

PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Program pro tvorbu logických obvodů



2014

Jan Novák

Anotace

Práce se zabývá vytvořením aplikace, která umožňuje uživateli kreslit schémata základních logických obvodů. Nakreslený obvod je možné dále testovat pomocí interaktivně nastavovaných vstupů uživatelem, nebo generováním pravdivostní tabulky. Aplikace byla navržena s ohledem na to, že by mohla sloužit jako učební pomůcka.

Rád bych poděkoval vedoucímu práce Mgr. Jiřímu Zaccalovi, Ph.D. za odborné vedení, věcné připomínky a rady a za ochotu vytvořenou aplikaci testovat.

Obsah

1. Úvod	8
2. Logické obvody	9
2.1. Logické proměnné a funkce	9
2.1.1. Logické funkce jedné proměnné	10
2.1.2. Logické funkce dvou proměnných	10
2.1.3. Logické funkce více proměnných	10
2.1.4. Booleova algebra	10
2.1.5. Pravdivostní tabulka	12
2.1.6. Vytváření logických funkcí	13
2.2. Kombinační logické obvody	13
2.2.1. Základní logické členy	13
2.3. Sekvenční logické obvody	17
2.3.1. Klopný obvod RS	18
2.3.2. Synchronní klopný obvod RS (RST)	19
2.3.3. Paměťový člen D	19
2.3.4. Klopný obvod JK	20
3. Implementace aplikace	21
3.1. Programovací jazyk a použité nástroje	21
3.2. Objektový návrh	21
3.2.1. Třída LPart	22
3.2.2. Třída Gate	23
3.2.3. Třída GPart	23
3.2.4. Třída Wire	24
3.2.5. Třída WorkPlace	24
3.3. Vytvoření grafických objektů a jejich použití	26
4. Popis aplikace	27
4.1. Instalace	27
4.2. Panel nástrojů	28
4.3. Hlavní nabídka	29
4.4. Nástroje k návrhu obvodu	30
4.5. Testování obvodu	32
4.6. Generování pravdivostní tabulky	32
4.7. Práce se soubory	33
4.8. Export obvodu do souboru	33
4.9. Historie akcí	34
4.10. Přizpůsobení zobrazení	34
4.11. Požadavky na systém	34

Závěr	35
Conclusions	36
Reference	37
A. Obsah přiloženého CD/DVD	38

Seznam obrázků

1.	Příklad zapojení klopného obvodu RS pomocí hradel NAND . . .	18
2.	Hlavní okno programu	27
3.	Přidání nového prvku na plátno	30
4.	Ukázka kreslení vodiče	31
5.	Manuální testování obvodu	32
6.	Vygenerovaná pravdivostní tabulka	33

Seznam tabulek

1.	Funkce jedné proměnné	10
2.	Funkce dvou proměnných	11
3.	Zápis funkcí pomocí pravdivostních tabulek	12
4.	Schematické značky a pravdivostní tabulka funkce AND	14
5.	Schematické značky a pravdivostní tabulka funkce OR	15
6.	Schematické značky a pravdivostní tabulka funkce NOT	15
7.	Schematické značky a pravdivostní tabulka funkce AND	15
8.	Schematické značky a pravdivostní tabulka funkce NOR	16
9.	Schematické značky a pravdivostní tabulka funkce XOR	16
10.	Schematické značky a pravdivostní tabulka funkce XNOR	16
11.	Schematická značka klopného obvodu RS a pravdivostní tabulka .	18
12.	Schematická značka klopného obvodu RST a pravdivostní tabulka	19
13.	Schematická značka klopného obvodu D a pravdivostní tabulka . .	19
14.	Schematická značka klopného obvodu JK a pravdivostní tabulka .	20
15.	Přehled tlačítek na panelu nástrojů	28
16.	Struktura programové nabídky	29

1. Úvod

Logické obvody jsou dnes všude kolem nás. Ať už jsou to počítače, mobilní telefony nebo automatické pračky, zařízení obsahující logické obvody nás obklopují na každém kroku a ulehčují nám život. Tato zařízení samozřejmě dnes obsahují zpravidla složité integrované obvody, ale i ty, jsou složeny pouze z těch základních, jednoduchých logických obvodů.

Cílem této práce je naprogramovat aplikaci pro návrh logických obvodů pomocí základních logických prvků. Uživatel této aplikace má tak možnost si vyzkoušet fungování logických obvodů od těch úplně nejjednodušších, postupně s nimi experimentovat a sledovat jejich chování. Aplikace je tak vhodná zejména jako učební pomůcka při výuce logických obvodů, proto uživatelské rozhraní a celkový koncept programu je navržen tak, aby práce s ním byla intuitivní a co nejjednodušší.

I když dnes existuje podobných programů desítky a některé z nich nabízejí rozsáhlé možnosti, je málo takových, které nabízejí jednoduché rozhraní a možnost experimentovat s obvody i pro úplného začátečníka. Aplikace provádí vyhodnocení obvodu po každé změně okamžitě, což bude jistě vhodné pro uživatele studující princip fungování logických obvodů.

Zpracovaná aplikace umožňuje sestavování logických obvodů libovolného typu z těch nejzákladnějších logických členů, tj. logických hradel NOT, AND, NAND, OR, NOR, XOR a XNOR. Z toho důvodu se budeme v prvních kapitolách věnovat popisu logických proměnných, funkcí a obvodů, Booleově algebře a základních logických členů, dále metodám optimalizace logických obvodů a v dalších kapitolách pak už konkrétní implementaci vytvořené aplikace.

2. Logické obvody

Logickým, číslicovým či digitálním obvodem rozumíme množinu logických členů, které pracují se dvěma diskrétními (nespojitémi) stavy. Tyto obvody lze definovat množinou vstupů, výstupů a vnitřních stavů, mezi kterými obvod postupně přechází a mezi kterými existují funkční vazby.

Logické obvody rozdělujeme do dvou hlavních skupin :

- *Kombinační logické obvody* se vyznačují tím, že hodnota každého z výstupů je přímo závislá na aktuálním nastavení vstupů, tj. jediné kombinaci vstupních proměnných odpovídá jediná výstupní kombinace.[1]
- *Sekvenční logické obvody* na rozdíl od kombinačních obvodů mají hodnoty svých výstupů závislé nejen na aktuálním nastavení vstupů, ale i na předchozích stavech, ve kterých se obvod nacházel. Pro jedinou kombinaci vstupních proměnných tak může odpovídat více výstupních kombinací.[1] Lze tedy říci, že sekvenční obvod se od kombinačního liší tím, že je schopen si „pamatovat“ předchozí stavy.

2.1. Logické proměnné a funkce

Logická proměnná je veličina, která může nabývat pouze dvou hodnot označovaných logická nula nebo logická jedna, případně můžeme tyto stavy chápat jako „pravda“ – „nepravda“. Přechod mezi hodnotami zároveň nemůže být spojitý. Formálně tedy platí axiomy:[1]

$$\begin{aligned}x &= 0 \text{ jestliže } x \neq 1 \\x &= 1 \text{ jestliže } x \neq 0\end{aligned}$$

Logická funkce n logických proměnných může tak jako všechny její proměnné nabývat pouze dvou hodnot – logická nula nebo logická jedna. Proto můžeme logickou funkci považovat zároveň za logickou proměnnou, protože může být argumentem jiné logické funkce. Logickou funkci y logických proměnných $x_1, x_2 \dots x_n$ můžeme vyjádřit vztahem

$$y = f(x_1, x_2 \dots x_n),$$

kde počet argumentů n může být 1, 2 případně více než 2 a pak mluvíme o funkci jedné, dvou, nebo více proměnných.

2.1.1. Logické funkce jedné proměnné

Logickou funkci y jedné proměnné x můžeme zapsat vztahem $y = f(x)$. Takové funkce existují pouze čtyři. Jsou to *nulová funkce*, *jednotková funkce*, *identická funkce* a *inverzní funkce (negace)* a jsou popsány v následující tabulce.

x	y
0	0
1	0

nulová
funkce

x	y
0	1
1	1

jednotková
funkce

x	y
0	0
1	1

identická
funkce

x	y
0	1
1	0

inverzní
funkce

Tabulka 1. Funkce jedné proměnné

2.1.2. Logické funkce dvou proměnných

Pro dvě proměnné x_1, x_2 můžeme vyjádřit celkem 16 různých logických funkcí a jsou popsány v tabulce 2. Z těchto funkcí považujeme za důležité především konjunkci a disjunkci (a jejich negace), protože ty se používají v Booleově algebře a budeme se jim dále věnovat.

