov v danej téme.

Oblasť návrhu digitálnych systémov je pomerne široká a preto sme sa rozhodli zamerať na oblasť, ktorá nám je najlepšie známa, a to oblasť kombinačných logických obvodov. Keďže bude program rozšíriteľný, bude možné v budúcnosti pridať aj moduly pre návrh sekvenčných logických obvodov, petriho sietí a ďalšie podľa potreby.

Po starostlivej analýze problematiky a predchádzajúcich projektov sme dospeli k nasledovným požiadavkám na systém:

- Textový editor – systém by mal obsahovať jednoduchý textový editor s funkciami uľahčujúcimi tvorbu zdrojového kódu, ako napríklad:
- zvýrazňovanie syntaxe – farebné označenie kľúčových a funkčných slov v zdrojovom kóde pre zlepšenie orientácie a prehľadnosti.
- konverzia súborových formátov – je možné využiť moduly tímového projektu z predchádzajúceho roku, kde sa moduly volali ako samostatné programy cez príkazový riadok. Konverzia bude zabezpečovať transformáciu medzi jednotlivými formátmi súborov ako sú napríklad PLA alebo BLIF.
- číslovanie riadkov – pre zlepšenie orientácie v zdrojovom kóde by mali byť riadky číslované.
- funkcie krok späť a krok dopredu (undo/redo)
- Grafický editor – mal by slúžiť na úpravu obvodov metodikou WYSIWYG.

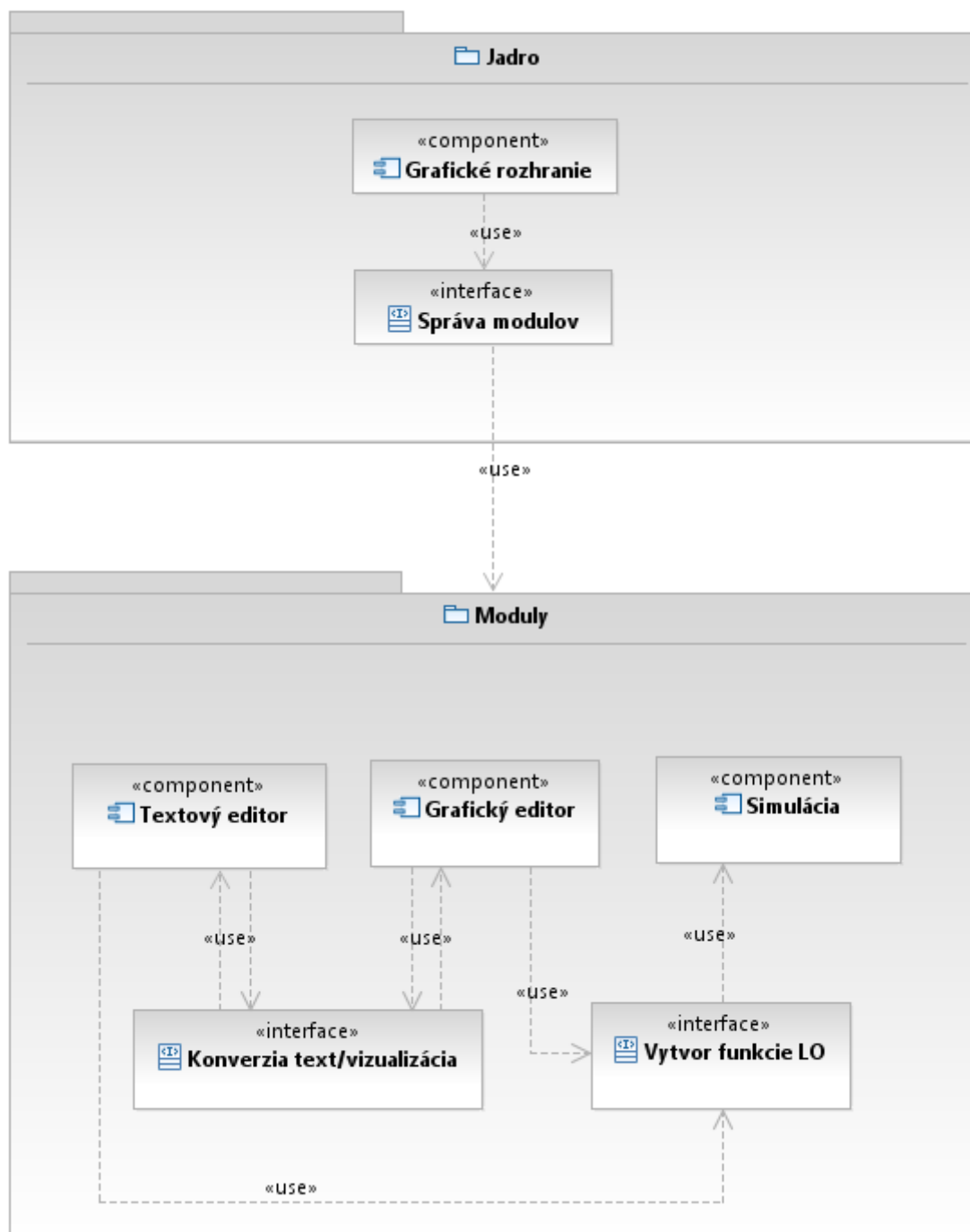
- prehľadnosť a jednoduché používanie – editor by mal byť jednoduchý a intuitívny, aby ho používateľ mohol používať automaticky hneď po inštalácii programu a nemusel venovať kopu času čítaniu manuálov a príručiek.
- vytváranie vlastných hradíel – okrem vopred definovaných základných hradíel nachádzajúcich sa v systéme, by mal mať používateľ možnosť jednoducho vytvoriť nové hradlá definovaním názvu, obrázku a kódu správania sa. Tu je pravdepodobne možné využiť editor z predchádzajúceho projektu.
- grafické zobrazenie kódu – systém by mal zvládnuť prepínanie medzi textovým a grafickým editorom a bude teda potrebné zabezpečiť prepis užívateľom zadaného zdrojového kódu do grafického znázornenia schémy a naopak.
- označenie vstupov a výstupov – editor by mal v základnej sade hradíel poskytovať 2 špeciálne koncové body a to vstup a výstup obvodu. Tento špeciálny koncový bod sa bude musieť pripojiť na všetky vonkajšie vstupy a výstupy, aby bolo možné simulovať správanie obvodu.
- Simulácia – simulácia slúži na overenie správania sa obvodu. Mali by byť minimálne 2 módy simulácie:
 - automatická – zoberie vytvorený obvod, načíta vstupy, výstupy a ich funkcie na výstupe. Následne vygeneruje pravdivostnú tabuľku pre všetky možné vstupy.
 - manuálna – zobrazí vytvorenú schému, ktorá sa však v simulačnom režime nebude dať upravovať a používateľ bude môcť kliknutím na vstupné koncové body prepínať ich hodnoty na vstupe a sledovať hodnoty na výstupe, čím môže overiť očakávané hodnoty na výstupe.

4 Hrubý návrh riešenia

Predložený návrh systému sa čiastočne podriaďuje faktu, že tento systém prevezme upravené časti z už existujúcich aplikácií. Konkrétne ide o aplikácie, ktoré vznikli pri riešení tímového projektu v ak. roku 2010/2011 tímom 2 a tímom 10 (oboru PKSS FIIT STU). Tím číslo 10 implementoval textový editor v jazyku JAVA. Tím číslo 2 grafický editor v jazyku C#. Pre implementáciu nášho systému sú vhodné oba jazyky. Ako tím sme sa priklonili k výberu jazyka C#, k čomu prispeli lepšie skúsenosti s týmto jazykom. Ďalšou výhodou tohto výberu je aj fakt, že môžeme jednoducho prebrať a upraviť grafický editor, ktorý je náročnejší na implementáciu ako textový editor.

4.1 Architektúra systému

Jednou zo základných požiadaviek na navrhovaný systém je dosiahnuť modulárnosť. Preto je navrhovaná aplikácia rozdelená na jadro a jednotlivé moduly. Komunikácia medzi jadrom a modulmi bude vykonávaná prostredníctvom rozhraní. Na obr. 1 je načrtnutý hrubý návrh architektúry aplikácie.



Obr. 19 Architektúra systému

4.2 Jadro

Jadro bude tvoriť základ aplikácie. Aby bola aplikácia jednoducho rozšíriteľná bude jadro poskytovať iba základné funkcie slúžiace predovšetkým na ovládanie modulov. Základné úlohy jadra aplikácie sú zhrnuté do nasledujúcich bodov:

- Načítanie, spustenie a správa modulov

- Poskytnutie grafického rozhrania slúžiaceho predovšetkým na zobrazenie modulov

4.3 Moduly

Jednotlivé moduly slúžia na samotné vykonávanie špecifikovaných funkcií aplikácie. Pri návrhu je dôležité vhodne rozdeliť funkcionality medzi moduly.

4.3.1 Textový editor

Modul textový editor bude prostredníctvom grafického rozhrania poskytovať nasledovné funkcie:

- Vytvoriť nový súbor
- Otvoriť/editovať/uložiť súbor
- Volanie externých programov na konverziu medzi súborovými systémami
- Zobrazíť textovú podobu logického obvodu z grafického editoru
- Kontrola a zvýrazňovanie syntaxe súborových systémov
- "undo/redo"

4.3.2 Grafický editor

Tento modul bude podporovať len prácu s logickými obvodmi. Prostredníctvom tohto modulu bude mať užívateľ sprístupnené nasledovné funkcie:

- Nakresliť vlastný logický obvod z dostupných hradíel
- Vykresliť a editovať logický obvod z uloženého súboru, alebo z textového editoru
- Uložiť nakreslený logický obvod do súboru
- Možnosť definovať vlastné hradlá

Na užívateľskú prívetivosť grafického rozhrania modulu bude kladený výnimočný dôraz.

4.4 Simulácia

Modul simulácia bude slúžiť na vyhodnotenie výstupov logických obvodov navrhnutých buď cez textový, alebo grafický editor. Užívateľ bude mať možnosť vykonať automatickú, alebo manuálnu simuláciu. Pri automatickej simulácii sa užívateľovi zobrazí pravdivostná tabuľka navrhnutého logického obvodu. Pri manuálnej simulácii sa užívateľovi vykreslí navrhnutý obvod. Užívateľ bude môcť interaktívne meniť hodnotu vstupu kliknutím na jeho grafickú reprezentáciu. Aplikácia následne graficky zobrazí príslušné hodnoty výstupov.

