PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Sekvenční číslicové obvody

2015 David Kornel

Anotace

Bakalářská práce se zaměřuje na popis a funkci základních kombinačních a sekvenčních číslicových obvodů. V práci jsou z kombinačních obvodů popsána základní hradla, ze sekvenčních obvodů pak klopné obvody, posuvné registry a čítače. V rámci práce vznikla aplikace SCEditor, která slouží pro návrh a simulaci zapojení s kombinačními a sekvenčními číslicovými obvody. Pro vývoj aplikace byl využit jazyk C# a pro grafické uživatelské rozhraní technologie WPF.

Děkuji RNDr. Arnoštu Večerkovi za odborné rady při vytváření této diplomové práce. Dále děkuji mé rodině a přítelkyni.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Číslicové obvody jsou elektronické obvody, sloužící pro realizaci logických funkcí. V praxi se se využívají v elektronických zařízeních jako například v RC modelech, kalkulačkách, počítačích atd...

Číslicové obvody se dělí na kombinační a sekvenční. Číslicové obvody pracují pouze se dvěma stavy, a to s logickou nulou a logickou jedničkou. Z kombinačních obvodů jsou v práci zastoupena hradla, ze sekvenčních jsou jsou v práci zastoupeny klopné obvody, posuvné registry a čítače.

Cílem práce bylo implementovat aplikaci pro práci se sekvenčními a kombinačními číslicovými obvody. Aplikace dle požadavků musí umožňovat uživateli vkládání číslicových obvodů na plátno, kreslení spojů a provádění simulace funkcionality sestavených schemat.

Implementace aplikace byla rozdělena do dvou částí. V první části byla implementována logika číslicových obvodů a algoritmus simulace sestavených schemat. V druhé části byla implementována prezentační vrstva aplikace. Vzniklá aplikace *SCEditor* může sloužit jako názorná učební pomůcka pro studenty elektrotechnicky zaměřených středních škol.

Aplikace *SCEditor* umožňuje sestavování obvodů a provádění interaktivní simulace nad sestavenými obvody. K dispozici je velká škála kombinačních i sekvenčních obvodů. Z kombinačních obvodů je to například hradlo AND, OR, NAND, NOR a další. Ze sekvenčních obvodů například klopné obvody RS, JK, D dále posuvné registry a další. Dále aplikace umožňuje ukládání a načítání sestavených schemat a export schematu do formátu PNG.

2. Základní pojmy

Booleovská funkce s n argumenty, je funkce, která pro libovolnou kombinaci hodnot 0 nebo 1 přiřadí hodnotu 0 nebo 1.

Hradlo je logický člen, který vyčísluje booleovskou funkci. Vstupy hradla reprezentují argumenty booleovské funkce. Hradlo je základním stavebním prvkem číslicových (logických) obvodů.

Stav obvodu je stav, kdy obvod má určité logické hodnoty na vstupech a výstupech.

Impulz je změna z výchozí logické hodnoty na opačnou a následně zpět k původní logické hodnotě.

Náběžná hrana je přechod vstupního signálu obvodu z hodnoty logická 0 do hodnoty logická 1.

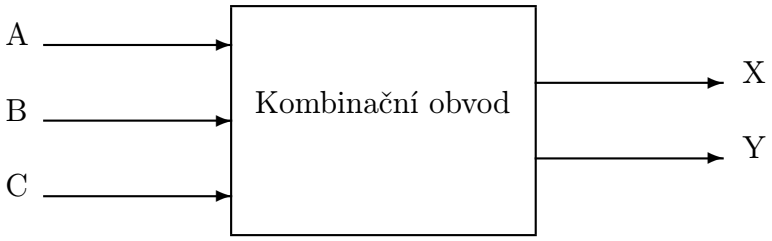
Sestupná hrana je přechod vstupního signálu obvodu z hodnoty logická 1 do hodnoty logická 0.

3. Číslicové obvody

Číslicové (logické) obvody jsou elektronické obvody sestavené s logických členů tzv. *hradel* a slouží pro realizaci *booleovských funkcí*. Číslicové obvody pracují se dvěma diskrétními stavy[1], logickou nulou a jedničkou. V číslicové technice jsou tyto hodnoty také reprezentované napěťovými úrovněmi (označovanými jako úroveň L, která odpovídá logické nule, a úroveň H, která odpovídá logické jedničce). Číslicové obvody dělíme na *kombinační číslicové obvody* 3.1. a *sekvenční číslicové obvody* 3.2.

3.1. Kombinační číslicové obvody

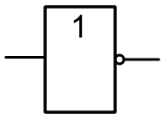
Kombinační číslicové obvody, dále jen *KO*, jsou obvody, u nichž stavy logických hodnot na výstupech X, Y... jsou závislé na okamžité kombinaci logických hodnot na vstupech[2] A, B, C... a nezávisí na předchozím stavu obvodu. Na obrázku 1. je znázorněno schema *KO*. *KO* implementují logické operace jako konjunkce, disjunkce, negace a ekvivalence. Tyto obvody jsou popsány níže.



Obrázek 1.: Kombinační číslicový obvod.

3.1.1. Hradlo NOT

Hradlo NOT implementuje unární logickou operaci negace, tzn. že hradlo na výstup nastaví opačnou logikou hodnotu, než jaká je na vstupu (viz tabulka 1.). Tato funkce je definovaná pomocí předpisu $X = \overline{A}$. Na obrázku 2. je schematická značka, na které je vidět, že hradlo má jeden vstup a jeden výstup. Hradlo NOT se vyrábí v integrované podobě jako obvod 7404.



Obrázek 2.: NOT značka

A	X
0	1
1	0

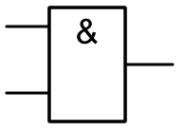
Tabulka 1.: NOT stavová tabulka

[1] Existují ale také obvody, které umí pracovat se spojitým (analogovým) signálem např. *analogově-digitální převodník*.

[2] S vyjimkou krátkého přechodového děje, způsobeného tranzistory.

3.1.2. Hradlo AND

Hradlo AND implementuje binární logickou operaci konjunkce. Disponuje dvěma vstupy a jedním výstupem. Hradlo AND nastaví na výstup hodnotu logická 1 pouze v případě, kdy jsou na obou vstupech hodnoty logická 1 (viz tabulka 2.). Tato funkce je definovaná předpisem $X = A \wedge B$. Na obrázku 3. je zobrazena schematická značka obvodu. Hradlo AND se vyrábí jako integrovaný obvod 7408.



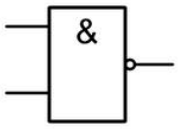
Obrázek 3.: AND značka

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Tabulka 2.: AND stavová tabulka

3.1.3. Hradlo NAND

Hradlo NAND vznikne sloučením hradla AND a hradla NOT tak, že na výstup hradla AND připojíme hradlo NOT. Disponuje dvěma vstupy a jedním výstupem. Hradlo NAND nastaví na výstup hodnotu logická 0 pouze v případě, kdy jsou na obou vstupech hodnoty logická 1 (viz tabulka 3.). Tato funkce je definovaná předpisem $X = (\overline{A \wedge B})$. Na obrázku 4. je zobrazena schematická značka obvodu. Hradlo NAND se vyrábí jako integrovaný obvod 7400.