2.1.3. Logické funkce více proměnných

Obecně pro n proměnných je možné vyjádřit $(2^2)^n$ logických funkcí. Vyjadřování logických funkcí s více proměnnými bývá často nepřehledné a v praxi se příliš nevyužívá. Proto takové funkce zpravidla skládáme z funkcí dvou proměnných, kterými se zabývá i Booleova algebra.

2.1.4. Booleova algebra

Algebraická struktura popisující vztahy mezi logickými proměnnými se nazývá Booleova algebra¹ a využívá tři základní logické operace: negace, logický součet a logický součin. Booleova algebra nám slouží k práci s algebraickými výrazy a jejich úpravě. Vhodnou úpravou zápisu logické funkce můžeme funkci minimalizovat a sestavit tak logický obvod z menšího množství logických prvků při zachování stejné funkcionality obvodu.

¹podle irského matematika George Boolea

x	y	y
0	0	0
0	1	0
1	0	0
1	1	0

nulová funkce
 $y = 0$

x	y	y
0	0	1
0	1	1
1	0	1
1	1	1

jednotková funkce
 $y = 1$

x	y	y
0	0	0
0	1	0
1	0	1
1	1	1

identická funkce
 $y = x_1$

x	y	y
0	0	0
0	1	1
1	0	0
1	1	1

identická funkce
 $y = x_2$

x	y	y
0	0	0
0	1	0
1	0	1
1	1	0

přímá inhibice
 $y = x_1 \bar{x}_2$

x	y	y
0	0	0
0	1	1
1	0	0
1	1	0

zpětná inhibice
 $y = \bar{x}_1 x_2$

x	y	y
0	0	1
0	1	0
1	0	0
1	1	1

ekvivalence
 $y = x_1 \equiv x_2$

x	y	y
0	0	1
0	1	0
1	0	0
1	1	1

neekvivalence
 $y = \bar{x}_1 \equiv \bar{x}_2$

x	y	y
0	0	1
0	1	1
1	0	0
1	1	1

přímá implikace
 $y = x_1 \Rightarrow x_2$

x	y	y
0	0	1
0	1	0
1	0	1
1	1	1

zpětná implikace
 $y = x_1 \Leftarrow x_2$

x	y	y
0	0	1
0	1	1
1	0	0
1	1	0

negace
 $y = \bar{x}_1$

x	y	y
0	0	1
0	1	0
1	0	1
1	1	0

negace
 $y = \bar{x}_2$

x	y	y
0	0	0
0	1	0
1	0	0
1	1	1

logický součin
 $y = x_1 x_2$

x	y	y
0	0	0
0	1	1
1	0	1
1	1	1

logický součet
 $y = x_1 + x_2$

x	y	y
0	0	1
0	1	1
1	0	1
1	1	0

negovaný logický součin
 $y = \bar{x}_1 \bar{x}_2$

x	y	y
0	0	1
0	1	0
1	0	0
1	1	0

negovaný logický součet
 $y = \overline{x_1 + x_2}$

Tabulka 2. Funkce dvou proměnných

V Booleově algebře platí následující axiomy :

- *Axiomy pro funkce jedné proměnné*

$$\begin{array}{lll} x \vee 0 = x & x \wedge 0 = 0 & \neg\neg x = x \\ x \vee 1 = 1 & x \wedge 1 = x & \\ x \vee x = x & x \wedge x = x & \\ x \vee \neg x = 1 & x \wedge \neg x = 0 & \end{array}$$

- *Komutativnost součtu a součinu*

$$\begin{array}{l} x \vee y = y \vee x \\ x \wedge y = y \wedge x \end{array}$$

- *Asociativnost součtu a součinu*

$$\begin{array}{l} x \vee y \vee z = (x \vee y) \vee z = x \vee (y \vee z) \\ x \wedge y \wedge z = (x \wedge y) \wedge z = x \wedge (y \wedge z) \end{array}$$

- *Distributivnost součinu vzhledem k součtu*

$$x(y \vee z) = x \wedge y \vee x \wedge z$$

- *Distributivnost součtu vzhledem k součinu*

$$x \vee yz = (x \vee y)(x \vee z)$$

- *de Morganův zákon*

$$\begin{array}{l} \neg(x \vee y \vee z \vee \dots) = \neg x \wedge \neg y \wedge \neg z \dots \\ \neg(x \wedge y \wedge z \wedge \dots) = \neg x \vee \neg y \vee \neg z \dots \end{array}$$

2.1.5. Pravdivostní tabulka

Boolean funkce lze vzhledem ke konečnému počtu funkčních hodnot zobrazit ve formě tabulky následujícím způsobem:

x	$\neg x$
0	1
1	0

NOT

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

OR

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

AND

Tabulka 3. Zápis funkcí pomocí pravdivostních tabulek

Počet záznamů v tabulce je vždy 2^n , kde n je počet vstupů a jednotlivé řádky představují všechny možné nastavení vstupů funkce. V tabulce jsou zleva zapsány hodnoty proměnných a za nimi následují odpovídající funkční hodnoty zobrazované logické funkce. V jedné tabulce může být definováno i více funkcí stejného počtu proměnných, protože každý další sloupec tabulky může popisovat další funkční hodnoty.[1]

2.1.6. Vytváření logických funkcí

Při řešení praktických úloh se často setkáváme s úkolem opačným, než kterému jsme se věnovali doposud, tj. sestavením formule logické funkce z pravdivostní tabulky. Pokud máme logickou funkci definovanou pomocí pravdivostní tabulky, můžeme použít *normálních forem logických formulí* přímo pro vyjádření funkce pomocí formule. Zde existují dvě možnosti:

- *disjunktivní normální forma (DNF)* - Všechny pravdivě ohodnocené řádky pravdivostní tabulky vyjádříme pomocí konjunkce všech proměnných, přičemž proměnnou, která v daném řádku nabývá hodnoty nepravda, znegujeme. Výsledná formule celé funkce je pak disjunkcí takto popsaných řádků.
- *konjunktivní normální forma (KNF)* - Všechny nepravdivě ohodnocené řádky pravdivostní tabulky vyjádříme pomocí disjunkce všech proměnných, přičemž proměnnou, která v daném řádku nabývá hodnoty pravda, znegujeme. Výsledná formule celé funkce je pak konjunkcí takto popsaných řádků.

Výše uvedenými metodami jsme schopni popsat všechny logické funkce, avšak takto vytvořená formule zpravidla není v minimálním tvaru a je možné ji dále zjednodušovat. Zjednodušování funkcí lze provádět několika způsoby. U jednodušších funkcí lze použít úprav Booleovy algebry, u složitějších funkcí je třeba využít některého zjednodušovacího algoritmu. Nejpoužívanějším algoritmem je zřejmě minilimizace funkcí pomocí *Karnaughovy mapy*. [1]

2.2. Kombinační logické obvody

2.2.1. Základní logické členy

Logické členy, nazývané také hradla, realizují základní logické operace. Na vstupy hradla přivedeme hodnoty logických proměnných a na výstupu hradla se objeví výsledná hodnota představované logické funkce.

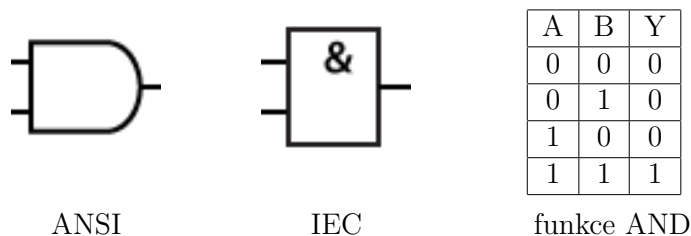
Mezi základní logické členy patří hradla realizující logické funkce logického součinu (AND), logického součtu (OR) a negace (NOT). Pomocí těchto základních funkcí lze realizovat jakýkoliv další a libovolně složitý logický obvod.[1]

Pouhým přidáním negace na výstup hradel logického součtu a součinu dostáváme další dva často používané logické členy – negovaný logický součet (NOR) a negovaný logický součin (NAND). Dále je možné jmenovat hradla exkluzivního součtu (XOR) a negovaného exkluzivního součtu (XNOR).

Tyto logické členy mají zpravidla dva vstupy, s výjimkou negace, která má vstup vždy pouze jeden. V následujícím textu si postupně popíšeme chování a značení každého z uvedených členů.