4.5 Požiadavky na systém

Hardvérové požiadavky

Processor Intel alebo AMD s taktovacou frekvenciou 300 MHz a viac

- 128 MB pamäte RAM a viac
- Monitor s rozlíšením 800x600 a vyššie
- Klávesnica a myš
- 35 MB voľného miesta na pevnom disku

Softvérové požiadavky

Aplikácia má minimálne požiadavky na systém na ktorom sa bude spúšťať. Je potrebný počítač s operačným systémom Windows XP a vyššie.

- Operačný systém Windows XP , Vista, Windows 7
- Platforma .NET 4.0 a vyššia

5 Návrh a implementácia

V tejto časti sa budeme venovať niektorým vybraným oblastiam návrhu a implementácie finálneho produktu.

5.1 Architektúra systému

Po spustení programu je na prvý pohľad vidieť rozdelenie systému na 3 základné moduly a to sú:

- Textový editor
- Grafický editor (tzv. drag&drop editor)
- Verifikácia

Hlavným elementom programu je **jadro**, aj keď nie je na prvý pohľad celkom viditeľné. Úlohou jadra je zabezpečiť komunikáciu medzi modulmi a poskytnúť niektoré všeobecné funkcie. V hrubom návrhu aplikácie bolo v pláne umožniť používateľovi načítať iné editory, ktoré by mohli poskytovať viac funkcií pre dané použitie. Nakoniec sme však rozhodli, že je takéto riešenie zbytočné a stačí mať jeden editor z každého druhu. Ako už bolo vyššie spomenuté, každý editor funguje viacmenej ako samostatný modul a pre komunikáciu medzi modulmi je potrebné využiť funkcie jadra.

Komunikácia funguje spôsobom, že keď chce niektorý modul odoslať svoje dáta inému, uloží všetky potrebné informácie o hradlách a prepojeniach do špeciálnej štruktúry v jadre a zavolá funkciu, ktorá tieto informácie zoserializuje a uloží do dočasného súboru na disku. Druhý modul pre načítanie údajov musí zavolať funkciu na deserializáciu tohto dočasného súboru, čím sa dáta z neho uložia naspäť do špeciálnej štruktúry v jadre, odkiaľ ich môže ktorýkoľvek editor čítať.

5.2 Prevod VHDL kódu do grafického editora

Určite veľmi zaujímavou funkciou nášho programu je prevod VHDL kódu, či už zo súboru alebo používateľom napísaného v textovom editore, do grafického editora v podobe schémy. Pre aktiváciu tejto funkcie je nutné mať zapnuté zvýrazňovanie

syntaxe na jazyk VHDL a prepnúť do grafického editora. Následne by sa malo zobrazíť informačné okno o úspešnom, resp. neúspešnom načítaní VHDL kódu a po kliknutí na tlačidlo OK sa obvod vykreslí. Samotné načítavanie funguje nasledovne:

Najprv sa do premennej načíta kód z textového editora (pri otvorení súboru sa jeho obsah uloží najprv do textového editora a odtiaľ sa pokračuje rovnakým spôsobom). Z premennej sa vytvorí pole jednotlivých riadkov kódu, ktorým program postupne prechádza a každý riadok rozdelí na slová do dočasného poľa, s ktorým následne pracuje. Z tohto poľa sa zoberie prvé slovo, ktoré v syntaxi VHDL väčšinou obsahuje príkaz. Tento príkaz sa vyhodnotí a následne sa podľa príkazu určí akcia. Ak sa daný príkaz nepodarilo rozpoznať, je používateľ informovaný a spracovanie kódu pokračuje ďalej. Všetky načítané komponenty, entity, prepojenia a podobne sa ukladajú do dočasných premenných. Po prejdení všetkých riadkov kódu sa zavolá funkcia Draw(), ktorej úlohou je pospájať získané informácie z viacerých polí a správne uložiť hradlá do štruktúry v jadre, kde sa následne serializuje a poskytne ostatným modulom k dispozícii. Súčasťou tejto funkcie je aj volanie funkcie MakePins, ktorá vytvára vstupné a výstupné piny pre zjednodušenie prechodu z grafického editora na verifikáciu.

Funkcia MakePins() funguje tak, že postupne prechádza všetky hradlá a zisťuje či sú všetky konektory zapojené. Ak niektorý nie je zapojený a neexistuje výstup z iného hradla s rovnakým názvom, ktoré by sa dali prepojiť, vytvorí MakeCons vstupný alebo výstupný pin, ktorý sa neskôr pomocou funkcie MakeCons() automaticky zapojí. Funkcia MakePins() má však ešte jednu veľmi dôležitú funkciu a to zarovnanie hradliel do akejsi hierarchickej schémy. Každé hradlo má v internej premennej uloženú úroveň hradla. Vždy keď sa vytvára spojenie medzi dvoma hradlami, porovnávajú sa ich hodnoty úrovní a ak je hodnota úrovne hradla z ktorého vychádza spojenie von vyššia ako hodnota úrovne hradla do ktorého vstupuje prepojenie, nastaví sa hodnota hradla do ktorého prepojenie vstupuje o 1 vyššia ako je hodnota úrovne hradla z ktorého prepojenie vychádza. Pri posielaní hradla do grafického editora sa jednoducho posunie pozícia hradla o výsledok násobenia hodnoty úrovne hradla a nejakej stanovenej konštanty.

Je nutné upozorniť, že prevod VHDL do grafického editora je stále v beta verzii a preto pri rôznych príkazoch a štýloch písania VHDL kódu môžu nastať chyby. Pre otestovanie odporúčam použiť nasledovný program, prípadne podobným štýlom opísané jednoduché obvody.

Ukážka odladeného prevoditeľného VHDL kódu:

```
----- NAND 2 vstupovy -----
entity oNAND2 is
  port(IN1:in BIT; IN2:in BIT; OUT1:out BIT);
end oNAND2;
architecture sprNAND2 of oNAND2 is
begin
  OUT1 <= IN1 nand IN2;
end sprNAND2;
----- AND 2 vstupovy -----
entity oAND2 is
  port(IN1:in BIT; IN2:in BIT; OUT1:out BIT);
end oAND2;
architecture sprAND2 of oAND2 is
begin
  OUT1 <= IN1 and IN2;
end sprAND2;
-----

entity obvod is
  port ( a:in Bit; b:in Bit; c:in Bit; negC:out Bit);
end obvod;
architecture Spravanie of obvod is
  component cNAND2
    port(IN1:in Bit; IN2:in Bit; OUT1:out Bit);
  end component;
  component cAND2
    port(IN1:in Bit; IN2:in Bit; OUT1:out Bit);
  end component;
  for xNAND2_a: cNAND2 use entity Work.oNAND2(sprNAND2);
  for xNAND2_b: cNAND2 use entity Work.oNAND2(sprNAND2);
  for xNAND2_c: cNAND2 use entity Work.oNAND2(sprNAND2);
begin
  xNAND2_a: cNAND2 port map (a, b, negA);
  xNAND2_b: cNAND2 port map (b, c, negB);
  xNAND2_c: cNAND2 port map (negA, negB, negC);
end Spravanie;
```

5.3 Grafický plugin

5.3.1 Vykresľovanie hradiel

Hlavnou funkciou grafického pluginu je funkcia grafického vykresľovania obvodou pomocou hradiel a prepojení používateľom a vykresľovanie jednotlivých hradiel a prepojení uložených v jadre, na plochu do *pictureboxu*. Viacero funkcií na vykresľovanie bolo pridaných z projektu tímu Z5, pričom boli prispôsobené našim požiadavkám. Hradlá na vykresľovanie sú načítavané a deserializované z xml súboru *testNode.xml* pomocou funkcie *SetGates()*, ktorý je možné upravovať v programe Digi Editor od tímu z5. Na výber je niekoľko hradiel typu AND, OR, NOR, NAND, XOR

a INVENTOR. Po výbere hradla spôsobom drag&drop je možné tieto upravovať (mazať, presúvať) a prepájať. Na zistenie, ktorá funkcia sa má konkrétne vykonať, používa program premennú *action*, ktorá v závislosti od zvoleného tlačidla nadobúda hodnotu *connect*, *delete*, *move* atď. Rovnako premenná *action2* (hodnoty *movecon*, *movepoint*) slúži na nastavenie potrebnej funkcie pri práci s prepojeniami. Hodnoty premennej *action* je možné nastaviť kliknutím na tlačidlá v lište na ľavej strane grafického editoru. Pri prechádzaní kurzorom nad daným hradlom, alebo prepojením sa zobrazia informácie o konkrétnej entite pomocou funkcie *setGroupNode()*, *SetGroupPort()* a nastavení jednotlivých labelov *Groupboxu groupCon*. Príslušné *Groupboxy* sa nastavujú na viditeľné, kedykoľvek používateľ prejde po konkrétnej entite kurzorom. Zobrazovanie pridaných hradiel a prepojení sa uskutočňuje na pravej lište. Tieto položky sa načítavajú zo zoznamov *NodeCtrl* a *Connections* pomocou funkcií *RefreshLostNodes()* a *RefreshLostCons()*.

5.3.2 Ukladanie a načítavanie hradiel

Veľmi dôležitou funkciou okrem kreslenia obvodov pomocou hradiel a prepojení, je aj ukladanie zoznamov do jadra a načítavanie týchto zoznamov z jadra. Tieto funkcie sú potrebné na ukladanie a načítavanie rôznych pluginov (napríklad pre prácu s rôznymi súborovými formátmi) z a do grafického editora.

Na ukladanie Zoznamov prepojení a hradiel do jadra slúži funkcia *SaveToCore()*, ktorá je automaticky volaná pri ukladaní súboru, alebo po kliknutí na tlačidlo odoslania do jadra, v hornej časti grafického modulu. Táto vymaže všetky položky zo zoznamov v jadre a zavolá funkcie jadra *SetNode()* a *SetCon()*, ktoré na základe zoznamu hradiel a prepojení v grafickom rozhraní, vytvorí zoznamy s potrebnými informáciami v jadre. Nakoniec je zavolaná funkcia *UlozUzly(„obvod_tmp.z5“)*, ktorá zoserializuje Zoznamy jadra do súboru *obvod_tmp.z5*.