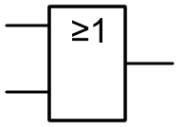
Obrázek 4.: NAND značka

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Tabulka 3.: NAND stavová tabulka

3.1.4. Hradlo OR

Hradlo OR implementuje binární logickou operaci disjuknce. Hradlo NOR nastaví na výstup hodnotu logická 0, pouze v případě, když na obou vstupech hradla je hodnota logická 0, v ostatních případech je na výstupu hodnota logická 1 (viz tabulka 4.). Tato funkce je definovaná předpisem $X = A \vee B$. Hradlo OR se vyrábí jako integrovaný obvod 7432. Schematická značka je na obrázku 5.



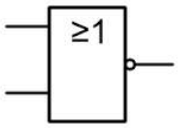
Obrázek 5.: OR značka

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Tabulka 4.: OR stavová tabulka

3.1.5. Hradlo NOR

Hradlo NOR vznikne spojením hradel OR a NOT tak, že na výstup hradla OR připojíme hradlo NOT. Hradlo NOR nastaví na výstup hodnotu logická 1, pouze v případě, když na obou vstupech hradla je hodnota logická 0, v ostatních případech je na výstupu hodnota logická 1 (viz tabulka 5.). Tato funkce je definovaná předpisem $X = (\overline{A \vee B})$. Hradlo NOR se vyrábí jako integrovaný obvod 7402. Schematická značka je na obrázku 6.



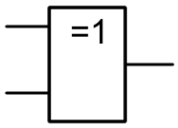
Obrázek 6.: NOR značka

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Tabulka 5.: NOR stavová tabulka

3.1.6. Hradlo XOR

Hradlo XOR, jinak exclusive OR, implementuje binární logickou operaci exkluzivní součet. Na výstupu hradlo nastaví hodnotu logická 1, pokud na vstupech A i B jsou shodné logické hodnoty (viz tabulka 6.). Tato funkce je definovaná předpisem $X = A \oplus B$. Na obrázku 7. je zobrazena schematická značka hradla. Hradlo XOR se vyrábí jako integrovaný obvod 74386.



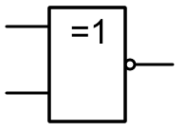
Obrázek 7.: XOR značka

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Tabulka 6.: XOR stavová tabulka

3.1.7. Hradlo XNOR

Hradlo XNOR implementuje binární logickou operaci ekvivalence. Vznikne spojení hradel XOR a NOT tak, že na výstup hradla XOR zapojíme hradlo NOT. Na výstupu hradlo nastaví hodnotu logická 1, pokud na vstupech A i B jsou shodné logické hodnoty (viz tabulka 7.). Tato funkce je definovaná předpisem $X = (\overline{A \oplus B})$. Na obrázku 8. je zobrazena schematická značka hradla. Hradlo XNOR se vyrábí jako integrovaný obvod 74266.



Obrázek 8.: XNOR značka

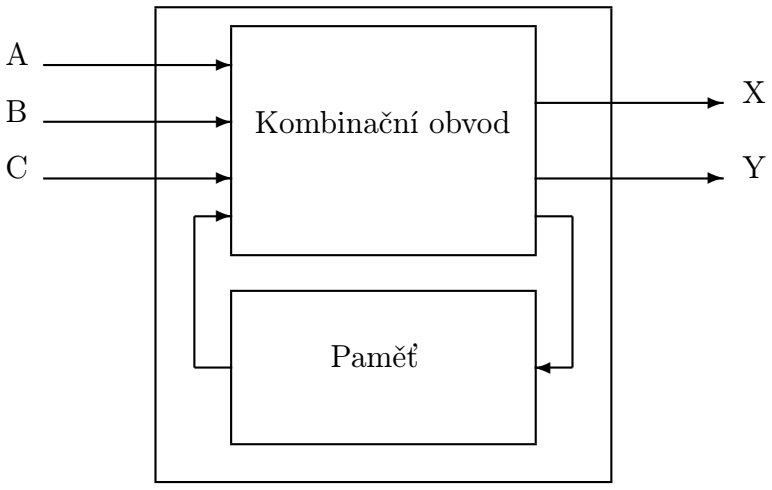
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Tabulka 7.: XNOR stavová tabulka

3.2. Sekvenční číslicové obvody

Sekvenční číslicové obvody, dále jen *SO*, jsou obvody u nichž logické hodnoty na výstupech X, Y... jsou závislé na kombinaci logických hodnot na vstupech A, B... a současně na předchozích stavech obvodu. *SO* se skládá s *KO* a paměťového členu, který uchovává informaci o předchozím stavu obvodu. Paměťový člen je implementován jako kombinační obvod, ve kterém je zavedena zpětná vazba. Tyto obvody nazývají *klopné obvody*.

SO se dělí na *synchronní*, které jsou řízené hodinovým signálem tzn. že na změnu logických hodnot na vstupech reagují až po přivedení náběžné či sestupné hrany hodinového signálu, a *asynchronní*, které nejsou řízené a na změnu logických hodnot na vstupech reagují ihned.



Obrázek 9.: Sekvenční číslicový obvod.

Klopný obvod je obvod, který může nabývat dvou odlišných stavů. Dělí se na *astabilní, monostabilní, bistabilní* a *Schmittův*.

Astabilní klopný obvod je obvod, který nemá stabilní stav, tzn. že obvod se neustále překlápí z jednoho stavu do druhého (osciluje). Tyto obvody se využívají jako generátory impulzů. Nejznámější implementací je integrovaný obvod 555.

Monostabilní klopný obvod je obvod, který má jeden stabilní stav, ze kterého je možné jej přepnou do nestabilního stavu. Obvod se sám přepne po určité době zpět do stabilního stavu. Tyto obvody se využívají jako časovače.

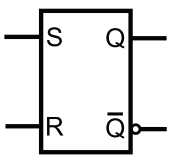
Bistabilní klopné obvody jsou obvody, které mají dva stabilní stavy, mezi kterými lze přepínat pomocí logických hodnot na vstupech obvodu. Tyto obvody mají mnoho variant provedení: *RS*, *JK*, *D*...

Schmittův klopný obvod je obvod, který slouží k úpravě tvaru impulzů. Využívá se také jako analogově digitální převodník.

3.2.1. Klopný obvod RS

RS je jedním ze základních klopných obvodů. Lze realizovat zapojením pomocí dvou hradel NOR či NAND se zpětnou vazbou tak, že výstup prvního hradla je zaveden na vstup druhého a výstup druhého hradla je zaveden na vstup prvního viz obrázek 11. Jedná se o jednobitový paměťový modul. Vstup S se nazývá Set a vstup R Reset viz obrázek 10. *RS* obvod existuje v asynchronní i synchronní variantě.