Logický člen AND

Výstup tohoto členu je definovaný jako logický součin (konjunkce) vstupů $Y = A \wedge B$, kde Y je výstup hradla, A , B vstupy. V tabulce 4. jsou uvedeny schematické značky podle norem ANSI a IEC a také pravdivostní tabulka funkce AND.



Tabulka 4. Schematické značky a pravdivostní tabulka funkce AND

Logický člen OR

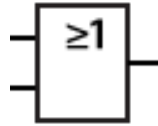
Výstup tohoto členu je definovaný jako logický součet (disjunkce) vstupů $Y = A \vee B$, kde Y je výstup hradla, A , B vstupy. V tabulce 5. jsou uvedeny schematické značky podle norem ANSI a IEC a také pravdivostní tabulka funkce OR.

Logický člen NOT

Výstup tohoto členu je definovaný jako negace (iverze) vstupu $Y = \neg A$, kde Y je výstup hradla, A vstup. V tabulce 6. jsou uvedeny schematické značky podle norem ANSI a IEC a také pravdivostní tabulka funkce NOT.



ANSI

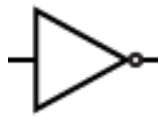


IEC

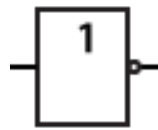
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

funkce OR

Tabulka 5. Schematické značky a pravdivostní tabulka funkce OR



ANSI



IEC

A	Y
0	1
1	0

funkce NOT

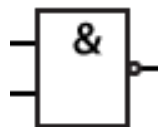
Tabulka 6. Schematické značky a pravdivostní tabulka funkce NOT

Logický člen NAND

Výstup tohoto členu je definovaný jako negace logického součinu (negace konjunkce) vstupů $Y = \overline{A \wedge B}$, kde Y je výstup hradla, A , B vstupy. V tabulce 7. jsou uvedeny schematické značky podle norem ANSI a IEC a také pravdivostní tabulka funkce NAND.



ANSI



IEC

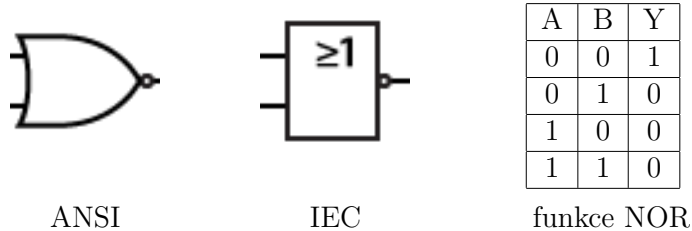
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

funkce NAND

Tabulka 7. Schematické značky a pravdivostní tabulka funkce AND

Logický člen NOR

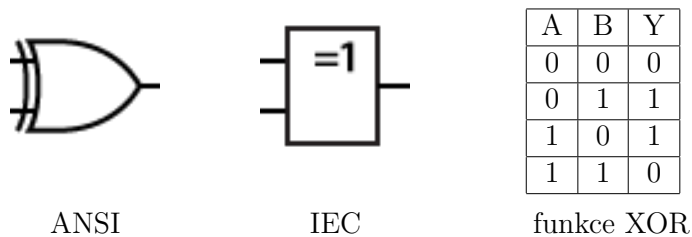
Výstup tohoto členu je definovaný jako negace logického součtu (negace disjunkce) vstupů $Y = \overline{A \vee B}$, kde Y je výstup hradla, A , B vstupy. V tabulce 8. jsou uvedeny schematické značky podle norem ANSI a IEC a také pravdivostní tabulka funkce NOR.



Tabulka 8. Schematické značky a pravdivostní tabulka funkce NOR

Logický člen XOR

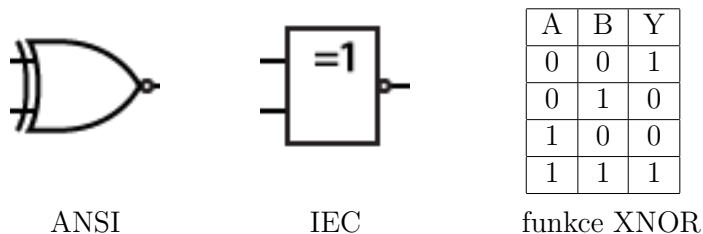
Výstup tohoto členu je definovaný jako výlučný logický součet (exklusivní disjunkce) vstupů $Y = A \oplus B$, kde Y je výstup hradla, A , B vstupy. V tabulce 9. jsou uvedeny schematické značky podle norem ANSI a IEC a také pravdivostní tabulka funkce XOR.



Tabulka 9. Schematické značky a pravdivostní tabulka funkce XOR

Logický člen XNOR

Výstup tohoto členu je definovaný jako negace výlučného logického součtu (negace exklusivní disjunkce) vstupů $Y = \overline{A \oplus B}$, kde Y je výstup hradla, A , B vstupy. V tabulce 10. jsou uvedeny schematické značky podle norem ANSI a IEC a také pravdivostní tabulka funkce NOR.



Tabulka 10. Schematické značky a pravdivostní tabulka funkce XNOR

2.3. Sekvenční logické obvody

Sekvenční logické obvody už jsme definovali jako obvody, které mají jejich výstup závislý nejen na aktuální kombinaci vstupních proměnných na vstupech obvodu, ale také na předchozích stavech vstupů. V podstatě se jedná o kombinační obvody, které mají zavedenu zpětnou vazbu, působící na další chování obvodu. Díky této zpětné vazbě se dokáží chovat sekvenční obvody jako obvody paměťové. Každou hodnotu, kterou si je schopen sekvenční obvod pamatovat, reprezentuje určitým stavem. Pokud pomocí vstupních proměnných chceme, aby si obvod zapamatoval jinou hodnotu, obvod se „překlopí“ do příslušného stavu. Proto se sekvenčním obvodům často říká *klopné obvody*. [1]

Klopný obvod, nebo také paměťový člen, je nejjednodušší sekvenční obvod, který je schopen si uchovat elementární informaci, hodnotu 0 nebo 1, i ve chvíli, kdy tato informace ze vstupu zmizí.

Klopné obvody dělíme podle schopnosti setrvat v nastaveném stavu na:

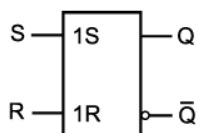
- *astabilní klopný obvod* – obvod má dva nestabilní stavy, neustále se překlápí. Využitím takového obvodu může být například generátor impulzů;
- *monostabilní klopné obvod* obvod má jeden stabilní a jeden nestabilní stav. Po nastavení nestabilního stavu obvod po určitém čase sám přepne zpět do stabilního stavu. Využívá se například jako časovač;
- *bistabilní klopný obvod* obvod má dva stabilní stavy, které je možné podle potřeby měnit a obvod v nich je schopen setrvat. Využívá se jako paměť, registr nebo čítač.

Dále dělíme klopné obvody podle převodu vstupního signálu na výstup na obvody:

- *asynchronní* – obvod se překlápí ihned po přivedení příslušné úrovně signálu na řídicí vstup (například klopné obvody RS, JK);
- *synchronní* – jeden ze vstupů obvodu je synchronizační, kde bývá zpravidla přiveden hodinový signál a obvod se po přivedení signálu na řídicí vstup překlápí až s příslušnou úrovní hodinového signálu (například klopné obvody D).

2.3.1. Klopný obvod RS

Klopný obvod RS má dva vstupy R (Reset), S (Set), výstup Q . a dva stabilní stavy – $Q = 0, Q = 1$. Obvod při hodnotách řídicích vstupů $R = S = 0$ setrvává v předchozím stavu. Při přivedení signálu na vstup R ($R = 1$) se obvod překlápí do stavu $Q=0$. Analogicky při nastavení vstupu $S = 1$ se obvod dostává do stavu $Q = 1$. Stav, kdy na vstupech R i S nastavíme hodnotu 1 by znamenal, že se snažíme překlápět obvod současně do dvou různých stavů a proto je tato kombinace vstupních hodnot zakázána.[1]



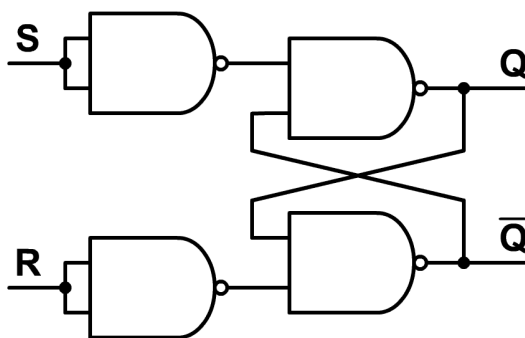
schematická značka

R	S	Q^{n+1}
0	0	Q^n
0	1	1
1	0	0
1	1	X

pravdivostní tabulka

Tabulka 11. Schematická značka klopného obvodu RS a pravdivostní tabulka

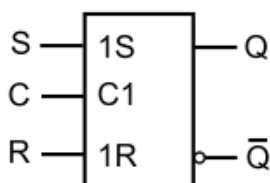
Realizace klopného obvodu mohou být různé, můžeme si však uvést příklad zapojení pomocí hradel NAND (viz obr 1.), kterých se v praxi často využívá. Invertování vstupů je realizováno taktéž hradly NAND spojením jejich vstupů.