Načítavanie zoznamov z jadra sa vykonáva pomocou funkcie *NacitajUzly(„obvod_tmp.z5“)*, ktorá vykoná deserializáciu dočasného súboru *obvod_tmp.z5*, v ktorom sú serializované zoznamy hradiel a prepojení z jadra. Po načítaní týchto zoznamov sa hradlá a prepojenia vykreslia na plochu grafického editora, kde sa následne znovu dajú upravovať. Táto funkcia sa automaticky vykonáva pri

načítavanie súborového formátu blif do jadra, alebo pri prepnutí karty na kartu grafického modulu.

Pri rôznych funkciách (napríklad načítavanie, alebo ukladanie) je často potrebné najskôr vymazať aktuálne zoznamy hradiel pomocou funkcie *Cleanup()* a následné nahradenie týchto zoznamov novými.

5.3.3 Riešenie problémov s modulom grafického editora

Pri grafickom module nastal problém s vykresľovaním jednotlivých hradiel a prepojení, ktoré boli posunuté vzhľadom na kurzor o niekoľko pixelov, tento posun sa menil aj pozíciou programu na ploche. Správne nakalibrovanie grafického editora bolo možné nastavením súradníc vzhľadom na jadro celej aplikácie a nie na grafický modul samotný

```
(  
int x = System.Windows.Forms.Cursor.Position.X -  
Logic_Designer.Form1.ActiveForm.Location.X - pictureBox.Left - 25 - modX;  
int y = System.Windows.Forms.Cursor.Position.Y -  
Logic_Designer.Form1.ActiveForm.Location.Y - pictureBox.Top - 100 - modY;  
).
```

Grafický modul je veľkou výhodou nášho programu, pretože umožňuje vizuálne manipulovať s obvody a tým umožňuje lepšiu prácu s formátmi ako *blif*, alebo *VHDL*.

5.3.4 Plugin Blif

Blif je plugin pre Hlavný program, pracujúci ako s textovým editorom, tak aj s grafickým rozhraním (a teda aj verifikáciou). Tento plugin používateľovi umožňuje načítavať a ukladať súbory z blif formátu a do blif formátu. Tento je možné zobrazovať v textovom editore (aj so zvýrazňovaním), alebo v grafickom editore.

5.3.5 Načítavanie z *Blif* formátu

V nasledujúcej kapitole popíšeme prevod obvodu z formátu blif do podoby zoznamov v jadre.

Pomocou hlavného menu (Súbor->otvoriť) otvoríme ľubovoľný *Blif* súbor s koncovkou *.blif*. Tento súbor je spracovávaný po riadkoch až do konca. Plugin spracováva nasledovné kľúčové slová:

.model – názov modelu

.inputs – globálne vstupy modelu

.outputs – globálne výstupy modelu

.names – časť obvodu

.end – koniec súboru

Hradlá obvodu sú ukladané do zoznamov po analýze časti obvodu za kľúčovým slovom *.names*. Vstupy v jednom riadku sú členené do hradla typu AND a viac riadkov jednej častiobvodu tvorí hradlo OR. Ak sú vstupy nulové, tak je pridané hradlo typu INVERTOR. Keďže formát BLIF uchováva časť obvodu ako súčet súčinov, tak preto sú aj obvody vykresľované pomocou hradiel AND,OR a INVERTOR [zdroj - Prostredie pre návrh digitálnych systémov projektu z5]. Hlavný program Tieto časti obvodu ukladá do zoznamu hradiel, generuje zoznam prepojení a následne sa s týmto zoznamom môže pracovať (graficky vykresliť atď.)

5.3.6 Načítavanie do *Blif* formátu

Ukladanie do formátu *Blif* má na starosti Jadro (Súbor->uložiť). Po zvolení možnosti uloženia do formátu *Blif*, výbere názvu súboru a potvrdení tlačidlom 'Uložiť', prebehne samotná konverzia na súbor *Blif*. Pri konverzii sa využíva metóda jadra programu *UlozUzly()*, ktorá vykoná serializáciu existujúcich zoznamov hradiel do dočasného súboru *blif_obvod_tmp.z5*. Tento modul je v rámci modulu načítaný a deserializovaný. Z deserializovaných prvkov sa vytvorí štruktúra súboru *BLIF*. Opäť sa využívajú vyššie spomenuté kľúčové slová, aby bol súbor v korektnom formáte. Je potrebné upozorniť na to, že BLIF súbor má mať na výstupe jednotkovú funkciu. Preto, ak sa v schéme objaví hradlo NOR, tak je prepísané do súboru pomocou hradiel INVENTOR a OR. Podobne hradlo NAND je prepísané pomocou INVENTOR a AND. Logická funkcia obvodu však zostáva nezmenená, a používateľ môže naďalej v programe pracovať (pridávať, spájať atď.) hradlá typu NAND a NOR.

Plug-in svojou funkcionalitou splňa požiadavky definované pre podporu logických obvodov vo formáte BLIF.

5.4 Textový modul

Textový modul je implementovaný ako trieda *NumberedRichTextBox*. Ako názov napovedá, tak táto trieda rozširuje štandardnú triedu *RichTextBox*. Medzi základné funkcie textového editora patrí číslovanie riadkov, zvýrazňovanie aktuálneho riadku a zobrazenie podrobností v statusovom riadku (suma riadkov a znakov, pozícia kurzora v texte).

Hlavnou funkciou textového editora je zvýrazňovanie syntaxe pre rôzne jazyky (vhdl, verilog, blif, kiss, ...). Syntax jazykov pre zvýrazňovanie nesie súbor *syntax.xml*, ktorý sa nachádza v hlavnom adresári aplikácie. Editáciou tohoto súboru je možné doplniť slová do syntaxe niektorého jazyka, alebo priamo pridať celý nový jazyk. Textový modul podporuje neobmedzené množstvo jazykov, pričom pre každý jeden jazyk je možné rozlišovať až medzi šiestimi rôznymi skupinami slov plus špeciálnou skupinou pre riadkový komentár. Príklad ako vyzerá fragment súboru *syntax.xml* pre vhd1 jazyk:

```
<Jazyk meno="vhd1">
  <Koncovka>vhd|vhd1</Koncovka>
  <KlucoveSlova meno="riadkovyKomentar">--</KlucoveSlova>
  <KlucoveSlova meno="skupina1">access|after|alias</KlucoveSlova>
  <KlucoveSlova meno="skupina2">abs|and|mod</KlucoveSlova>
  <KlucoveSlova meno="skupina3">left|right|low</KlucoveSlova>
  <KlucoveSlova meno="skupina4">now|readline|read</KlucoveSlova>
  <KlucoveSlova meno="skupina5">std|ieee|work</KlucoveSlova>
  <KlucoveSlova
meno="skupina6">boolean|bit|character</KlucoveSlova>
</Jazyk>
```

Čo je dobré si všimnúť na príklade je to, že každé kľúčové slovo syntaxe je oddelené znakom "|". Pri načítaní súboru textovým editorom sa automaticky zvýrazní syntax podľa poskytnutej koncovky (napr. ak má súbor koncovku vhd alebo vhd1, tak sa zvýrazní vhd1 syntax).

Najväčším problémom pri implementácii textového editora bolo dosiahnutie akceptovateľnej rýchlosti zvýraznenia syntaxe, predovšetkým pri otvorení väčšieho súboru. Pri riešení tohoto problému bol zdrojový kód optimalizovaný viacerými technikami. Najvýraznejšie prispelo k zvýšeniu výkonu, zvýraznenie syntaxe v dočasnom neviditeľnom *RichTextBoxe* pri otvorení súboru (implementované vo funkcii

"*highlightSyntaxOnOpen()*"). Tento text je potom jednoducho skopírovaný do textového modulu.

5.4 Modul verifikácie

Modul verifikácie je implementovaný v triede *verifikacia*. Skladá sa z dvoch častí a to z generátora pravdivostného vektora a z grafickej simulácie obvodu. Čo je nutné spomenúť, tak tento modul závislý od grafického modulu, keď pristupuje priamo k jeho objektom (hradlá a spojenia).

Hradlo je reprezentované všeobecnou triedou *Gate*, pričom však každý funkčne špecifický typ hradla (AND, OR, NAND, ...) je rozšírením tejto triedy *Gate*. Inak povedané, v triede *verifikacia*, je možné definovať nový objekt typu *Gate*, ktorý však nedokáže zmeniť hodnotu na výstupe tohoto hradla. Ak však pretypujem hradlo *Gate* na niektoré rozšírenie ako napr. *AND*, tak sa bude hradlo správať podľa predpísaného správania k danému typu hradla. Každý objekt, predstavujúci hradlo z grafického modulu, je potom pri otvorení verifikácie reprezentovaný pre účely simulácie pomocou rozšírení triedy *Gate* (AND, OR, NAND,...). Toto riešenie umožňuje naplno využiť výhody objektovo orientovaného programovania ako napr. polymorfizmus (, ktoré je vo verifikácii rozsiahle využívané).

Najdôležitejšou funkciou v celej verifikácii je funkcia *resolveCircuit()*. Pri zavolaní tejto funkcie sa vyhodnotia a uložia všetky hodnoty aktuálne platné pre celý obvod. Obvod sa postupnými iteráciami vyhodnocuje od vstupov po výstupy. Pri tvorbe pravdivostného vektora sa jednoducho na vstupy postupne priradujú rôzne hodnoty tak, aby sa otestovala každá možnosť (štandardným postupom pre tvorbu pravdivostnej tabuľky). Výsledné pravdivostné vektory sú potom hodnoty odčítané z výstupov obvodu. Dôvodom prečo sa v aplikácii neuvádza celá pravdivostná tabuľka je šetrenia miesta a zvýšenie prehľadnosti, keď napr. už pri ôsmich rôznych vstupoch by mala tabuľka až 256 riadkov (vektor je ale tiež dlhý až 256 znakov).

Pre pohodlné a pútavejšie overovania obvodu je pripravená grafická simulácia obvodu. Pri otvorení okna s vizualizáciou sa načíta obvod z grafického editora. Základ funkcionality je spoločný s generátorom pravdivostného vektora, ale hodnoty na vstupoch môže užívateľ kontrolovať a ovládať jednoduchým kliknutím myši (prepne aktuálnu hodnotu). Hodnoty v obvode je možné odčítať podľa farby spojenia, keď

červená znamená nula (low, false) a zelená jedna (high, true). Po každej zmene hodnoty vstupov sa aktualizuje obvod spomínanou funkciou *resolveCircuit()* a následne sa prekreslia spojenia.