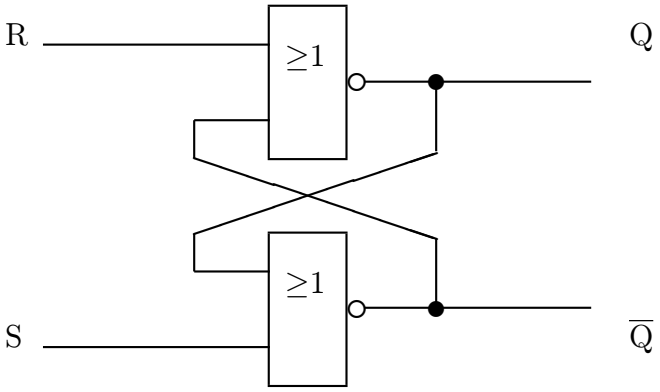
Pokud je na vstup S přivedena hodnota logická 1 a současně je na vstupu R hodnota logická 0, pak obvod na výstup Q nastaví hodnotu logická 1 a na výstup $\overline{Q}$ hodnotu logická 0 (provede se tzv. *set obvodu*). Pokud je na vstup R přivedena hodnota logická 1 a současně je na vstupu S hodnota logická 0, pak na výstup Q obvod nastaví hodnotu logická 0 a na výstup $\overline{Q}$ hodnotu logická 1 (provede se tzv. *reset obvodu*). Ve chvíli, kdy jsou na obou vstupech hodnoty logická 0, pak obvod zachovává svůj předchozí stav (jedná se o tzv. *paměťový stav*). V opačném případě, kdy na vstupech je hodnota logická 1, se jedná o tzv. *zakázaný stav*, kdy není jasné v jakém stavu bude obvod viz tabulka 8.



Obrázek 10.: RS značka

S	**R**	$\mathbf{Q}_n$	$\overline{\mathbf{Q}}_n$	**Popis**
0	0	Q_{n-1}	$\overline{Q}_{n-1}$	Paměťový stav
0	1	0	1	Reset obvodu
1	0	1	0	Set obvodu
1	1	X	X	Zakázaný stav

Tabulka 8.: RS stavová tabulka



Obrázek 11.: RS klopný obvod pomocí hradel NOR.

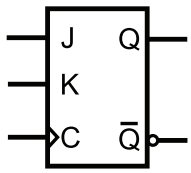
3.2.2. Klopný obvod JK

Dalším klopným obvodem je obvod *JK*, který se vyrábí pouze v synchronní formě, tzn. že obvod je řízen hodinovým signálem. Na obrázku 12. je schematická značka obvodu *JK* řízeného náběžnou hranou hodinového signálu. Mimo náběžné hrany může být obvod *JK* řízen i sestupnou hranou a nebo úrovní hodinového signálu. Obvod *JK* je nazván podle svého vynálezce jménem *Jack Kilby*[3], který tento obvod přestavil v roce 1958.

Obvod *JK* plní stejnou funkci jako obvod *RS*. Vstup J obvod setuje a vstup K obvod resetuje. V okamžiku, kdy na obou vstupech jsou hodnoty logická 1, obvod neguje hodnoty výstupů v intervalech hodinového signálu na vstupu C viz tabulka 9.

Jednou z možností, jak využít obvod *JK*, je zapojení nazývané *JK Master-Slave*. To vznikne zapojením dvou obvodů *JK* kaskádovitě za sebou. Výstupy Q a $\overline{Q}$ prvního obvodu *Master* jsou zapojeny na vstupy J a K druhého obvodu *Slave*. Obvod *Master* je řízen náběžnou hranou hodinového signálu, kdežto obvod *Slave* je řízen hranou sestupnou. Při náběžné hraně obvod *Master* načte logické hodnoty na vstupech a nastaví dle kombinace logické hodnoty na své výstupy, které jsou připojeny na vstupy obvodu *Slave*. Při sestupné hraně obvod *Slave* přečte logické hodnoty na svých vstupech a dle kombinace nastaví logické hodnoty na výstupy.

[3]Jack Kilby byl významným americkým elektroinženýrem, v roce 2000 získal Nobelovu cenu za fyziku.



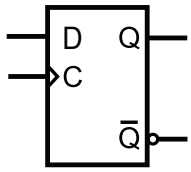
Obrázek 12.: JK značka

J	**K**	**C**	$\mathbf{Q}_n$	$\overline{\mathbf{Q}}_n$
0	0	↑	Q_{n-1}	$\overline{Q}_{n-1}$
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	$\overline{Q}_{n-1}$	Q_{n-1}
X	X	1	Q_{n-1}	$\overline{Q}_{n-1}$

Tabulka 9.: JK stavová tabulka

3.2.3. Klopný obvod D

Stejně jako klopný obvod *RS* a *JK*, tak i obvod *D* implementuje jednobitovou paměť [3]. Obvod *D* vznikne z obvodu *RS* tak, že připojíme negovaný vstup *S* na vstup *R*. Obvod *D* má dva vstupy viz obrázek 13. Vstup *D* slouží pro data, vstup *C* slouží jako řídící vstup obvodu. Obvod kopíruje logické hodnoty na datovém vstupu *D* na výstup *Q*, a to vždy ve chvíli, kdy na vstupu *C* je náběžná hrana. V ostatních případech je hradlo v paměťovém módu. Tato funkce je popsána v tabulce 10.



Obrázek 13.: D značka

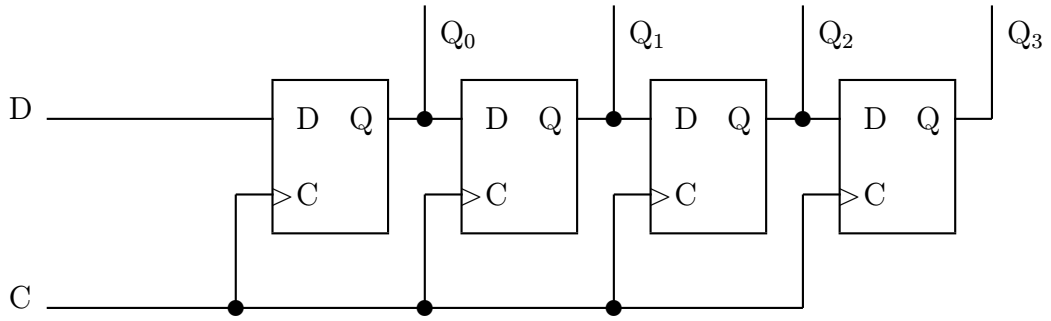
D	**C**	Q_n	$\overline{\mathbf{Q}}_n$
0	↑	0	1
1	↑	1	0
1	1	Q_{n-1}	$\overline{Q}_{n-1}$
1	0	Q_{n-1}	$\overline{Q}_{n-1}$
X	↓	Q_{n-1}	$\overline{Q}_{n-1}$

Tabulka 10.: D stavová tabulka

3.2.4. Posuvné registry

Registry jsou *SO*, které umožňují vkládání a uchování informací v binární podobě. Posuvný registr vždy při náběžné hraně provede načtení dat z datového vstupu na první výstup a současně provede posun dat z výstupu na následující výstup.