Obrázek 1. Příklad zapojení klopného obvodu RS pomocí hradel NAND

2.3.2. Synchronní klopný obvod RS (RST)

Tento klopný obvod je synchronní variantou klopného obvodu RS s tím rozdílem, že reakce výstupu na vstup proběhne vždy až s příchodem impulsu hodinového signálu přiváděného na vstup C. Zakázaný stav pro nastavení vstupů $R = S = C = 1$ platí i v tomto případě, avšak pouze s impulzem hodinového signálu. Bez něj obvod na vstupy nezareaguje a zůstává v předchozím stavu ($Q^{n+1} = Q^n$). Chování obvodu je popsáno v pravdivostní tabulce 12.



schematická značka

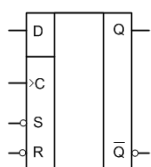
R	S	C	Q^{n+1}
0	0	0	Q^n
0	0	1	Q^n
0	1	0	Q^n
0	1	1	1
1	0	0	Q^n
1	0	1	0
1	1	0	Q^n
1	1	1	X

pravdivostní tabulka

Tabulka 12. Schematická značka klopného obvodu RST a pravdivostní tabulka

2.3.3. Paměťový člen D

Tento paměťový člen je odvozen z klopného obvodu RS. S každým hodinovým impulzem, přiváděným na vstup C, přenáší obvod hodnotu ze vstupu D na výstup Q. Asynchronní vstupy R, S slouží k okamžitému nastavení obvodu na požadovanou hodnotu. Funkcionalitu obvodu popisuje pravdivostní tabulka 13.[1]



schematická značka

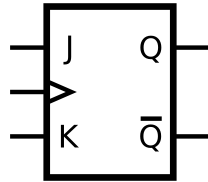
D	C	Q^{n+1}
0	0	Q^n
0	1	0
1	0	Q^n
1	1	1

pravdivostní tabulka

Tabulka 13. Schematická značka klopného obvodu D a pravdivostní tabulka

2.3.4. Klopný obvod JK

Klopný obvod JK² je synchronní klopný obvod vzniklý dalším rozšířením klopného obvodu RST. Jeho vstupy J , K jsou obdobou vstupů $R(K)$, $S(J)$. Odlišností od obvodu RST je ošetření zakázaného stavu při nastavení vstupů $J = K = C = 1$. V tomto případě obvod neguje svůj předchozí stav ($Q^{n+1} = \overline{Q^n}$). Klopné obvody JK se v praxi často využívají, protože jsou univerzální a lze pomocí nich nahrazovat klopné obvody dalších typů.[1]



schematická značka

J	K	C	Q^{n+1}
0	0	0	Q^n
0	0	1	Q^n
0	1	0	Q^n
0	1	1	0
1	0	0	Q^n
1	0	1	1
1	1	0	Q^n
1	1	1	$\overline{Q^n}$

pravdivostní tabulka

Tabulka 14. Schematická značka klopného obvodu JK a pravdivostní tabulka

²Pojmenován podle Jacka Kilbyho, pracujícího v té době u Texas Instruments.

3. Implementace aplikace

3.1. Programovací jazyk a použité nástroje

Na samotném začátku vývoje bylo potřebné zvolit vývojové prostředí a nástroje, které budou použity k vytvoření aplikace LogicBoard. Aplikace by měla sloužit jako učební pomůcka na učebnách vybavených počítači s operačním systémem MS Windows. Pro vývoj nativních Windows aplikací se standardním uživatelským rozhraním je dnes vhodné použít některý z jazyků platformy Microsoft .NET, ze kterých jsem si, čistě z důvodu osobní preference, zvolil jazyk C#. Dále jsme se museli rozhodnout, zda vytvořit aplikaci staršího typu Windows Forms, nebo aplikaci modernějšího typu WPF (Windows Presentation Foundation). Jednou pro nás zajímavou vlastností WPF aplikací je nativně akcelerované 2D grafické prostředí. Navíc WPF využívá vektorově popsaných grafických objektů a to je pro nás velice výhodné při kreslení značek logických obvodů na kreslicí plátno. Bez větších problémů tak můžeme implementovat i změnu měřítko plátna bez zhoršení kvality grafických objektů, což by při použití bitmapových obrázků bylo zbytečně komplikované. Aplikace tedy bude využívat frameworku .NET, konkrétně technologie WPF a kombinaci jazyků C# a XAML.

Z výše uvedených důvodů a na základě vlastní zkušenosti jsem se rozhodl pro samotnou implementaci aplikace použít vývojové prostředí MS Visual Studio 2012. Dále bylo zapotřebí vytvořit schematické značky jednotlivých logických prvků. Jedná se o kreslení vektorových obrázků, k čemuž jsem použil aplikaci Adobe Illustrator a následně jsem pomocí aplikace MS Expression Studio převedl vektorovou grafiku do kódu XAML, který už bylo možné vložit přímo do zdrojového kódu. K úpravám bitmapové grafiky (ikony, obrázky nápovědy aj.) byla použita aplikace Adobe Photoshop.

3.2. Objektový návrh

Návrh programových tříd patří k nejdůležitějším fázím vývoje, protože odráží hlavní koncepci programu. Při návrhu tříd pro aplikaci LogicBoard nebylo vzhledem k menšímu rozsahu projektu nutné vypracovávat kompletní a finální návrh, avšak rozvržení hlavních tříd bylo potřebné si dobře promyslet. Spousta dalších, méně důležitých a rozsahem menších tříd, vzniklo dodatečně.

Nejdůležitější a zároveň nejrozsáhlejší třídou je třída *Workplace* (viz obr 2.), která reprezentuje pracovní prostor sloužící k samotnému návrhu logických obvodů. Objekty, které se umísťují na plátno, jsou instancemi třídy *GPart*, která je odpovědná za grafickou reprezentaci libovolného prvku logického obvodu. Jedním

z atributů této třídy je odkaz na instanci třídy *LPart*. Třída *LPart* je však označena jako abstraktní a popisuje součást logického obvodu pouze obecně. Z této třídy jsou odvozeny třídy *Input*, *Output* a *Gate*, ve kterých jsou řešeny specifika jednotlivých komponent. Třídy *Input* a *Output* představují komponenty sloužící jako zdroj signálu a jako zobrazení úrovně signálu na výstupu. Logická hradla jsou popsána třídou *Gate*, která podle nastaveného typu hradla vyhodnotí hodnoty na vstupech a aktualizuje pomocí odpovídající logické funkce hodnotu na výstupu. Jednotlivé kontakty jsou reprezentovány třídou *Pin*, ze které jsou analogicky odvozeny třídy *InputPin* a *OutputPin*. Každý objekt třídy *LPart* pak obsahuje odkazy na svoje vstupy a výstupy. Poslední důležitou komponentou kreslených logických obvodů jsou vodiče, které jsou implementovány pomocí třídy *Wire*. Zbytek tříd slouží k zabezpečení běžnějších funkcí, např. manipulace s historií akcí (třída *History*) nebo funkcí otevření/uložení souboru. Některé z uvedených tříd si dále probereme podrobněji.

3.2.1. Třída *LPart*

Logickou reprezentaci prvků sestavovaných obvodů zajišťuje třída *LPart*. Jedná se o abstraktní třídu, která není potomkem žádné další třídy. Jsou zde definovány vnitřní proměnné *inputPins* a *outputPin*, které se odkazují na příslušné kontakty představované součástky a které jsou zveřejněny ostatním objektům odpovídajícími vlastnostmi typu read-only. Vstupních kontaktů může být různý počet, proto je proměnná *inputPins* definovaná jako pole objektů třídy *InputPins*, o jehož inicializaci se stará až příslušný potomek třídy *LPart*.