6 Testovanie produktu

Testovanie aplikácie prebiehalo už od vytvorenia prvého prototypu programu. Testovala sa funkčnosť aplikácie a následne sa opravovali zistené nedostatky, prípadne sa aplikácia obohatila o nové nápady a vylepšenia.

Testy prebiehali ešte počas tvorby jednotlivých častí aplikácie, aby testovanie nezdržovalo vývoj častí programu na seba nadväzujúcich.

Testovanie bolo rozdelené na 4 väčšie celky. Testovanie pre modul textový editor. Testovanie pre modul grafický editor Drag 'n Drop. A testovanie pre modul verifikácie. Toto testovanie prebiehalo pre každý modul zvlášť. Následne vo štvrtej časti bola testovaná celá aplikácia so všetkými integrovanými modulmi, kde sa overovala vzájomná kompatibilita a interakcia samotných modulov medzi sebou. Výstupom všetkých testov boli akceptačné testy, ktoré boli prehľadne zobrazené v tabuľkách. Každý akceptačný test má priradené svoje identifikačné číslo ID, aby sa dalo ľahko zistiť k akej časti testovania patrí. Ďalej obsahuje názov testu, dátum uskutočnenia, úroveň splnenia testu, ktorú musí daný test dosiahnuť. Každý akceptačný test obsahuje autora časti, ktorá sa testuje a testovacie rozhranie. Ďalšie parametre testu je jeho účel a vstupné a výstupné podmienky testu. Nakoniec nasledujú jednotlivé kroky a akcie testu s očakávanými a skutočnými reakciami. Testovanie bolo rozdelené vždy podľa zložitosti. Od jednoduchších obvodov až po zložitejšie s množstvom vstupov, výstupov a hradiel. Týmto spôsobom sa nám podarilo odhaliť chyby čo najrýchlejšie.

Ako prvé sa testoval modul pre textový editor. Testovanie bolo zamerané na správne načítavanie zdrojového kódu zo súborov s viacerými formátmi ako napr. VHDL a BLIF. Ďalej nasledovalo správne uloženie napísaného zdrojového kódu do súboru s vybraným formátom. Sledovalo sa tiež správne zvýrazňovanie syntaxe podľa aktuálne vybraného súborového formátu. Testovala sa aj robustnosť súboru syntax.xml, v ktorom boli podmienky pre zvýraznenie syntaxe.

Druhá časť testovania sa týkala grafického editora Drag 'n Drop. Testovalo sa správne vykreslenie hradiel, vstupov, výstupov a prepojení, aby celkové zobrazenie logického obvodu bolo správne. Testovali sa všetky funkcie, ktoré tento modul

podporuje na editovanie logického obvodu. Súčasťou testov bolo aj správne zobrazovanie informácií v príslušných oknách grafického editora.

Tretia časť testovania sa zaoberala funkčnosťou modulu verifikácie. Tu sa skúmalo správne vypísanie pravdivostného vektora výstupnej funkcie logického obvodu a tiež správne vykreslenie logického obvodu s možnosťou zadávať hodnoty vstupov a tým sledovať meniace sa správanie jednotlivých vodičov a celkového výstupu obvodu. Toto testovanie sa uskutočnilo s viacerými logickými obvodmi s rôznym počtom vstupov, výstupov a hradiel. Výsledok sa porovnával s hodnotami v programe LOG, aby sme správne overili funkčnosť.

Štvrtá časť testovania skúmala prepojenie všetkých modulov a ich správnu funkcionálnosť. Testoval sa prevod načítaného VHDL a BLIF súboru do grafického modulu a následne do modulu verifikácie. V tejto časti sa testovala aj celková stabilita programu.

Testovanie prebiehalo dovtedy, kým všetky zistené chyby neboli odstránené. Všetky testy sú zapísané v akceptačných testoch a sú súčasťou dokumentácie riadenia.

7 Záver

Tento dokument obsahuje výsledky práce tímu číslo 3 Digitalists, ktorý bol vypracovaný v rámci predmetu Tímový projekt 1 a 2 na FIIT STU s názvom témy Prostredie pre návrh digitálnych systémov. V práci sme analyzovali viacero existujúcich riešení problému a zistili sme, že dosiaľ neexistuje vhodný modulárny systém poskytujúci komplexné prostredie pre návrh digitálnych systémov. Porovnali sme práce tímov z predchádzajúcich rokov, ktorých funkcionality sme následne otestovali a rozhodli sme sa vytvoriť vlastný modulárny systém používajúci prvky týchto projektov. Po následnej analýze jednotlivých súborových formátov a aktuálnych požiadaviek na systém sme sa rozhodli vytvoriť produkt poskytujúci textové a grafické rozhranie na vytváranie logických obvodov z jednotlivých hradiel, tieto obvody následne ukladať a konvertovať na vhodný súborový formát, pričom pridáme funkciu simulácie daného obvodu. Práca obsahuje dôkladnú analýzu problému, návrh nami vytvoreného systému a opis realizácie jednotlivých programových častí.

Výsledná aplikácia je navrhnutá pre jednoduché používanie. Samotný systém obsahuje 3 samostatné integrované moduly, ktoré medzi sebou komunikujú. V hlavnom okne programu tak môžeme jednoducho prepínať medzi tromi panelmi: Textový editor, Grafický editor Drag & Drop a panel pre Verifikáciu a Simuláciu.

Textový editor podporuje prácu s viacerými súborovými formátmi. Tieto súbory je možné v ňom vytvárať, editovať, načítať a ukladať. Taktiež podporuje zvýrazňovanie syntaxe pre viaceré známe súborové formáty.

Grafický editor nám umožňuje nakresliť logický obvod pomocou editačných prvkov alebo načítať obvod priamo z dostupného súboru VHDL alebo BLIF. V okne grafického editora je k dispozícii množstvo logických hradiel, ktoré môžeme využiť pri návrhu logického obvodu. Tiež nám zobrazuje prehľad o všetkých uzloch a prepojeniach v danom obvode.

Modul verifikácie nám poskytne možnosť výpisu pravdivostného vektora všetkých výstupov obvodu s možnosťou simulácie, kde si môžeme sami určiť logické hodnoty na vstupoch obvodu a sledovať správanie na vodičoch a výstupoch daného obvodu pomocou farebného odlíšenia.

Na začiatku práce na tímovom projekte sme si zvolili štyri všeobecné požiadavky: modularita, rozšíriteľnosť, grafické rozhranie a overiteľnosť. Všetky spomínané požiadavky výsledná aplikácia spĺňa a sú úspešne zrealizované.

Takmer všetky funkcionálne požiadavky, ktoré sme si stanovili na začiatku tohto semestrálneho projektu sa nám podarilo úspešne implementovať. Vo výslednom projekte by bolo možné doplniť do grafického editora možnosť tvorby vlastných hradiel a tiež pridať podporu viacerých súborových formátov. Tieto požiadavky sme mali stanovené, no v programe sa nenachádzajú. Program splnil spomínané požiadavky na modularitu a rozšíriteľnosť, takže v budúcnosti je možné tieto funkcie do aplikácie implementovať.

Pri riešení projektu sme získali cenné skúsenosti nielen s tvorbou rozsiahlejšieho modulárneho systému, ale aj s tímovou prácou, riadenia všetkých členov tímu, rozdeľovania zadaných úloh, vzájomnej komunikácie a spoločného riešenia problémov.

8 Použité skratky

PKSS - Počítačové a komunikačné systémy a siete

ASIC - Application-specific integrated circuit

BLIF - Berkeley Logic Interchange Format

KISS - Keep It Simple, Stupid

SLIF - Stanford Logic Interchange Format

VHDL - VHSIC hardware description language – VHSIC hardvér opisujúci jazyk

FPGA - Field-programmable gate array – programovateľné pole logických členov

CMOS - Complementary Metal Oxide Semiconductor - dopĺňajúci sa kov-oxid-polovodič

TTL – transistor transistor logic – tranzistorovo-tranzistorová logika

GUI – Graphical User Interface – grafické používateľské rozhranie

VLSI - Very-large-scale integration – veľmi veľká integrácia

BDD - binary decision diagram – binárny rozhodovací diagram

CUDD - Colorado University Decision Diagram – rozhodovací diagram z Coloradskej Univerzity

ADD - Algebraic Decision Diagram – algebraický rozhodovací diagram

ZBDD - Zero-suppressed Binary Decision Diagram – binárny rozhodovací diagram s potlačením núl

RD - rozhodovací diagram

XML - Extensible Mark-up Language – rozšíriteľný značkový jazyk

FSM - finite state machine – konečný automat

PLA - Programmable Logic Array – programovateľné logické pole

EQN - Equation format – rovnicový formát

WYSIWYG – What You See Is What You Get – doslova „čo vidíš, to dostaneš“

9 Použitá literatúra

- [1] ModelSim PE - Simulation and Debug, <http://model.com/content/modelsim-pe-simulation-and-debug> [16.1.2012]
- [2] Klogic, 2011. <http://www.a-rostin.de/> [16.1.2012]
- [3] Černý, Michal, 2010. Tvorba elektrických obvodů a simulace. <http://www.root.cz/clanky/tvorba-elektricky-obvodu-a-simulace/> [16.1.2012]
- [4] Machálková, Katarína, 2004. TkGate - návrh a simulace logických obvodů snadno a rychle. <http://www.root.cz/clanky/tkgate-navrh-simulace-logicky-obvodu/> [16.1.2012]
- [5] Hansen, Jeffery P. TKGate. <http://www.tkgate.org/> [16.1.2012]
- [6] MVSIS: Logic Synthesis and Verification. <http://embedded.eecs.berkeley.edu/mvsis/> [16.1.2012]
- [7] Somenzi, Fabio, 2009, CUDD: CU Decision Diagram Package. <http://vlsi.colorado.edu/~fabio/CUDD/> [16.1.2012]
- [8] CUDD - Tutorial Pages. <http://www.cs.uleth.ca/~rice/cudd.html> [16.1.2012]
- [9] Schreiber, Ethan L. 2008. A CUDD Tutorial. http://www.cs.ucla.edu/~ethan/documents/schreiber_cudd_tutorial.pdf
- [10] YANG, C., CIESIELSKI, M.: BDS: a BDD-Based Logic Optimization System. Amherst: Dept. of Electrical & Computer Engineering, University of Massachusetts, July 2002. ISBN:1-58113-187-9. Bds-tcad02.pdf.
- [11] CHYTIL, R., JÁNOŠ, M., LORINCZ, T., TAKACS, T., VIRKLER, R.: Prostredie pre návrh digitálnych systémov (Tímový projekt). Bratislava: FIIT STU, 2011. 94 s.
- [12] Berkeley Logic Interchange Format (BLIF). <http://www1.cs.columbia.edu/~cs4861/s07-sis/blif/index.html> [16.1.2012]
- [13] File Formats for SIS Package. http://www2.engr.arizona.edu/~veda/cadtools/sis/tutorials/sis_fileformats.doc [16.1.2012]
- [14] Binary Decision Diagram. http://en.wikipedia.org/wiki/Binary_decision_diagram [16.1.2012]

10 Prílohy

Táto kapitola obsahuje prílohy dokumentácie

10.1 Príloha A - Používateľská príručka

Príloha A - Používateľská príručka

Táto príručka slúži k predstaveniu programu Logic Designer používateľovi a k pochopeniu princípu činnosti programu.