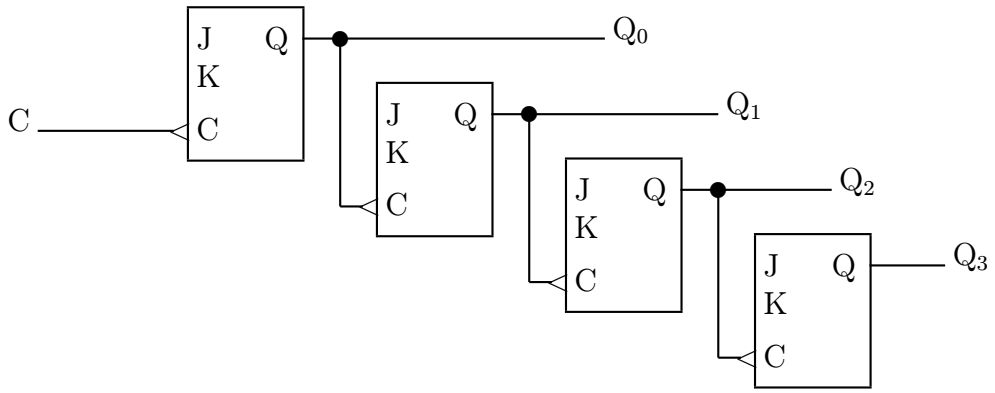
Posuvný registr vznikne vhodným zapojením jednodušších klopných obvodů, např. obvodů *D* tak, že výstup *Q* obvodu je zapojený do vstupu *D* následujícího obvodu, jak je vidět na obrázku 14.



Obrázek 14.: Zapojení 4-bitového posuvného registru.

3.2.5. Čítače

Čítače jsou *SO*, které čítají vstupní impulzy a zobrazují výsledek na výstupu v určité podobě. Podle podoby výstupního výsledku se čítače dělí na binární, dekadické atd... Čítače lze realizovat pomocí klopných obvodů. Jedno z možných zapojení pomocí obvodů *JK*, které mají vstupy J a K nastaveny na hodnotu logická jedna, je znázorněno na obrázku 15. Čítače se v praxi vyrábějí jako integrované obvody, například čítač vpřed jako obvod 7490.



Obrázek 15.: Zapojení čítače pomocí JK obvodů

4. Algoritmus pro simulaci schemat číslicových obvodů

V této kapitole je popsán algoritmus pro simulaci sestavených schemat s číslicovými obvody v aplikaci *SCEditor*.

4.1. Popis algoritmu pro simulaci schemat

Číslicové obvody implementují booleovské funkce. Na základě logických hodnot na vstupech a booleovské funkce, číslicový obvod nastaví logické hodnoty na své výstupy. Dále tento postup nazývejme výpočet obvodu. Simulace sestaveného schematu probíhá tak, že se nejdříve pro všechny číslicové obvody provede výpočet obvodu. V dalším cyklu všechny číslicové obvody nastaví své výstupní hodnoty na vstupy připojených číslicových obvodů. Algoritmus z textu je popsán na obrázku 16.

- Pseudokód obsahuje jednu proceduru s názvem *RunSimulation*. Procedura *RunSimulation* v cyklu spouští simulaci sestaveného schematu.
- Procedura *RunCompute*, volaná v rámci procedury *ComputeGates*, provede výpočet obvodu.
- Procedura *SetOutputGateInputValue*, volaná v rámci procedury *ComputeGates*, provede nastavení vypočtených výstupních logických hodnot na vstupy připojených číslicových obvodů.
- Funkce *IsActiveSimulation*, volaná v rámci procedury *RunSimulation*, ověřuje, zda uživatel neprovedl ukončení simulace.
- Proměnná *arrayGates* obsahuje seznam číslicových obvodů ve schematu.

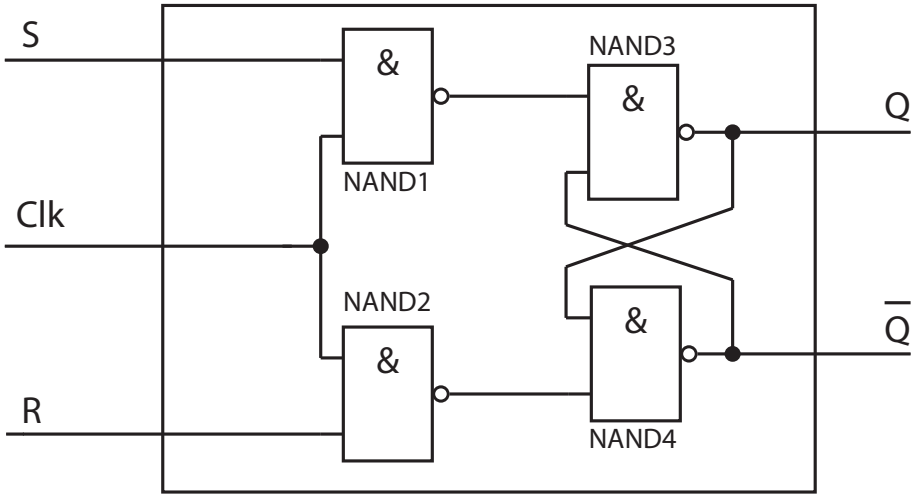
```
1: procedure RunSimulation(arrayGates)
2:     while IsActiveSimulation() do
3:         for all gate in arrayGates do
4:             RunCompute(gate)
5:         end for
6:         for all gate in arrayGates do
7:             SetOutputGateInputValue(gate)
8:         end for
9:     end while
10: end procedure
```

Obrázek 16.: Algoritmus pro výpočet logických hodnot schematu

4.2. Aplikace algoritmu pro simulaci schemat

Nyní na příkladu zapojení s číslicovými obvody předvedu aplikaci algoritmu pro výpočet logických hodnot, který je uveden v části 4.1. V příkladu je číslicový obvod *RS*, který je zapojen pomocí hradel NAND. Obvod *RS* je zobrazen na obrázku 17.

Ve výchozím stavu, kdy dojde k sestavení schématu a na všech vstupech je hodnota logická 0, je na výstupu Q hodnota logická 0 a na výstupu $\overline{Q}$ hodnota logická 1. Nyní nastavíme na vstupy R a Clk hodnotu logická 1 a spustíme simulaci.