Třída také zabezpečuje uchování nastavené logické hodnoty na výstupu součástky v případě generování pravdivostní tabulky. V tomto případě totiž dochází ke generování všech možných kombinací vstupních hodnot obvodu a jednotlivé součástky obvodu se tak dostávají do různých stavů. Proto se pomocí metody *SaveValue()* hodnota napřed uloží a po vygenerování tabulky se pomocí metody *LoadValue()* se zase obnoví na původní hodnotu. Po vygenerování tabulky tak uživatel má obvod nastaven přesně tak, jak měl předtím.

Další metodou je abstraktní metoda *ResetConnection()*, kterou je každý potomek povinen implementovat tak, aby bylo zabezpečeno logické odpojení prvku od jakékoli jiné části obvodu a zároveň nastavení logických hodnot na hodnoty výchozí.

Velice důležitou metodou je virtuální metoda *UpdateAndForward()*, která je volána při vyhodnocování obvodu postupně na všech logických prvcích zapojené do cesty signálu. Tato metoda obnoví hodnotu na výstupu součástky a zavolá samu sebe na všech součástkách připojených k výstupu.

3.2.2. Třída Gate

Logická hradla jsou charakteristickou skupinou součástek v sestavovaných logických obvodech, proto jsou implementovány pomocí samostatné třídy, která je potomkem třídy *LPart*. Není přitom potřebné vytvářet jednotlivé třídy pro konkrétní hradla, protože se zpravidla liší pouze v logické funkci, kterou hradlo realizuje. Z toho důvodu je v této třídě definován jediný konstruktory, který je přetížený a jako jediný parametr vyžaduje uvedení typu operace (parametr je výčtového typu *LogicOperation*). Při inicializaci objektu pomocí takového konstruktory se nastaví hodnota privátní proměnné *operation* a při volání metody *Update()* se vyhodnotí příslušná logická funkce. Pokud se má vytvářet hradlo pro některou z inverzních funkcí, např. *nand*, nastaví se privátní proměnná *invert* na hodnotu *true* a proměnná *operation* na odpovídající funkci. Inverze se tak provede až dodatečně a není nutné definovat přechodových funkcí dvojnásobek.

3.2.3. Třída GPart

Prvky obvodu logiky představované třídou *LPart* je nutné zobrazovat na plátně. Každý takový objekt je proto navázán na svoji grafickou reprezentaci, tj. na objekt třídy *GPart*. Třída je potomkem třídy *UserControl* a uchovává informace o poloze objektu na plátně, jeho velikosti, jeho vybrání uživatelem a poloze jeho kontaktů.

Inicializace objektu se provádí pomocí konstruktory, kterému se jako parametr předá ta instance třídy *LPart*, která má být reprezentována. Během inicializace se dále vytvoří kolekce relativních souřadnic kontaktů, aby bylo možné později detekovat připojené vodiče. K tomuto účelu je definována metoda *CheckForConnection()*, která na dotaz na souřadnici odpoví vrácením odkazu na kontakt součástky – pokud se na daných souřadnicích některý nachází.

Při zobrazení obrázku schematické značky využíváme grafiky uložené ve formátu XAML. Metoda *UpdateResource()* načte příslušný kód XAML a tím dojde k zobrazení součástky. Navíc je u této třídy přepsána metoda *OnRender()*, které při aktivním označení vykreslí přerušovaný červený čtverec kolem vybraného prvku a při aktivních popiscích součástek vypíše u prvku text s označením (toho se využívá při zobrazení pravdivostní tabulky, aby bylo zřejmé, o které vstupy či výstupy se v tabulce jedná).

3.2.4. Třída Wire

Uživatel během návrhu obvodu spojuje jednotlivé komponenty vodiči, které jsou řešeny třídou *Wire*. Třída je odvozena od třídy *UIElement*, aby mohl být vodič bez problému přidán na plátno a aby mohl být vykreslen pomocí přepsané metody *OnRender()*. Ve třídě je definována privátní proměnná *anchors*, což je kolekce bodů, které představují jednotlivé pevné body, přes které vodič prochází. Třída řeší i náhled té části vodiče, kterou uživatel právě kreslí a to tak, že od posledního pevného bodu se vykresluje čára k aktuální pozici kurzoru. Každá instance této třídy může také odpovědět pomocí metody *IsJointAt(Point)*, zda bod předaný jako parametr leží na cestě vodiče. Využitím této metody další metoda nazvaná *SelectConnectedWires()* vrací kolekci všech vodičů, které jsou s tímto vodičem spojeny. Další metody a vlastnosti této metody pouze zabezpečují rozhraní pro práci s objektem vodiče.

3.2.5. Třída WorkPlace

Všechny prvky obvodu jsou umísťovány na pracovní plochu, která je popsána třídou *WorkPlace*. Ta je potomkem třídy *System.Windows.Controls.Canvas*, tedy kreslicího plátna. Díky tomu máme možnost přidávat na tuto plochu další grafické objekty a jednoduše s nimi manipulovat. Pro jednodušší umísťování komponent obvodu a kreslení vodičů je plátno rozděleno na políčka, na které se kurzor při pohybu nad plátnem zachytává. Zároveň jsou zde zpracovávány veškeré události dějící se na plátně, což představuje většinu akcí celé aplikace. Mimo to obsahuje třída metody pro přidávání a odebrání komponent obvodu, výběr editačního nástroje a také metody obsahující algoritmy pro hledání spojů a vyhodnocení obvodu.

Proměnné a konstanty

Jak jsme již zmínili dříve, kreslicí plátno je nejdůležitější prvek celé aplikace. A jeho parametry, stav a obsah jsou definovány právě v této třídě:

- *DOCUMENT_HEIGHT* – výška plátna (konstanta)
- *DOCUMENT_WIDTH* – šířka plátna (konstanta)
- *MARGIN* – okraj plátna (konstanta)
- *SEGMENT_SIZE* – velikost jednoho políčka (konstanta)

- *isSaved* – stav uložení obsahu plátna
- *isGridShowed* – stav zobrazení zachytávací mřížky
- *IEC* – norma IEC / ANSI
- *isLabelsShowed* – stav zobrazení popisků vstupů a výstupů
- *selectedTool* – aktuálně vybraný editační nástroj
- *mouseLocation* – poloha kurzoru myši, přichycená na mřížku
- *parts* – kolekce všech prvků na plátně (objektů třídy *GPart*)
- *wires* – kolekce vodičů nakreslených na plátně

Metody

V této třídě je definováno více metod, avšak jedná se převážně o metody pro manipulaci s objekty, obsluhu událostí, zasílání zpráv všem objektům na plátně a podobně. Těmi se nebudeme v tomto textu zabývat, uvedeme si pouze několik důležitých metod:

Metoda *ReloadConnections()*, která má za úkol analyzovat obvod a sestavit kolekci spojení mezi jednotlivými komponentami obvodu. Pro uložení těchto spojení se využívá objektů třídy *Connection*, které obsahují vždy právě jeden výstup některého prvku obvodu a vstupy těch prvků, se kterými je daný výstup vodivě spojen. K vytvoření těchto spojení je však potřebné napřed analyzovat obvod a vytvořit segmenty vodičů, které jsou spolu navzájem propojené. Pro tento účel je definována metoda *CompileWireSegments()*, které tyto segmenty vrací jako kolekci kolekcí vodičů.

Metodou *UpdateValues()* nyní můžeme obnovit veškeré hodnoty hradel v obvodu a vyhodnotit výstupy samotného obvodu. Metoda pracuje v několika cyklech, přičemž postupně volá metodu *UpdateAndForward()* na každý z obvodových vstupů, dokud se hodnoty na výstupech obvodu neustálí. Je zde ošetřen i stav, kdy by mohlo dojít k nekonečnému vyhodnocování obvodu, takže program by v takovém případě vyhodnocování zanechal a oznámil uživateli dialogovým oknem zprávu, že obvod se nechází v nestabilním stavu.