A.1 Predstavenie programu Logic Designer

Program Logic Designer predstavuje funkčný prototyp systému pre návrh digitálnych systémov. Je to modulárny program, ktorý obsahuje viacero pluginov pre návrh a verifikáciu digitálnych systémov. Umožňuje prácu s textovým editorom, kreslenie logických obvodov a verifikáciu navrhnutých riešení.

Základné vlastnosti programu:

- Tvorba zdrojového kódu v textovom editore so zvýrazňovaním syntaxe
- Tvorba logických sietí v grafickom editore
- Prepis logickej siete z grafického editora do príslušného kódu v textovom editore a naopak
- Simulácia na overenie správania sa obvodu
- Konverzia súborových formátov

A.1.1 Dostupnosť programu

Program je možné stiahnuť na stránke tímu:

<http://labss2.fiit.stuba.sk/TeamProject/2011/team03pkss>

A.1.2 Inštalácia

Inštalácia prebieha prostredníctvom sprievodcu inštalácie do vybraného priečinka. Tým je aplikácia plne funkčná a pripravená na používanie. Program nerobí zásah do systémových registrov.

A.1.3 Hardvérové požiadavky

Procesor Intel alebo AMD s taktovacou frekvenciou 300 MHz a viac

- 128 MB pamäte RAM a viac
- Monitor s rozlíšením 800x600 a vyššie
- Klávesnica a myš
- 35 MB voľného miesta na pevnom disku

A.1.4 Softvérové požiadavky

Aplikácia má minimálne požiadavky na systém na ktorom sa bude spúšťať. Je potrebný počítač s operačným systémom Windows XP a vyššie.

Softvérové nároky :

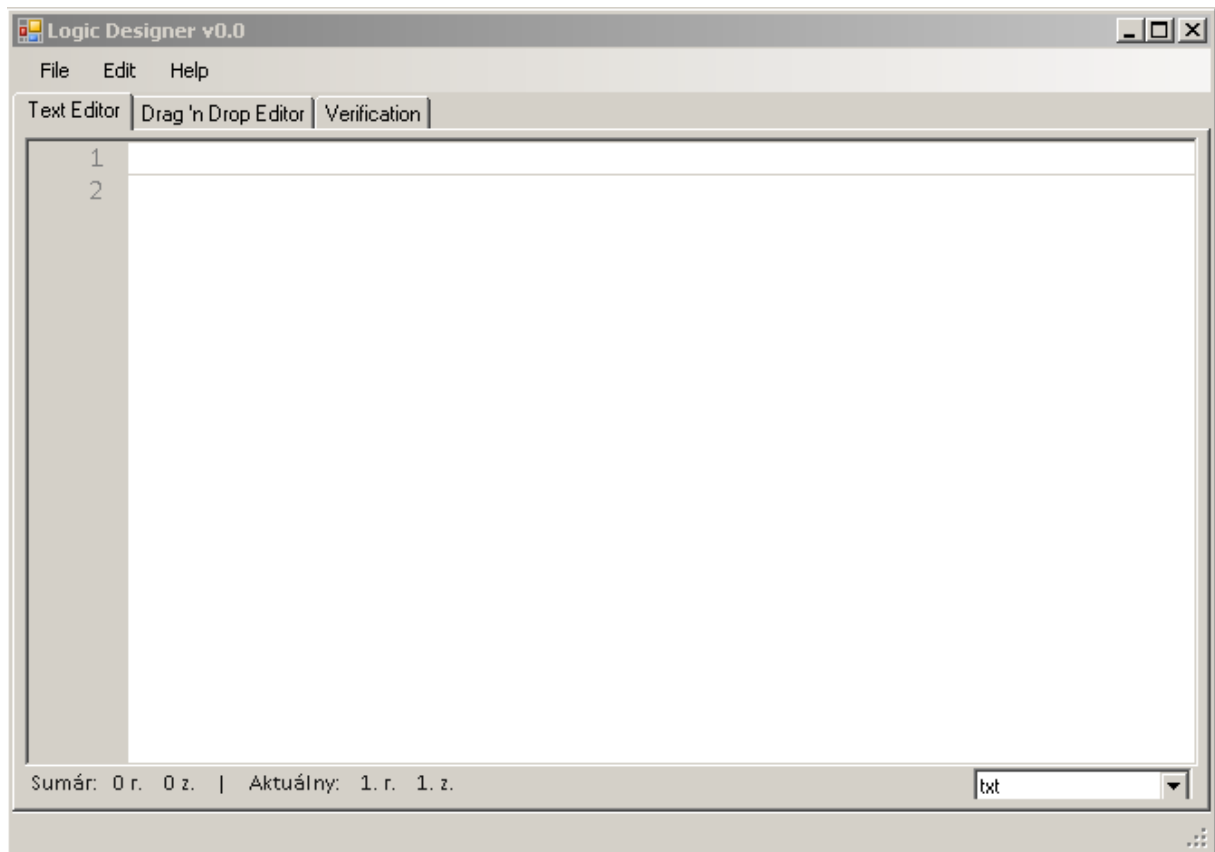
- Operačný systém Windows XP , Vista, Windows 7
- Platforma .NET 4.0 a vyššia

A.2 Používateľské prostredie

Táto kapitola popisuje prácu s programom Logic Designer. Obsahuje popis jednotlivých modulov systému a vysvetľuje účel ovládacích prvkov. Na tomto mieste je uvedený popis okna hlavného programu. Práca s modulmi je uvedená v samostatných kapitolách.

Po spustení programu sa zobrazí hlavné okno s menu s tromi prepínacími panelmi: Text Editor, Drag'n Drop Editor a Verification.

Primárne je nastavený ako vybraný prepínací panel Text Editor.

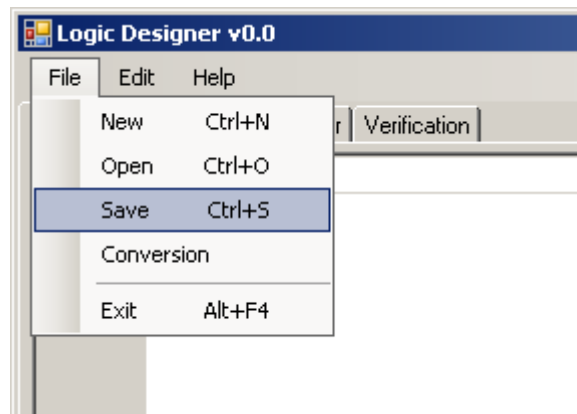


Obr. č. 1: Text Editor

A.2.1 Menu programu

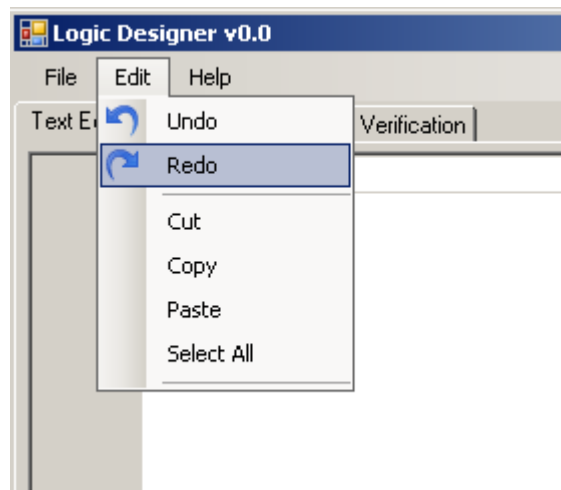
Menu programu obsahuje tieto položky:

- **File**, po kliknutí sa zobrazia nasledujúce možnosti s príslušnými klávesovými skratkami:
 - New – vytvorenie nového súboru (Ctrl + N)
 - Open – otvorí súbor v danom formáte (Ctrl + O)
 - Save – uloží súbor do vybraného adresára, názov formátu si môžeme zvoliť buď BLIF, PLA KISS, Verilog alebo VHDL (Ctrl + S)
 - Conversion – zavolá sa externý program, ktorý umožňuje konvertovanie súborových formátov ako SLIF, BLIF, PLA a EQN, pričom výsledok sa uloží do príslušného formátu na vybrané miesto.
 - Exit - ukončenie programu (Alt + F4 S)



Obr. č. 2: Menu File

- **Edit**, po kliknutí sa zobrazia nasledujúce možnosti:
 - Undo – krok späť
 - Redo – krok dopredu
 - Cut - vystrihnúť
 - Copy - kopírovať
 - Paste - vložiť
 - Select All – vybrať všetko

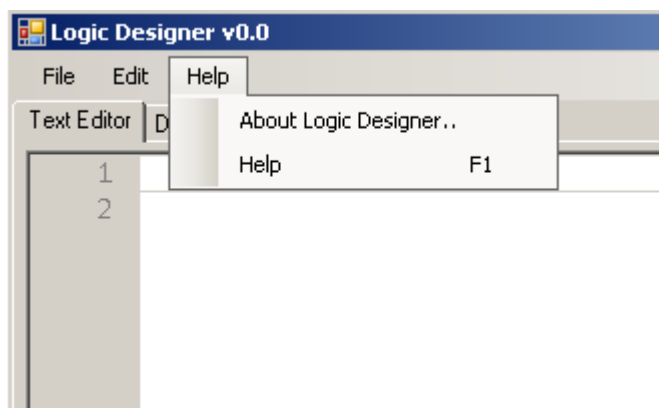


Obr. č. 3: Menu Edit

Funkcie menu Edit sú dostupné len v module Text Editor

- **Help**, po kliknutí sa zobrazia nasledovné možnosti:

- Help – zobrazí pomocníka programu (F1)
- About Logic Designer – Zobrazí informácie ohľadom programu



Obr. č. 4: Menu Help

A.3 Popis Modulov

Táto kapitola sa zaoberá popisom jednotlivých modulov (pluginov) systému, z ktorých sa skladá hlavný program.