Obrázek 17.: Zapojení obvodu *RS* pomocí obvodů Nand

- Procedura *RunSimulation* je volaná s polem číslicových obvodů. V našem případě jsou to hradla NAND1, NAND2, NAND3 a NAND4.
- Uživatel simulaci nezastavil, algoritmus pokračuje do těla cyklu while.
- Nyní se pro všechna hradla provede spuštění výpočtu obvodu.
  - Hradlo NAND1 po výpočtu nastaví na výstup hodnotu logická 0.
  - Hradlo NAND2 po výpočtu nastaví na výstup hodnotu logická 1.
  - Hradlo NAND3 po výpočtu nastaví na výstup hodnotu logická 0.
  - Hradlo NAND4 po výpočtu nastaví na výstup hodnotu logická 1.
- Nyní se pro všechna hradla provede předání výstupní logické hodnoty na vstupy připojených hradel.
  - Hradlo NAND1 nastaví na první vstup hradla NAND3 hodnotu logická 0.

  - Hradlo NAND2 nastaví na druhý vstup hradla NAND4 hodnotu logická 1.
  - Hradlo NAND3 nastaví na první vstup hradla NAND4 hodnotu logická 0.
  - Hradlo NAND4 nastaví na druhý vstup hradla NAND3 hodnotu logická 1.

- Uživatel simulaci nezastavil, algoritmus opět vstupuje do těla cyklu while.
- Nyní se pro všechna hradla provede spuštění výpočtu obvodu.
  - Hradlo NAND1 po výpočtu nastaví na výstup hodnotu logická 0.
  - Hradlo NAND2 po výpočtu nastaví na výstup hodnotu logická 1.
  - Hradlo NAND3 po výpočtu nastaví na výstup hodnotu logická 1.
  - Hradlo NAND4 po výpočtu nastaví na výstup hodnotu logická 0.
- Nyní se pro všechna hradla provede předání výstupní logické hodnoty na vstupy připojených hradel.
  - Hradlo NAND1 nastaví na první vstup hradla NAND3 hodnotu logická 0.
  - Hradlo NAND2 nastaví na druhý vstup hradla NAND4 hodnotu logická 1.
  - Hradlo NAND3 nastaví na první vstup hradla NAND4 hodnotu logická 0.
  - Hradlo NAND4 nastaví na druhý vstup hradla NAND3 hodnotu logická 1.

5. Architektura a funkce aplikace SCEditor

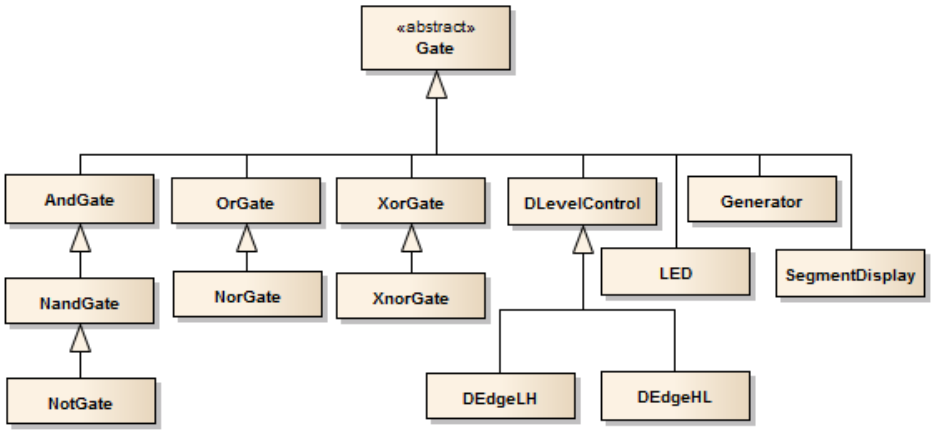
Pro vytvoření aplikace jsem se rozhodl využít objektově orientovaného programování, dále jen *OOP*. Čtení následujícího textu je podmíněno znalostí *OOP*. Číslicové obvody jsou tedy reprezentovány objekty, které obsahují atributy a metody. Aplikaci *SCEditor* jsem rozdělil do dvou vrstev. První vrstva obsahuje implementaci logiky číslicových obvodů. Druhá vrstva obsahuje implementaci editoru číslicových obvodů.

5.1. Architektura a funkce logiky číslicových obvodů

Při návrhu logické vrstvy číslicových obvodů jsem přemýšlel nad společnými vlastnostmi číslicových obvodů. Číslicové obvody mají vstupy a výstupy. Na každý výstup číslicového obvodu je možno připojit vstupy libovolného množství dalších číslicových obvodů. Každý číslicový obvod na základě kombinace vstupních logických hodnot nastaví na výstupy určité logické hodnoty, dle booleovské funkce, kterou vyčísluje.

Po přezkoumání společných vlastností jsem se rozhodl, že veškerou společnou funkcionalitu bude implementovat abstraktní třída *Gate*. Všechny třídy popisující konkrétní číslicové obvody dědí ze třídy *Gate*, jak je vidět v UML diagramu na obrázku 18. Výhodou *OOP* je také, že objekty tříd, které dědí ze třídy *Gate*, jsou současně i objekty typu *Gate*.

Jelikož číslicové obvody mají různé počty vstupů a výstupů, rozhodl jsem se, že třída *Gate* bude obsahovat dvě pole typu bool. Pole jsem zvolil z důvodu pevně dané velikosti počtu vstupů a výstupů. Jedno pole s názvem *input* reprezentuje vstupy číslicového obvodu a druhé s názvem *output* reprezentuje výstupy číslicového obvodu.



Obrázek 18.: Diagram tříd třídy *Gate* a některých jejich potomků

Konstruktor třídy *Gate* obsahuje dva argumenty, jeden pro určení počtu vstupů, druhý pak pro určení počtu výstupů. Argumenty konstruktoru pak definují velikost obou polí. Každá třída, která dědí ze třídy *Gate*, pak ve svém konstruktoru volá konstruktor třídy *Gate* s konkrétním počtem vstupů a výstupů, dle toho, jaký číslicový obvod reprezentuje.

Třída *Gate* obsahuje také pole seznamů s názvem *outputGates*, které reprezentuje připojené číslicové obvody. Každý prvek pole reprezentuje jeden výstup číslicového obvodu. Na každý výstup může být připojeno libovolné množství číslicových obvodů, což jsou v našem případě objekty typu *Gate*. Každý prvek seznamu musí uchovávat navíc i informaci, který vstup připojeného číslicového obvodu je připojen k výstupu daného číslicového obvodu. K tomuto účelu bylo vhodné vytvořit třídu obsahující jak informaci o vstupu, což je index pole *input*, tak i o samotném číslicovém obvodu, což je reference na objekt typu *Gate*.

5.2. Architektura a funkce editoru

Na začátku návrhu jsem měl jasnou vizi, aby byla aplikace *SCEditor* interaktivní. Tomu jsem podřídil i návrh editoru. Editor poskytuje funkcionalitu pro práci s číslicovými obvody, k tomu využívá vrstvu s logikou číslicových obvodů.