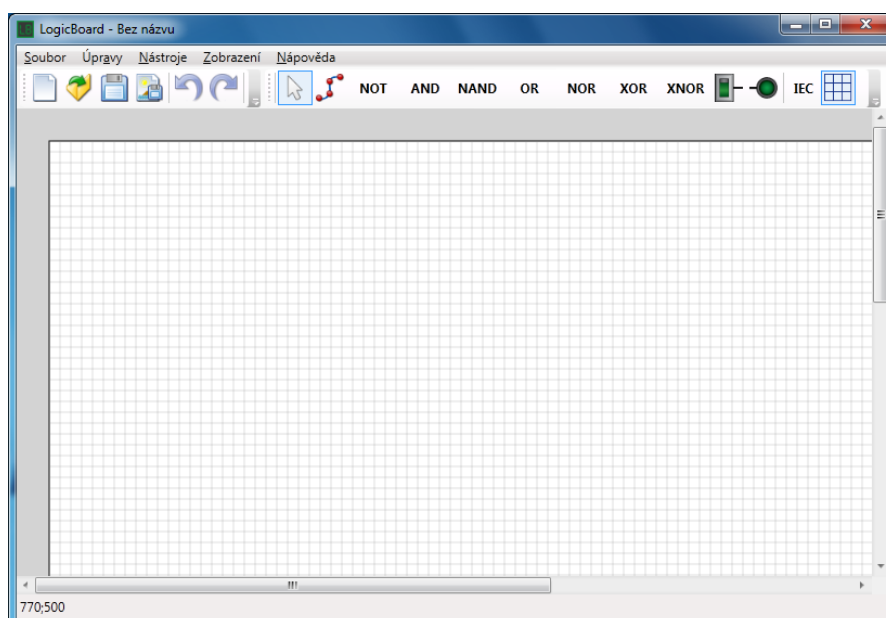
3.3. Vytvoření grafických objektů a jejich použití

Při prvotním návrhu aplikace se jako nejjednodušší metoda, jak řešit zobrazování schematických značek, nabízelo použití bitmapových obrázků. V konceptu jsem toto řešení dokonce vyzkoušel, ale nakonec jsem se rozhodl pro vektorovou grafiku. Pokud totiž použijeme grafické prvky kreslené pomocí křivek, můžeme jednoduše přizpůsobit velikost prvků. Díky tomu můžeme také poměrně jednoduše implementovat zvětšování či zmenšování celého plátna. Tím značně obohatíme možnosti uživatelského rozhraní. Neméně důležitým argumentem pro výběr vektorové grafiky je výrazně lepší kvalita exportovaných schémat do obrázků. Při použití bitmapové grafiky byl totiž exportovaný obrázek značně neostrý.

Pro vytvoření vektorové grafiky můžeme použít libovolný vektorový editor. V tomto případě byla použita aplikace Adobe Illustrator, avšak pro potřeby kreslení takto jednoduché grafiky bychom mohli použít méně sofistikovaný software. V této fazi máme grafiku nakreslenou, ale uloženou ve formátu aplikace Adobe Illustrator (.ai) a to není pro použití v naší aplikaci vhodné. Nicméně aplikace Microsoft Expression Design dokáže otevřít soubor .ai a konvertovat grafiku v něm uloženou do formátu *XAML Resource Dictionary*. To je už formát, se kterým dokáže pracovat Visual Studio a dokonce jsme schopni z takového zdroje grafických objektů načíst grafiku dynamicky, přímo při vykonávání kódu. Této metody je použito právě v metodě *UpdateResource()* třídy *GPart*.

4. Popis aplikace

V této kapitole se podrobně seznámíme s možnostmi a funkcemi vytvořené aplikace Logic Board a s jejím uživatelským rozhraním. Po spuštění aplikace se otevře hlavní okno s prázdným, nepojmenovaným dokumentem (viz obr 2.), který je představován kreslicím plátnem zaplňujícím většinu plochy okna. Nad kreslicím plátnem se nachází panel nástrojů, který obsahuje tlačítka pro manipulaci s dokumentem a všechny nástroje potřebné pro návrh obvodu. V záhlaví okna je pak umístěna hlavní nabídka, kde je možné najít všechny funkce programu. Ve spodní části je umístěn stavový řádek, na kterém jsou vidět souřadnice kurzoru, který je přichycený na mřížku plátna.














Obrázek 2. Hlavní okno programu

4.1. Instalace

Instalace aplikace se provádí běžným způsobem. Po spuštění instalátoru postupně projdeme několik dialogových oken, kde si můžeme změnit umístění souborů programu. Po ukončení instalátoru máme ikonu pro spuštění aplikace umístěnou na ploše a v nabídce Start.

4.2. Panel nástrojů

Protože aplikace LogicBoard charakterem připomíná například grafický editor, bude panel nástrojů pravděpodobně nejdůležitějším ovládacím prvkem při návrhu obvodu pro většinu uživatelů. V tabulce 15. je uveden popis jednotlivých tlačítek.

	vytvoří nový dokument
	otevře existující dokument
	uloží dokument
	exportuje dokument do obrázku
	provede krok zpět
	provede krok vpřed
	nástroj výběru
	nástroj pro kreslení vodičů
NOT .. XNOR	nástroje pro kreslení logických hradel
	vstup
	výstup
	zobrazení zachytávací mřížky

Tabulka 15. Přehled tlačítek na panelu nástrojů

4.3. Hlavní nabídka

Nabídka programu poskytuje přístup ke všem funkcím aplikace a je navržena v souladu s HIG³ pro systém Microsoft Windows. V tabulce 16. je znázorněno schéma nabídky.

Soubor >> Nový	Vytvoří nový dokument
Soubor >> Otevřít...	Otevře existující dokument
Soubor >> Uložit	Uloží dokument
Soubor >> Uložit jako...	Uloží dokument pod jiným jménem
Soubor >> Export...	Exportuje do obrázku
Soubor >> Ukončit	Ukončí aplikaci
Úpravy >> Zpět	Krok v historii zpět
Úpravy >> Opakovat	Krok v historii vpřed
Úpravy >> Odstranit	Odstraní označené objekty na plátně
Nástroje >> Nástroj výběru	Vybere nástroj výběru
Nástroje >> Kreslení vodičů	Vybere nástroj pro kreslení vodičů
Nástroje >> Vstup	Vybere nástroj pro kreslení vstupů
Nástroje >> Výstup	Vybere nástroj pro kreslení výstupů
Nástroje >> Hradlo NOT .. XNOR	Vybere nástroj pro kreslení hradla
Nástroje >> Generovat pravdivostní tabulku	Otevře okno s pravdivostní tabulkou
Zobrazení >> Norma IEC	Přepíná normu zobrazení (ANSI/IEC)
Zobrazení >> Zobrazit mřížku	Zobrazení zachytávací mřížky plátna
Zobrazení >> Zvětšit	Zvětší plátno
Zobrazení >> Zmenšit	Zmenší plátno
Nápověda >> Nápověda	Otevře nápovědu aplikace
Nápověda >> O aplikaci LogicBoard	Zobrazí informace o aplikaci

Tabulka 16. Struktura programové nabídky

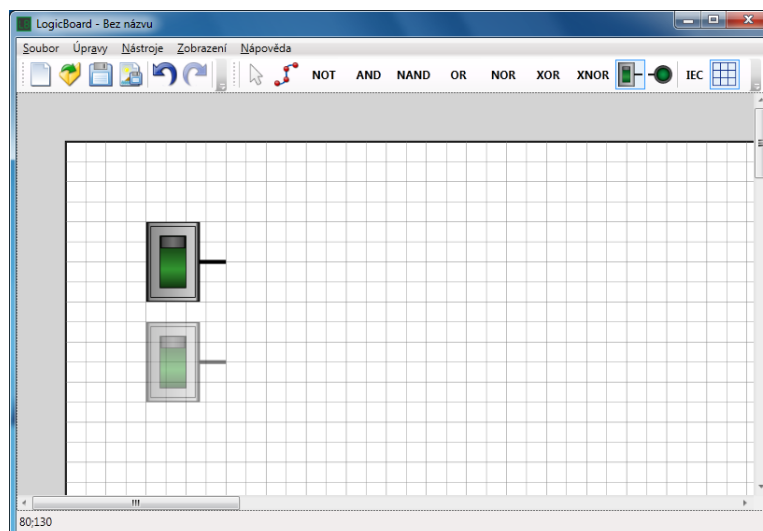
³Human Interface Guideline

4.4. Nástroje k návrhu obvodu

Během kreslení navrhovaného logického obvodu můžeme volit z několika nástrojů. Všechny nástroje lze nalézt na panelu nástrojů a také v programové nabídce **Nástroje**:

Nástroj výběru je základní, univerzální editační nástroj. V tomto editačním režimu kliknutím na komponentu obvodu a tažením myši můžeme komponentu libovolně posouvat po plátně. Pouhým kliknutím levým tlačítkem myši komponentu nebo vodičoznačíme. Označení více objektů na plátně provádíme za stisknuté klávesy **Ctrl**. Opětovným kliknutím na již označený objekt se označení zruší. Pokud klikneme do prázdného místa na plátně, dojde ke zrušení označení všech objektů. Pokud máme označený jeden nebo více objektů, klávesou **Delete**, nebo příkazem nabídky **Úpravy > Odstranit** je odstraníme z plátna. Pokud je aktivní některý z dalších nástrojů, kliknutím pravým tlačítkem v oblasti plátna se vybere nástroj výběru.