Program obsahuje nasledujúce moduly:

- Modul textového editora – Text Editor
- Modul grafického editora – Drag'n Drop Editor
- Modul pre verifikáciu – Verification

A.3.1 Text Editor

Textový editor poskytuje rôzne funkcie na prácu so zdrojovým kódom logického obvodu. Slúži predovšetkým na písanie zdrojového kódu v príslušnom súborovom formáte a prácu s už vytvorenými súbormi.

Textový editor poskytuje nasledovné funkcie:

- Vytvoriť nový súbor

- Otvoriť/editovať/uložiť súbor
- Kontrola a zvýrazňovanie syntaxe súborových systémov BLIF, PNML KISS, Verilog alebo VHDL
- Vykresliť logický obvod do modulu Drag´n Drop z napísaného zdrojového kódu

```

1 ----- NAND 2 vstupovy -----
2 entity oNAND2 is
3     port(IN1:in BIT; IN2:in BIT; OUT1:out BIT);
4 end oNAND2;
5 architecture sprNAND2 of oNAND2 is
6 begin
7     OUT1 <= IN1 nand IN2;
8 end sprNAND2;
9 ----- AND 2 vstupovy -----
10 entity oAND2 is
11     port(IN1:in BIT; IN2:in BIT; OUT1:out BIT);
12 end oAND2;
13 architecture sprAND2 of oAND2 is
14 begin
15     OUT1 <= IN1 and IN2;
16 end sprAND2;
17 -----
18 entity obvod is
19     port ( a:in Bit; b:in Bit; c:in Bit; negC:out Bit);
20 end obvod;
21 architecture Spravanie of obvod is
22     component cNAND2
23         port(IN1:in Bit; IN2:in Bit; OUT1:out Bit);
24     end component;
25     component cAND2
26         port(IN1:in Bit; IN2:in Bit; OUT1:out Bit);
27     end component;
28     for xNAND2_a: cNAND2 use entity Work.oNAND2(sprNAND2);
29     for xNAND2_b: cNAND2 use entity Work.oNAND2(sprNAND2);
30     for xNAND2_c: cNAND2 use entity Work.oNAND2(sprNAND2);

```

Sumár: 35 r. 1101 z. | Aktuálny: 1. r. 1. z.

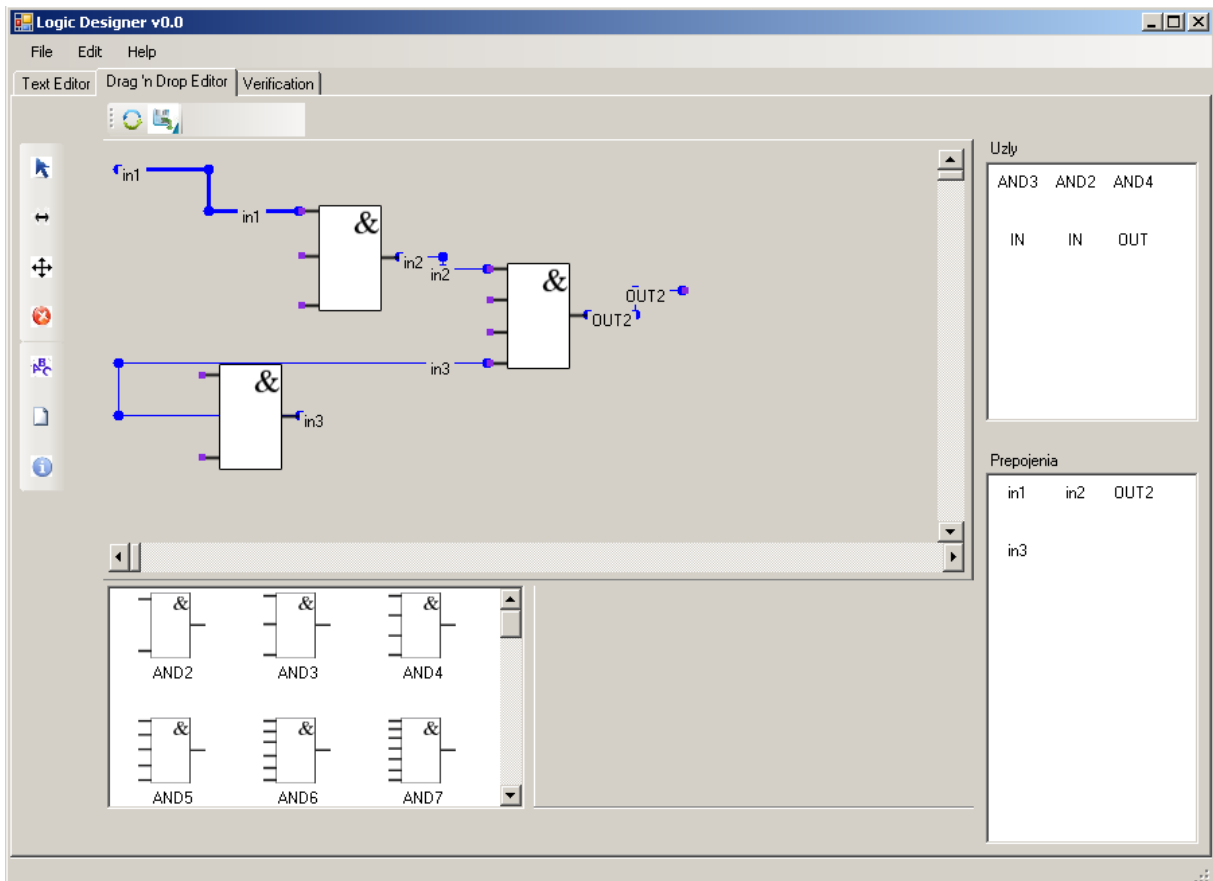
Obr. č. 5: Textový editor

Textový editor v spodnej časti okna zobrazuje informácie o danom kóde. Ako počet riadkov, počet znakov a aktuálny riadok a znak. Taktiež je možnosť v pravom dolnom rohu si zvoliť súborový formát, podľa ktorého sa má zvýrazňovať syntax kódu. Pokiaľ chceme previesť zdrojový kód VHDL alebo BLIF na grafickú podobu, stačí prepnúť na panel Drag ´n Drop editor, kde sa logický obvod vykreslí.

A.3.2 Drag´n Drop Editor








Grafický Drag'n Drop editor slúži na prácu s logickými obvodmi. Prostredníctvom tohto modulu má užívateľ sprístupnené nasledovné funkcie:

- Nakresliť vlastný logický obvod z dostupných hradíel
- Vykresliť a editovať logický obvod z uloženého súboru, alebo z textového editoru
- Uložiť nakreslený logický obvod do súboru VHDL alebo BLIF





Obr. č. 6: Grafický editor Drag'n Drop

A.3.2.1 Popis bočného menu grafického editora Drag'n Drop

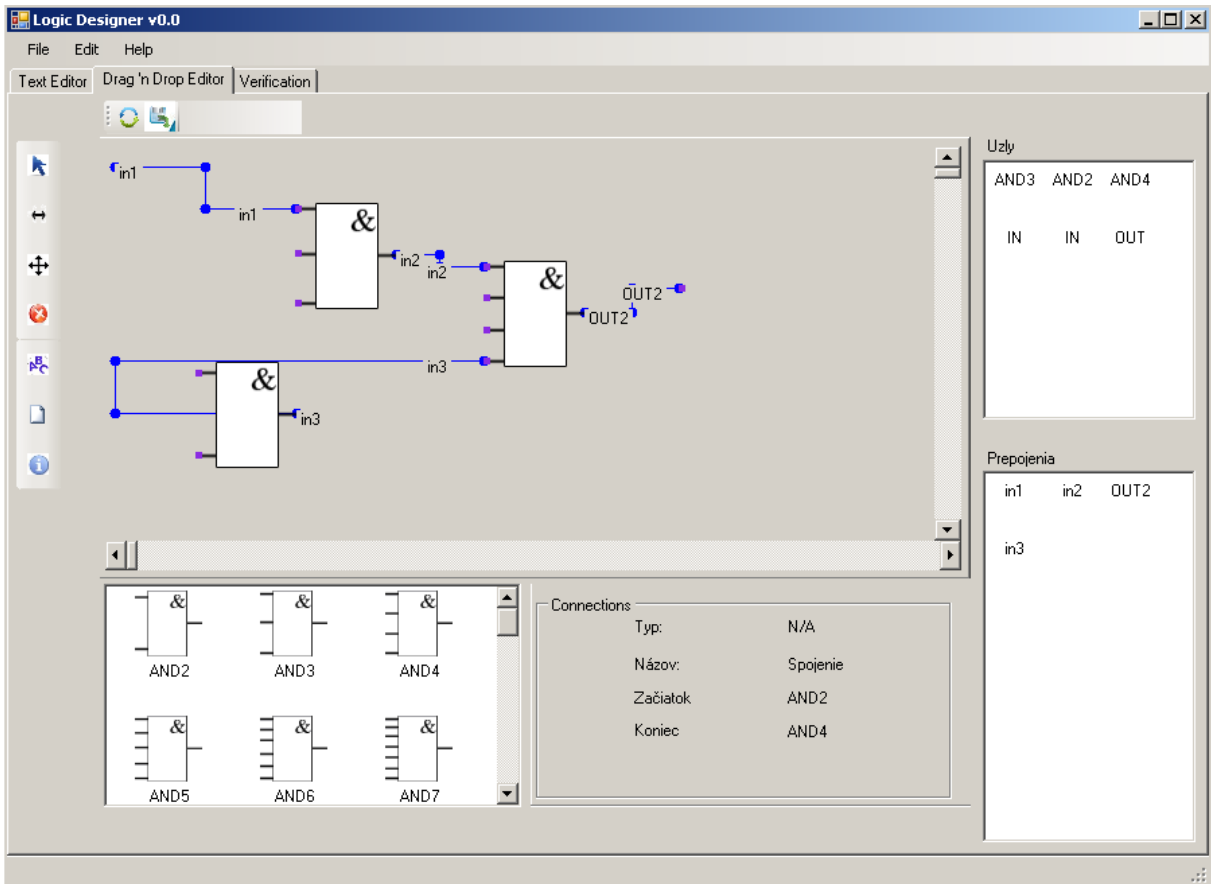
-  **Označ** – slúži na označenie objektov
-  **Spoj** – spája vstupy (žltý štvorček) hradiel alebo celého obvodu s výstupmi (modrý štvorček) jednotlivých hradiel alebo celého obvodu
-  **Posuň** – posúva objekty po pracovnej ploche, umožňuje zmeniť rozloženie častí kreslenej schémy
-  **Vymaž** – po kliknutí na toto tlačidlo a následne na objekt alebo spoj sa vymaže zo schémy
-  **Zobraz / skry menovky** – slúži na zobrazenie menoviek daného obvodu
-  **Nový** – funkcia otvorí čistý editor na prácu
-  **Informácie** – slúži na výpis informácií o danom obvode