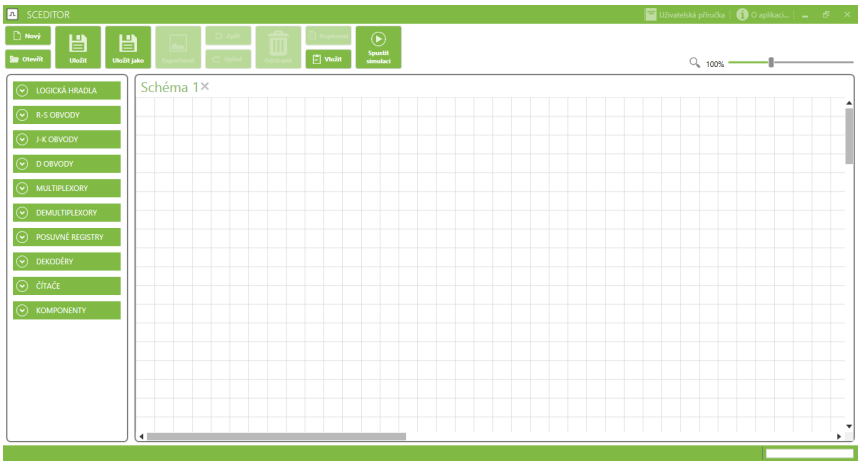
5.2.1. Prezentační vrstva editoru

Jako první jsem navrhl třídu *GUIGate*. Jedním z atributů třídy, s názvem *LogicGate*, je číslicový obvod, tedy objekt typu *Gate*. Při práci s aplikací *SCEditor* tedy uživatel pracuje s třídami *GUIGate*. Třída *GUIGate* zobrazuje schematickou značku daného číslicového obvodu pomocí .NET třídy *Image*. Konstruktor třídy *GUIGate* přebírá řetězec s názvem druhu číslicového obvodu. Na základě toho zvolí vhodnou schematickou značku pro zobrazení a také vhodného potomka třídy *Gate* do atributu *LogicGate*.

Jako další bylo potřeba vyřešit, kde se bude s objekty typu *GUIGate* pracovat a kde se budou zobrazovat. Navrhl jsem tedy třídu *GateCanvas*, která funguje jako kontejner pro objekty typu *GUIGate*. *GUIGate* reprezentuje plátno pro práci s číslicovými obvody. Umožňuje pohyb objektů typu *GUIGate*, rotace a také zobrazení schematické značky číslicového obvodu.

Další potřebnou částí prezentační vrstvy bylo vytvoření třídy s názvem *Connector* pro reprezentaci spojů mezi číslicovými obvody. Tato třída plní pouze vizuální úlohu. Spoje mezi číslicovými obvody se vytváří pomocí funkcionality *drag&drop*.

Důležitou součástí návrhu editoru bylo vytvoření návrhu rozmístění ovládacích prvků v okně aplikace *SCEditor*. Nejvíce prostoru okna aplikace *SCEditor* vyplňuje plocha pro práci s číslicovými obvody, což je objekt typu *GateCanvas*. Seznam dostupných číslicových obvodů jsem umístil vlevo vedle plátna. Lištu s nástroji jsem umístil do vrchní části. Navržené GUI je na obrázku 19.



Obrázek 19.: Uživatelské rozhraní aplikce *SCEditor*

5.2.2. Logická část editoru

Ukládání a načítání sestavených schemat číslicových obvodů probíhá pomocí binární serializace a deserializace. Při ukládání se veškerý obsah objektu typu *GateCanvas*, což jsou objekty typu *GUIGate* a *Connector*, převede pomocí serializace na proud bytů a ten se uloží do souboru s koncovkou *sce*. Při načítání se přečte obsah souboru s koncovkou *sce* a pomocí deserializace se přečtený proud bytů převede na objekt typu *GateCanvas*.

Funkcionalita kopírování a vkládání objektů typu *GUIGate* je také realizování pomocí serializace a deserializace. Při kopírování se nejdříve provede serializace a následně se proud bytů uloží do schránky. Při vložení se obsah schránky deserializuje a vzniklé objekty vloží do objektu typu *GateCanvas*.

Pro návrh funkcionality pro vracení úprav v aplikaci *SCEditor* vpřed a zpět jsem navrhl abstraktní třídu *EditorCommand*. Ta deklaruje metody *Do* a *Undo* a také obsahuje atribut *ID*, který vrací jednoznačný identifikátor. Všechny třídy implementující konkrétní operace, jako například operace posunu, rotace, spojení konektorem a další, dědí ze třídy *EditorCommand*.

Dále jsem navrhl třídu *UndoRedoManager*, která obsahuje dva zásobníky pro objekty typu *EditorCommand*. První s názvem *UndoStack*. Druhý s názvem *RedoStack*. Dále obsahuje metody *AddCommand*, *Undo* a *Redo*. Před provedením operace, například operace posunutí číslicového obvodu, se vytvoří nový objekt typu *MoveCommand*. Zavolá se metoda *Do* a objekt je vložen na zásobník. Při zavolání metody *Undo* se nejdříve ověří, jestli není *UndoStack* prázdný. Pokud není prázdný, zavolá metodu *Undo* objektu na vrcholu zásobníku a provede přesun objektu ze zásobníku *UndoStack* na vrchol zásobníku *RedoStack*. Při zavolání metody *Redo* se ověří, jestli není *RedoStack* prázdný. Pokud není prázdný, zavolá metodu *Do* objektu na vrcholu zásobníku *RedoStack* a provede se přesun objektu na vrchol zásobníku *UndoStack*.

6. Implementace aplikace SCEditor

V této kapitole jsou popsány třídy aplikace *SCEditor*, použité nástroje a technologie. Následující text slouží jako programátorská příručka.

6.1. Použité nástroje a technologie

Aplikace *SCEditor* je implementována pro systém Microsoft Windows 7 a novější.

Pro implementaci logické vrstvy aplikace jsem tedy zvolil platformu .NET 4.5 za použití jazyka C#, který preferuji z osobních důvodů.

Pro implementaci prezentační vrstvy jsem zvolil technologii WPF[4] (Windows Presentation Foundation), která podporuje nativní akceleraci 2D grafiky. Další výhodou technologie WPF oproti starší WF (Windows Forms) je využití vektorové grafiky, díky které lze provádět graficky bezztrátovou změnu velikosti prvků. WPF zavádí pro prezentační vrstvu jazyk XAML (e**X**tensible **A**pplication **M**arkup **L**anguage). XAML je značkovací jazyk, stejně jako například HTML či XML.

Při implementaci jsem používal *Microsoft Visual Studio 2013 Ultimate*. Dále jsem pro vytvoření animací použil nástroj *Blend for Visual Studio*. Hradla jsem kreslil pomocí *Microsoft Expression Studia*, ze kterého jsem obrázky exportoval přimo do jazyka XAML.