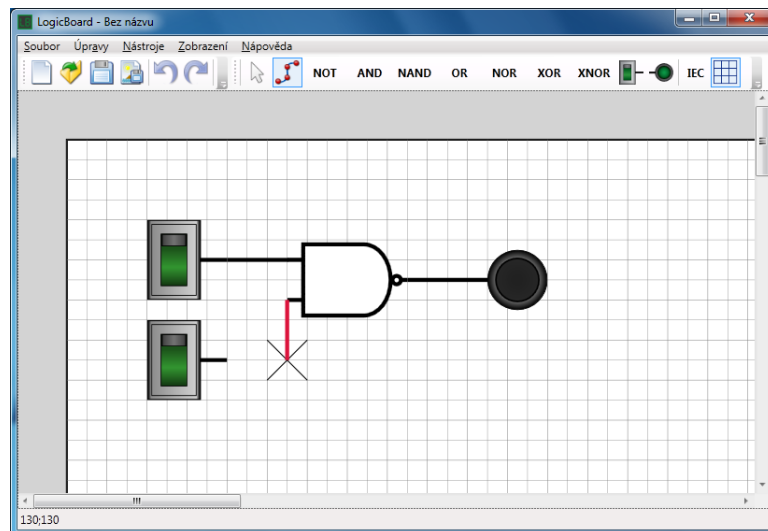
Nástroje *Vstup* a *Výstup* slouží k přidání říditelných vstupů a signalizací výstupů na plátno. Po výběru nástroje se při pohybu kurzoru myši nad plátnem zobrazuje náhled prvku a po kliknutí na plochu plátna se tento prvek na plátno vloží (viz obr.3.). Můžeme tak vložit několik prvků po sobě, nástroj zůstává aktivní až do výběru nástroje jiného.



Obrázek 3. Přidání nového prvku na plátno

Hradlo *NOT*, *AND*, *NAND*, *OR*, *NOR*, *XOR* a *XNOR* jsou nástroje pro kreslení logických členů. Fungují stejně jako nástroje pro kreslení vstupů a výstupů. Po aktivování konkrétního nástroje můžeme rozmísťovat na plátno hradla vybraného typu, dokud nástroj nezrušíme kliknutím pravým tlačítkem myši či zvolením jiného nástroje.

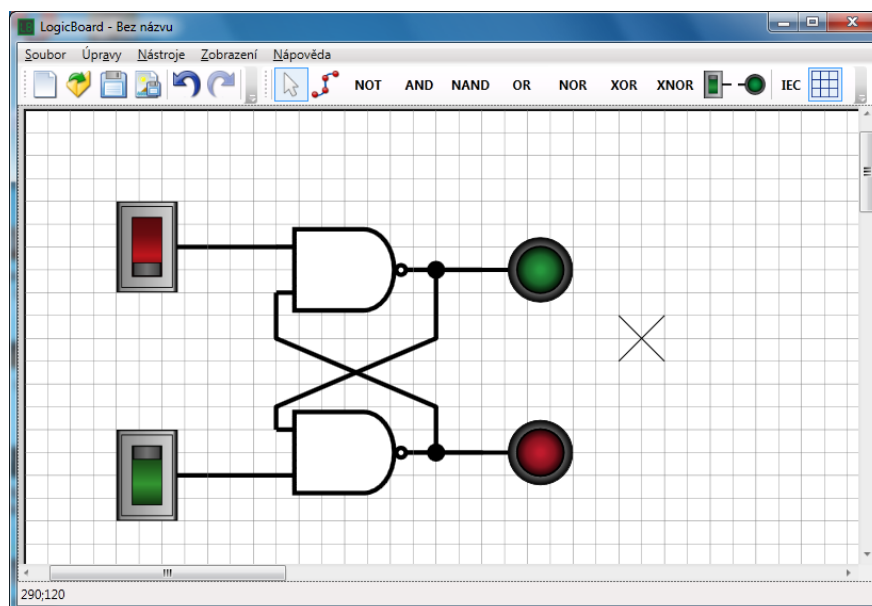
Nástroj *Kreslení vodičů* slouží ke spojování jednotlivých komponent obvodu vodiči. Každý prvek umístěný na plátně je opatřen vývody, ke kterým lze vodiče připojit. Po aktivování nástroje *Kreslení vodičů* kliknutím na konec vývodu nebo na prázdné místo na plátně začneme kreslit vodič. V tu chvíli se totiž vloží na plátno první pevný bod vodiče a k aktuální poloze kurzoru nám vede červená čára, sloužící jako náhled toho, kudy vodič povede. Každé další kliknutí vždy vloží další pevný bod vodiče. Vodič ukončíme tak, že poslední jeho pevný bod umístíme na konec vývodu jiné součástky, na jiný vodič, nebo stiskneme pravé tlačítko myši. V tom momentě jsme připraveni kreslit další vodič. Program nutí vést vodiče pouze svislé nebo vodorovné, tomuto chování však můžeme zabránit držením klávesy **Ctrl** během kreslení. Poté můžeme kreslit vodiče pod libovolným úhlem. Protože však takový vodič neprochází přesně body zachytávací mřížky, může být obtížnější takový vodič označit pro případné vymazání, nebo nemusíme být schopni k takovému vodiči připojit vodič jiný. Přednostně proto kreslíme vodiče ortogonálně. Vodiče můžeme libovolně propojovat a větvit. V takových případech vznikne na vodičích uzel.



Obrázek 4. Ukázka kreslení vodiče

4.5. Testování obvodu

Vyhodnocování obvodu na plátně se provádí okamžitě po každé změně obvodu. Jako zdroj signálu slouží prvek vytvořený pomocí nástroje *Vstup*. Jeho hodnota po vložení na plátno je 0, ale kliknutím na prvek pravým tlačítkem myši hodnotu můžeme měnit. Takto si můžeme nastavit hodnoty všech vstupů, které jsme si na plátno vložili. Po průchodu signálu obvodem si výstupní hodnoty můžeme odečíst na prvcích vytvořených nástrojem *Výstup*. Takto můžeme testovat funkci obvodu jako celku a studovat jeho chování.

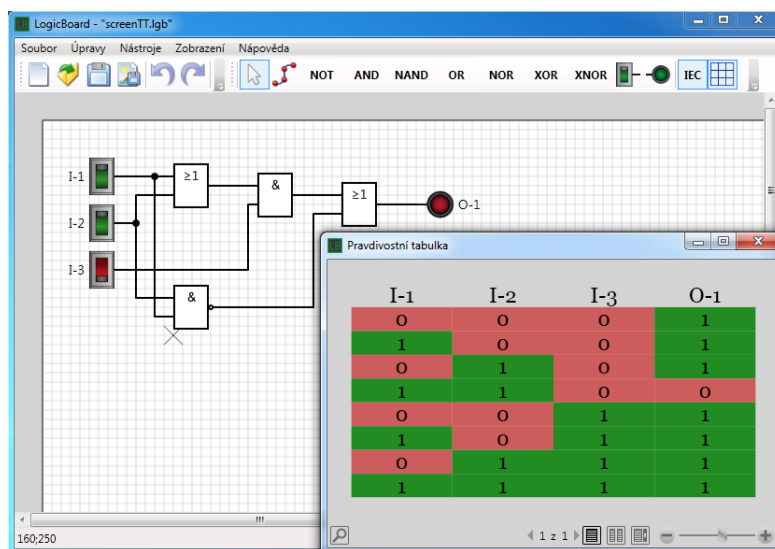


Obrázek 5. Manuální testování obvodu

4.6. Generování pravdivostní tabulky

Další možností, jak otestovat sestavený obvod je automatické testování pomocí vygenerování pravdivostní tabulky. To provedeme pomocí programové nabídky **Nástroje** » **Generovat pravdivostní tabulku**. V chvíli se u jednotlivých vstupů a výstupů na plátně objeví jejich označení, aby je bylo možné jednoznačně identifikovat. Otevře se okno s pravdivostní tabulkou, kde jsou nad sloupci vyznačeny odpovídající vstupy a výstupy. Snadno tak zjistíme, jaké výstupní hodnoty odpovídají konkrétnímu nastavení vstupních hodnot.

Vygenerovaná tabulka nicméně nedokáže zobrazit hodnoty výstupů, které jsou závislé na předcházejících stavech obvodu. Proto tabulka nemusí obsahovat naprosto přesné údaje při vyhodnocování sekvenčních obvodů. Pro kombinační obvody, které pracují s aktuálními hodnotami na vstupech však pracuje bezchybně.