A.3.2.2 Popis horného menu grafického editora Drag ´n Drop

Tieto 2 prvky sú zautomatizované v programe, takže nie je potrebné ich používať.

-  **Ulož** – uloží logický obvod do jadra programu
-  **Refresh** – použije sa len v prípade načítania iného logického obvodu do Drag ´n Drop editora.

A.3.2.3 Popis pracovných okien grafického editora Drag ´n Drop



Obr. č. 7: Grafický editor Drag'n Drop rozloženie okien

V pravej hornej časti programu sa nachádza okno Uzly, v ktorom sa vypisuje zoznam všetkých vykreslených hradiel, vstupov a výstupov.

Pod týmto oknom sa nachádza okno Prepojenia, kde sa vypisujú jednotlivé spojenia logického obvodu.

Vedľa okna Prepojenia je informačné okno, ktoré zobrazuje informácie týkajúce sa k miestu, na ktoré ukazujeme kurzorom. V prípade, že ukazujeme na spojenie v logickom obvode, tak sa zobrazia informácie o tomto spojení. Typ sa udáva len pri logických hradlách. Názov je Spojenie a začiatok znamená uzol, z ktorého toto spojenie vychádza a koniec, v ktorom uzle spojenie končí.

Connections	
Typ:	N/A
Názov:	Spojenie
Začiatok	IN
Koniec	AND3

Obr. č. 8: Informácie k ukazovanému spojeniu

V prípade nastavenia kurzora na logického hradla sa zobrazia informácie ako typ hradla, jeho názov a počet vstupov a výstupov ukazovaného hradla.

Nodes	
Typ	AND3
Názov	AND3
Počet vstupov	3
Počet výstupov	1

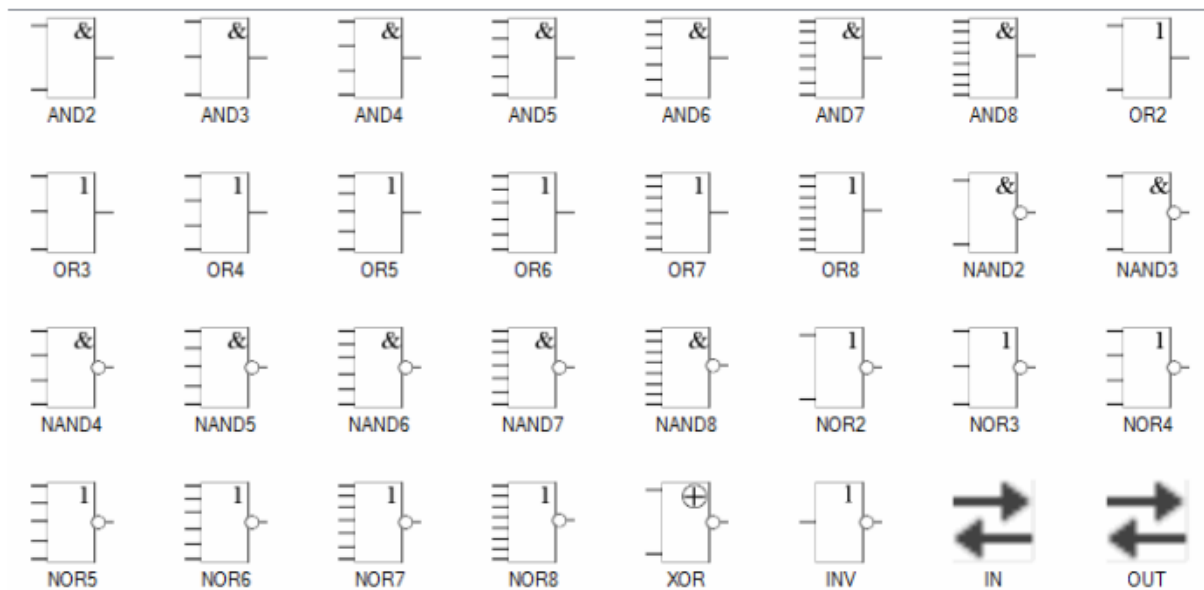
Obr. č. 9: Informácie k ukazovanému hradlu

V dolnej časti programu je okno, kde si užívateľ môže vybrať z množstva preddefinovaných logických členov, ktoré chce použiť pri návrhu logického obvodu.

Je možné si vybrať s nasledujúcich typov objektov:

- **AND** – obvod vykonávajúci operáciu logického súčinu, program ponúka možnosť vybrať si dvojjstupový až osemvstupový AND
- **OR** – obvod vykonávajúci operáciu logického súčtu, program ponúka možnosť vybrať si dvojjstupový až osemvstupový OR
- **NAND** – obvod vykonávajúci operáciu negácie logického súčinu, program ponúka možnosť vybrať si dvojjstupový až osemvstupový NAND
- **NOR** – obvod vykonávajúci operáciu negácie logického súčtu, program ponúka možnosť vybrať si dvojjstupový až osemvstupový NOR
- **XOR** – obvod vykonávajúci operácie neekvivalencie, exkluzívny OR
- **INV** – invertor, invertuje vstupnú hodnotu na výstup
- **IN** – vstup celého obvodu, vstupný port

- **OUT** – výstup celého obvodu, výstupný port



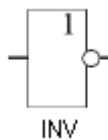
Obr. č. 10: Zoznam logických hradieľ v grafickom editore Drag ´n Drop

A.3.2.1 Pravdivostné tabuľky použitých logických členov

Logické hradlo invertor


Prvok, ktorý obráti logickú hodnotu vstupu.

Funkcia: $Y = \neg A$

	Vstup A	Výstup Y
	0	1
	1	0

Dvojvstupové hradlo AND:

Funkcia: $A \text{ and } B = Y$



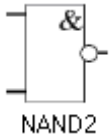
Vstup A	Vstup B	Výstup Y
0	0	0
0	1	0
1	0	0
1	1	1

Viacstupové hradlo AND:

Na výstupe Y bude logická hodnota 1, len v prípade, že všetky vstupy budú mať hodnotu 1.

Dvojjstupové hradlo NAND:

Funkcia: $A \text{ nand } B = Y$



Vstup A	Vstup B	Výstup Y
0	0	1
0	1	1
1	0	1
1	1	0

Viacstupové hradlo NAND:

Na výstupe Y bude logická hodnota 0, len v prípade, že všetky vstupy budú mať hodnotu 1.

Dvojjstupové hradlo OR:

Funkcia: $A \text{ or } B = Y$



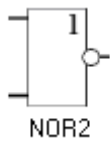
Vstup A	Vstup B	Výstup Y
0	0	0
0	1	1
1	0	1
1	1	1

Viacstupové hradlo OR:

Na výstupe Y bude logická hodnota 0, len v prípade, že všetky vstupy budú mať hodnotu 0.

Dvojjstupové hradlo NOR:

Funkcia: $A \text{ nor } B = Y$



Vstup A	Vstup B	Výstup Y
0	0	1
0	1	0
1	0	0
1	1	0

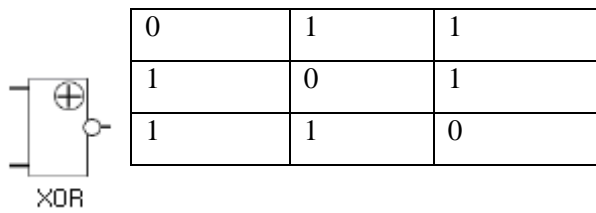
Viacstupové hradlo NOR:

Na výstupe Y bude logická hodnota 1, len v prípade, že všetky vstupy budú mať hodnotu 0.