Pro změnu barvy, stylu okna a dalších vizuálních prvků po vzoru nového stylu Microsoftu jsem využil knihovny *MahApps.Metro*, která je dostupná na webu *GitHub.*

6.2. Popis tříd logické vrstvy číslicových obvodů

Logickou vrstvu jsem implementoval jako knihovnu funkcí, jejíž výstupním zkompilovaným souborem je soubor *dll*. Knihovnu funkcí jsem zvolil z důvodu znovupoužitelnosti v jiných aplikacích.

Třída Gate je abstraktní třídou, implementující veškeré společné operace číslicových obvodů. Implementuje rozhraní *INotifyPropertyChanged*. Událost *PropertyChanged* je vyvolána vždy po změně stavu objektu typu *Gate*.

Třída Pin je třídou reprezentující propoj mezi číslicovými obvody. Obsahuje dva atributy. Jeden s názvem *ConnectedGate* pro referenci na objekt typu *Gate*. Druhý typu *int* pro index v poli *input* objektu v atributu *ConnectedGate*.

[4]WPF je dostupný ve windows již od .NET verze 3.0, tedy od roku 2006, od té doby WPF prošlo značným vývojem a rozšířením.

Potomci třídy Gate implementují metodu *Compute*. Metoda *Compute* bere jako argument pole *input* a jeho výstupem je pole typu *bool*. Dále implementují metodu *Identifikator*, která vrací řetězec s názvem číslicového obvodu.

Třída Simulator obsahuje metodu *RunSimulation*, která prochází všemi objekty typu *Gate* a spouští jejich metodu *RunCompute*.

6.3. Popis tříd editoru

Prezentační vrstva je implementována pomocí technologie WPF. Výstupem je spustitelný soubor *exe*. Ve WPF se vzhled prvků a rozložení definuje pomocí jazyka XAML, kdežto logika a reakce na události pomocí jazyka C#. Tomuto přístupu se říká *Code behind*. Tohoto přístupu jsem využil při vývoji prezentační vrstvy. Všechna okna a všechny prvky jsem definoval pomocí jazyka XAML, pomocné třídy a reakce na události oken jsem implementoval pomocí jazyka C#.

Třída GUIGate je třídou, umožňující práci uživatelů s číslicovými obvody. Třída *GUIGate* dědí z .NET třídy *UserControl*. Vizuální část třídy *GUIGate* je definována pomocí jazyka XAML, reakce na události a pomocné metody pomocí jazyka C#. Atribut *LogicGate* obsahuje referenci na objekt typu *Gate*. Třída *GUIGate* implementuje reakci na událost *PropertyChanged* objektu typu *Gate*, která se jmenuje *PropertyChangedMethod*.

Třída Connector plní pouze vizuální funkci. Obsahuje atribut typu *Polyline*, což je .NET třída vykreslující lomenou čáru. Třída *Connector* obsahuje dva atributy typu *Point*. Ty slouží pro informaci pro počáteční a koncový bod konektoru. Dále obsahuje metody pro úpravu vykreslování lomené čáry.

Třída GateCanvas reprezentuje plátno pro práci s číslicovými obvody. Dědí z .NET třídy *Canvas*, která slouží pro kreslení a zobrazování objektů typu *UserControl* nebo *UIElement*. Je to tedy třída sloužící jako kontejner objektů. Třída *GateCanvas* umožňuje vkládání, pohyb a rotace objektů typu *GUIGate*. Implementuje rozhraní *INotifyPropertyChanged*. Událost *PropertyChanged* je vyvolána vždy při změně jakéhokoliv objektu typu *GUIGate*.

Třída GatePicker slouží jako nabídka dostupných číslicových obvodů. Dědí z .NET třídy *UserControl*. Obsahuje prvky typu *Expander*. *Expander* je .NET třída a slouží jako kontejner. *Expander* přepíná mezi dvěma stavy, první stav se nazývá *collapsed*, druhý *expanded*. Ve stavu *collapsed* jsou všechny objekty, které *Expander* obsahuje, skryté. Ve stavu *expanded* jsou viditelné. *Expander* obsahuje objekty typu *Button*. Při dvojkliku na *Button* se zavolá metoda, která vytvoří nový objekt typu *GUIGate* a přidá ho do kontejneru

GateCanvas. Tato metoda je vyvolána i pomocí funkcionality *drag&drop*, kde *drag* objekt je typu *Button* a *drop* objekt je typu *GateCanvas*.

Třída EditorWindow je samotné okno aplikace *SCEditor*. Je potomkem třídy *MetroWindow* z knihovny *MahApps.Metro*.

Třída ExportImport je třída implementující statické metody *SaveSchema*, *LoadSchema* a *ExportToPng*. Metody *SaveSchema* a *ExportToPng* přebírají jako argument objekt typu *GateCanvas*. Metody *SaveSchema* a *LoadSchema* používají binární serializaci pro zakódování a dekódování binárních dat. Pro uložení dat ze souboru a načtení dat ze souboru využívají obě metody objektu typu *Stream* a jeho metody *FileOpen*. Metoda *ExportToPng* využívá pro vytvoření souboru typu png objekt typu *RenderTargetBitmap*.

Třída EditorCommand je abstraktní třída. Obsahuje deklaraci abstraktních metod *Do* a *Undo*. Dále obsahuje vlastnost *ID*. Ze třídy *EditorCommand* dědí veškeré třídy, které implementují operace s objekty typu *GUIGate*. Třídy, které dědí ze třídy *EditorCommand*, musí implementovat metody *Do* a *Undo*.

Třída UndoRedoManager obsahuje dva atributy typu *Stack*. Ty obsahují objekty typu *EditorCommand*. Dále obsahuje metody *Undo*, *Redo*, *CanUndo* a *CanRedo*.

Třída EditorHelper obsahuje statické metody, které jsou volány v rámci reakcí na události objektu typu *EditorWindow*.

7. Uživatelská příručka aplikace SCEditor

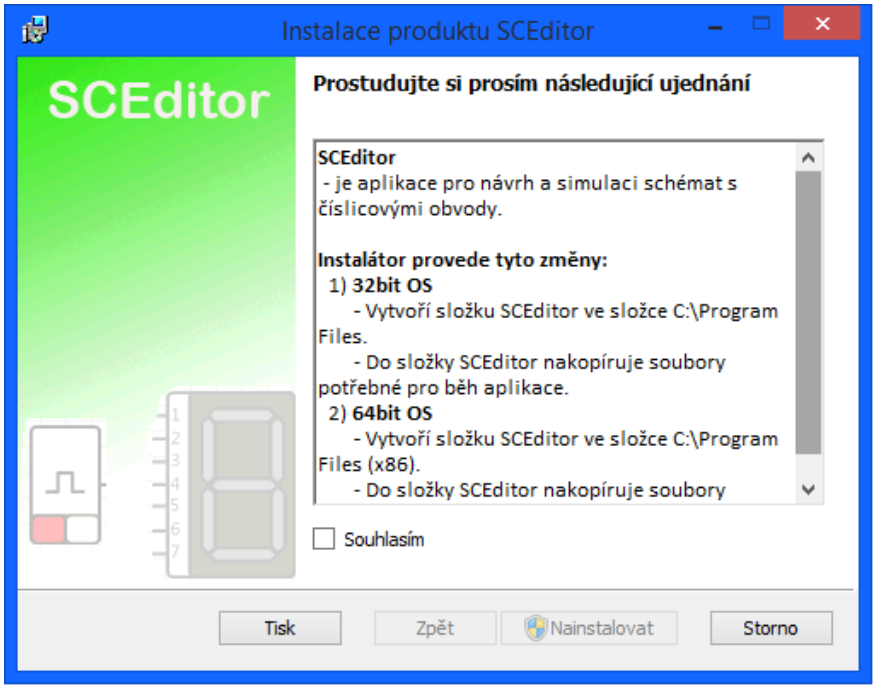
Tato kapitola obsahuje popis práce s aplikací *SCEditor* a slouží jako uživatelská příručka.