Obrázek 6. Vygenerovaná pravdivostní tabulka

4.7. Práce se soubory

Schémat vytvořená aplikací LogicBoard lze uložit do souboru pomocí standardního dialogu pro ukládání souborů. Vyvoláme ho z programové nabídky **Soubor** \gg **Uložit jako...**. Po zvolení názvu souboru se soubor uloží do formátu *.lgb*. Takto uložené soubory lze opětovně otevřít pomocí příkazu nabídky **Soubor** \gg **Otevřít...**.

4.8. Export obvodu do souboru

Této aplikaci lze bez problému využít i k nakreslení schémat logických obvodů pro další využití, například v dokumentaci. Příkazem nabídky **Soubor** \gg **Export...** můžeme pomocí standardního dialogového okna uložit nakreslené schéma jako obrázek *.png*. Během ukládání si aplikace sama na potřebný čas zruší zobrazení zachytávací mřížky, aby nebyla vidět na výsledném obraze.

4.9. Historie akcí

Během kreslení obvodu na plátně je každá akce uživatele zapamatována. Proto je možné používat příkazy **Úpravy** **Zpět** a **Úpravy** **Vpřed**, nebo odpovídající tlačítka na panelu nástrojů, až k první úpravě obvodu.

4.10. Přizpůsobení zobrazení

Vzhled aplikace je po spuštění nastaven tak, aby byl vhodný pro většinu uživatelů při standartních úkonech. Někdy však může být vhodné tyto nastavení změnit. Jedná se o následující funkce:

Zobrazení zachytávací mřížky Mřížka je ve výchozím nastavení zobrazena, protože usnadňuje orientaci na plátně a snadněji se pomocí ní zarovnávají kreslené komponenty obvodu. Její vypnutí provedeme příkazem nabídky **Zobrazit** **Zobrazit mřížku**. Plátno s vypnutou mřížkou je například použito při exportování schématu do obrázku.

Zvětšení / zmenšení plátna Při kreslení detailnějších částí obvodu může být vhodné si takovou část obvodu zvětšit. Ovládat zvětšení lze pomocí kolečka myši za současně držené klávesy **Ctrl**, nebo pomocí příkazu nabídky **Zobrazit** **Zvětšit**, resp. **Zobrazit** **Zmenšit**. Příklad zvětšeného plátna můžeme vidět například na obrázku 3.

Změna normy schematických značek Aplikace podporuje zobrazení schematických značek logických hradel dle norem IEC a ANSI. Při spuštění programu je zvolena norma ANSI, přepnutí do normy IEC lze provést kliknutím na ikonu s textem *IEC* na panelu nástrojů, nebo příkazem nabídky **Zobrazení** **Norma IEC**. Zobrazení dle normy IEC je vidět na obrázku 6.

4.11. Požadavky na systém

Program je možné spustit na libovolném počítači platformy x86/x64 s operačním systémem Microsoft Windows XP a vyšších a instalovaným frameworkem Microsoft .NET 4.0.

Závěr

Cílem této práce bylo vytvořit aplikaci pro kreslení a simulaci logických obvodů sestavených ze základních logických prvků — hradel. V práci měla být zároveň zpracována teorie vytváření logických obvodů. Všechny stanovené požadavky ze zadání byly splněny a měla by tedy být vhodná i pro použití jako učební pomůcky.

První, teoretická, část práce se věnuje teorii vytváření logických obvodů. Popsali jsme si základní pojmy, vysvětlili logické proměnné a funkce a postupně jsme se věnovali popisu jak kombinačních, tak sekvenčních logických obvodů.

Samotná aplikace vyhovuje zadaným požadavkům. Logické obvody je možné navrhovat pomocí jednoduchého a intuitivního rozhraní a testování kresleného obvodu navíc probíhá průběžně, zcela automaticky. Pro nakreslený obvod je možné nechat si sestavit pravdivostní tabulku logické funkce, kterou obvod realizuje, nebo obvod testovat manuálně, pomocí nastavování vstupních signálů. V aplikaci je implementována historie akcí, takže všechny provedené akce je možné vrátit zpět, popřípadě je opět zopakovat. V neposlední řadě je možné si vzhled aplikace v některých ohledech přizpůsobit. Například je možné měnit zvětšení obvodu nebo zobrazení hradel dle dvou různých norem. Výsledný obvod je možné exportovat jako obrázek a použít ho tak k dalším účelům.

Jako vylepšení aplikace by bylo možné rozšířit nabídku prvků, ze kterých se obvody sestavují. Užitečným prvkem by byl například generátor hodinového signálu nebo některé složitější, ale běžně využívané obvody, zapouzdřené do jednoho prvku. Bylo by také vhodné doplnit podporu pro práci se schránkou, aby bylo možné například kopírovat části obvodu. Dalším nedostatkem je nesprávné generování pravdivostní tabulky pro sekvenční obvody, protože tato funkce nedokáže vzít v úvahu předešlé stavy obvodu. Pouhým kombinováním vstupních hodnot proto může dojít ke zkresleným údajům v tabulce.

Na dnešním bohatém softwarovém trhu existují spousty aplikací, které umožňují návrh logických obvodů. Mnoho jich je na dobré úrovni, většina z nich je ovšem zbytečně komplikovaná a taková aplikace by jako učební pomůcka nemusela být nejvhodnější. Výhodou vytvořené aplikace je tedy především její jednoduchost a případná možnost dalších rozšíření a úprav z důvodu, že k ní budou dostupné i zdrojové kódy.

Conclusions

The main aim of this project was to implement an application allowing drawing and simulation of logic circuits built of elementary logic components – logic gates. This thesis should have also elaborated the theory involved with creating logic circuits. All assigned tasks have been accomplished accordingly therefore the application should be suitable to be used as a teaching aid.

The first section of this thesis is devoted to theory involved with creating logic circuits. We have described basic terms, explained logic variables and we have described both types of logic circuits – boolean and sequential.

The application itself suites given requirements. Logic circuits can be designed using a simple and intuitive user interface. Moreover, testing of the built circuit is done continuously during the drawing. A truth table can be generated for designed circuit, showing the corresponding logic function. Also manual testing of the circuit can be done by changing the inputs values and observing the output changes. Application has a history of actions implemented so any editing steps can be reverted and re-done again. Some of the other features allow user to change an appearance of the canvas. User can use a magnifier to zoom in and out the canvas and also to switch between two different types of scheme standards. The finished circuit can be exported to an image so it can be processed further.

Some improvements of the applicatio could be absolutely done to exted functionality. Set of components could be enriched – e.g. clock impuls generator or some of the more complex but still widely used circuits embedded into one component. A clipboard support could be implemented as well so it could be possible to copy a section of a circuit, for instance. A drawback which could be resolved is also incorrect generated truth table in case of sequential circuits since the used algorithm cannot consider the previous states of the outputs.

Loads of applications can be found on nowadays' software market which allow desinging of logic circuits. Many of them are nicely done however most of them are too complicated to be used as a teaching aid. The main advantage of our application is its simplicity. Also the availability of source codes makes it possible for further changes and improvements.

Reference

- [1] Jean-Michel Bernard, Jean Hugon, Robert le Corvec : Od logických obvodů k mikroprocesorům I. : Základy kombinačních a sekvenčních obvodů, SNTL - Nakladatelství technické literatury, Praha, 1984, ISBN 0201157675.
- [2] John Sharp : Microsoft Visual C# 2010 Krok za krokem, Computer Press, a.s., Brno, 2010, ISBN 978-80-251-3147-3.
- [3] Chris Sells : C# a WinForms - programování formulářů Windows, Zoner Press, Brno, první vydání, Brno, 2005, ISBN 80-86815-25-0

A. Obsah přiloženého CD/DVD

Zde je uveden stručný popis obsahu přiloženého CD/DVD.

bin/

Instalátor programu *LogicBoard.exe*.

doc/

Dokumentace práce ve formátu PDF a všechny soubory nutné pro vygenerování PDF souboru dokumentace (v ZIP archivu).

src/

Kompletní zdrojové texty programu LogicBoard.

readme.txt

Instrukce pro instalaci a spuštění programu LogicBoard, včetně požadavků pro jeho provoz.

Navíc CD/DVD obsahuje:

data/

Uložené ukázkové obvody pro potřeby prezentace. (soubory *.lgb)

install/

Instalátor použitého frameworku.