Dvojjstupové hradlo XOR:

Funkcia: $A \text{ xor } B = Y$

Vstup A	Vstup B	Výstup Y
0	0	0

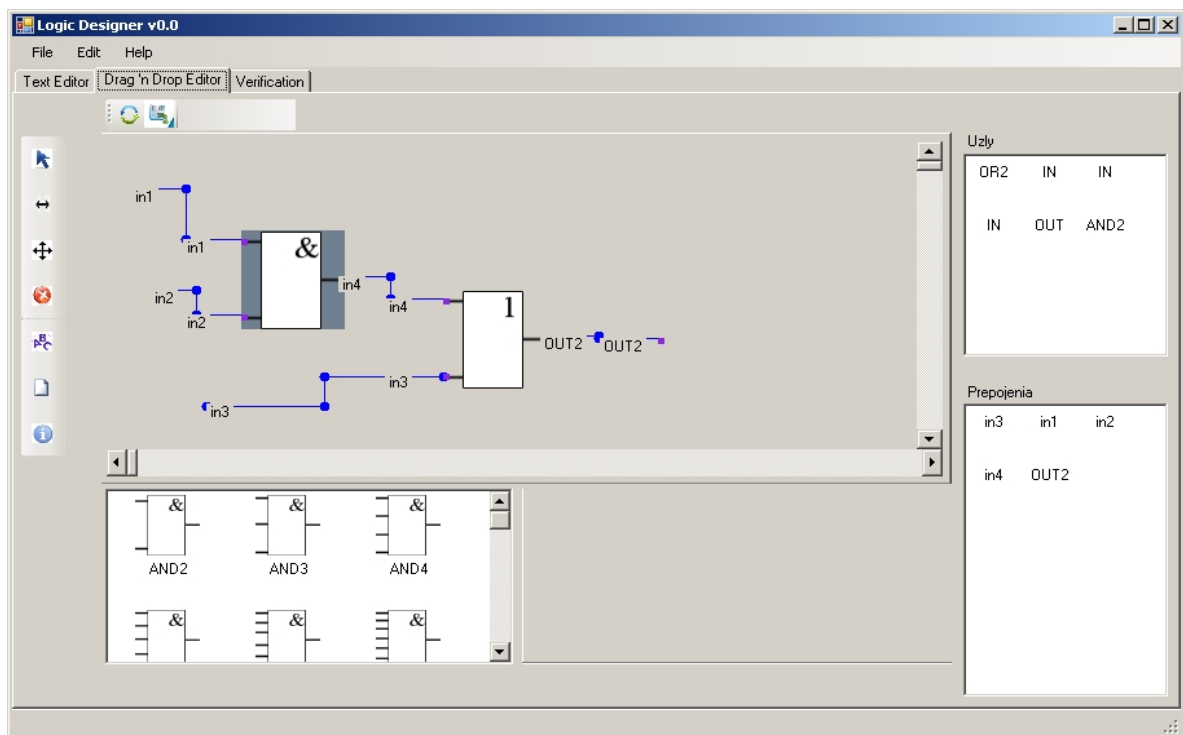


XOR je binárna operácia, čiže je definovaná len na dvoch vstupoch.

Na výstupe Y bude logická hodnota 1, len v prípade, keď sú jeho vstupy rozdielne.

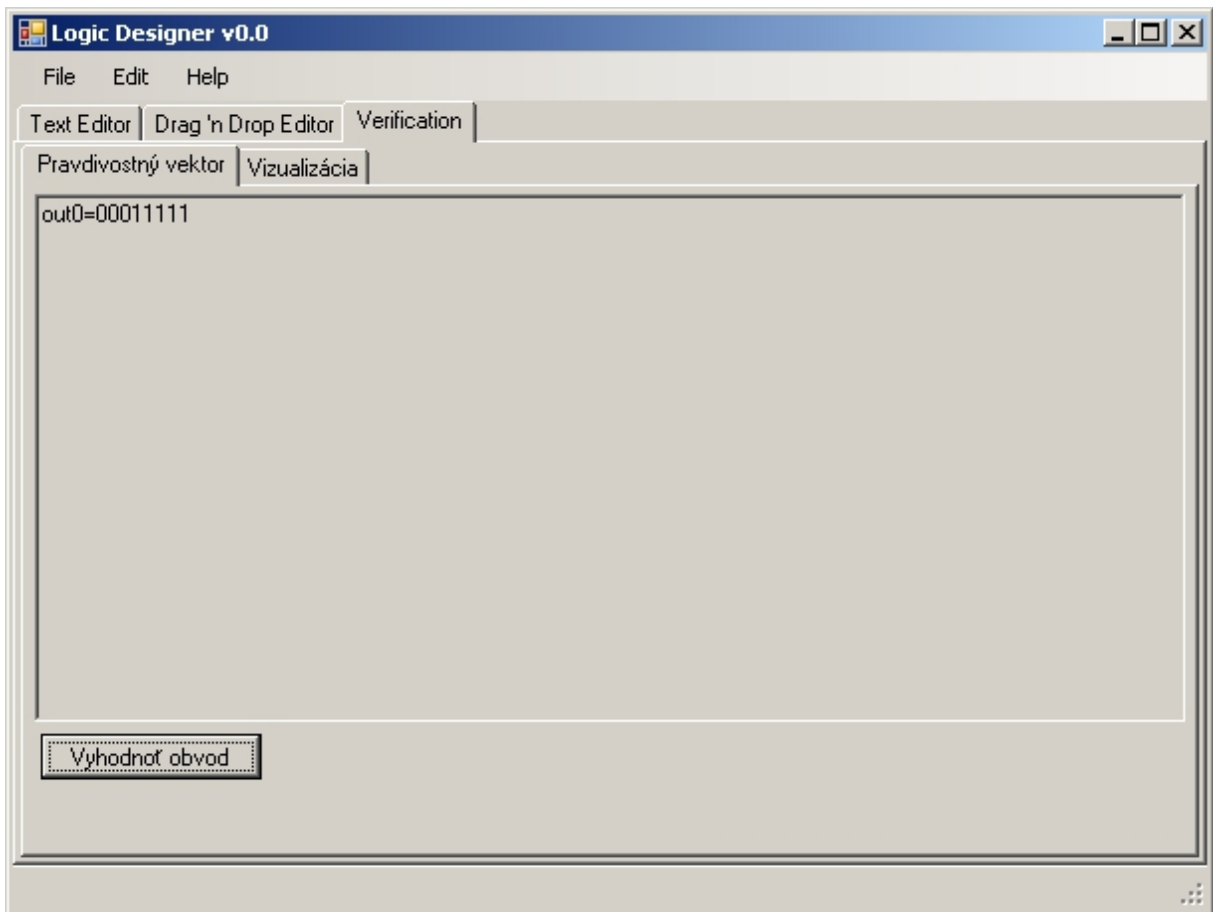
A.3.3 Verification – modul pre verifikáciu

Modul verifikácie slúži na vypísanie výstupného vektora podľa pravdivostnej tabuľky z logického obvodu, ktorý sa nachádza v grafickom editore Drag 'n Drop. Ďalšou funkciou tohto modulu je aj simulácia logického obvodu, kedy si používateľ určí logické hodnoty na vstupe obvodu a môže sledovať zmeny v správaní na jednotlivých vodičoch obvodu.



Obr. č. 11: Zobrazený obvod v Drag 'n Drop editore

V prípade, že máme v grafickom editore zobrazený logický obvod, tak stačí prepnúť panel Verification a stlačiť tlačidlo Vyhodnoť obvod.

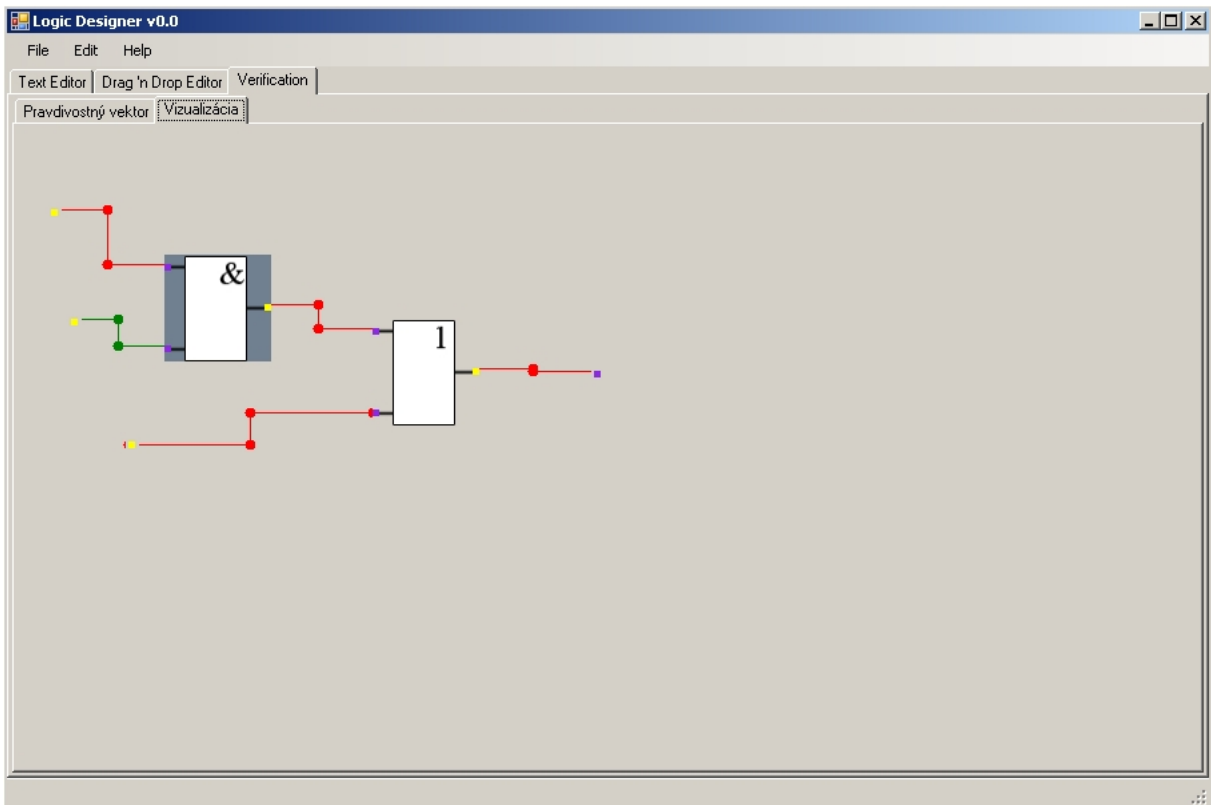


Obr. č. 12: Modul Verification a panel Pravdivostný vektor

Následne sa v tabuľke zobrazia všetky výstupy z logického obvodu s pravdivostným vektorom. V uvedenom príklade, má obvod 3 vstupy a jeden výstup. Preto je v aplikácii zobrazený jeden výstup out0 , ktorý má prislúchajúci 8 bitový vektor, čo predstavuje 2^3 kombinácií 0 a 1.

Vstup:	000	001	010	011	100	101	110	111
Výstup out0:	0	0	0	1	1	1	1	1

Tab. č. 1: Pravdivostná tabuľka



Obr. č. 13: Modul Verification a panel Vizualizácia

Na panely vizualizácia sa nám zobrazí celkový logický obvod s farebnými vodičmi, kde červená symbolizuje logickú 0 a zelená logickú 1. Používateľ môže po kliknutí na vstup zmeniť logickú hodnotu a sledovať správanie jednotlivých vodičov ako aj dôsledok zmeny na celkový výstup.