7.1. Požadavky

Pro plynulý běh aplikace *SCEditor* je potřeba, aby na počítači byl nainstalován Microsoft Windows 7 nebo novější. Dále je nutné mít nainstalován .NET framework 4.5. Aplikace je schopna běžet na fyzickém i virtuálním počítači.

7.2. Instalace aplikace SCEditor

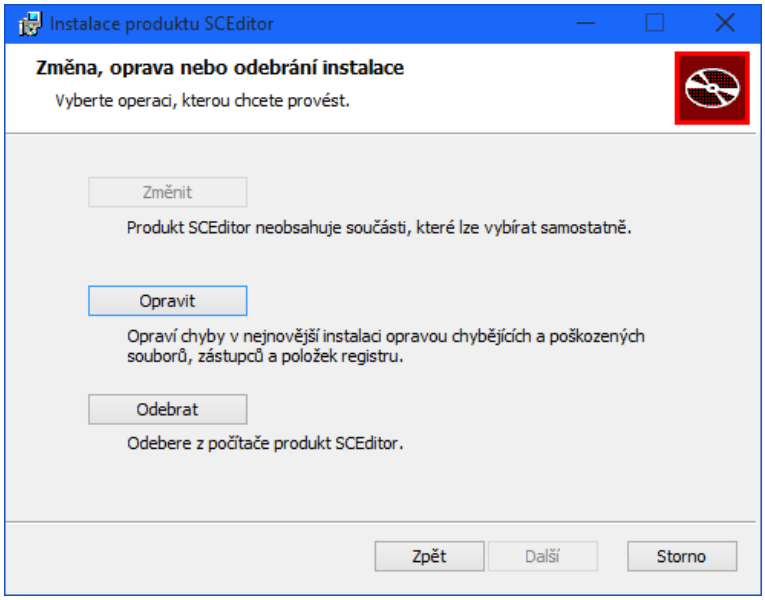
Instalace aplikace probíhá pomocí vytvořeného instalátoru *SCEditor.msi*. Po spuštění instalátoru se otevře průvodce instalace, viz obrázek 20. V průvodci instalace je nyní text, ve kterém jsou informace o aplikaci a také jaké se po instalaci provedou změny v operačním systému počítače. Pokud chceme dále pokračovat, potvrdíme tyto změny a klikneme na tlačítko *Nainstalovat*. Nyní instalátor provede instalaci aplikace *SCEditor*. Po instalaci instalátor vytvoří na ploše zástupce s názvem *SCEditor*, přes kterého můžeme aplikaci *SCEditor* pohodlně spouštět.



Obrázek 20.: Instalátor aplikace *SCEditor*

7.3. Odinstalace aplikace SCEditor

Pro odinstalaci aplikace *SCEditor* stačí spustit instalátor *SCEditor.msi*. Zobrazí se dialogové okno, viz obrázek 21. V něm klikneme na tlačítko *Odebrat*. Na další stránce klikneme na tlačítko *Odinstalovat* a aplikace *SCEditor* se z počítače odinstaluje.



Obrázek 21.: Odinstalace aplikace *SCEditor*

7.4. Návrh schemat s číslicovými obvody

K návrhu a simulaci zapojení s číslicovými obvody slouží v aplikaci *SCEditor* plátno. Zde se vkládají a propojují číslicové obvody.

Nové číslicové obvody se nacházejí v levém postranním panelu. Zde jsou číslicové obvody roztříděny do skupin. Po kliknutí na danou skupinu se skupina rozbalí a zobrazí se tlačítka s číslicovými obvody. Pro vložení číslicového obvodu na plátno stačí provést dvojklik na tlačítku s číslicovým obvodem. Číslicový obvod se zobrazí na plátně aplikace *SCEditor* v levém horním rohu.

Nový číslicový obvod lze na plátno umístit i tak, že klikneme levým tlačítkem myši na tlačítko s číslicovým obvodem a táhneme nad plátno. Nad plátnem tlačítko myši pustíme. Na místě na plátně, kde jsme tlačítko myši pustili, se zobrazí číslicový obvod.

Pro propojení číslicových obvodů klikneme na výstup číslicového obvodu na plátně levým tlačítkem myši a táhneme na vstup číslicového obvod. Nad vstupem číslicového obvodu pustíme tlačítko myši, vznikne nový propoj, který se zobrazí na plátně.

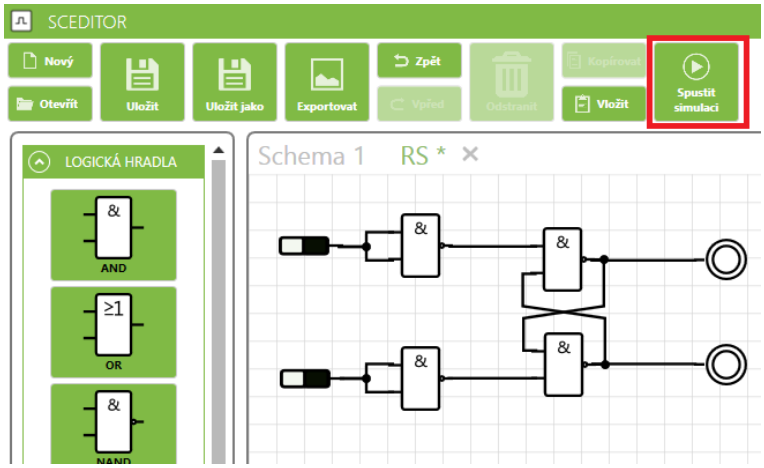
Pro vymazání číslicového obvodu nebo propoje z plátna je musíme nejdříve označit. Po označení stačí stisknout klávesu *del* nebo *delete*. Vymazáni lze také provést pomocí kontextové nabídky, kterou vyvoláme stisknutím pravého tlačítka myši nad číslicovým obvodem na plátně.

Číslicové obvody lze na plátně také posouvat. Pro posun klikneme na plátně na číslicový obvod levým tlačítkem myši a táhneme na chtěnou pozici na plátně. Pro umístění číslicového obvodu pustíme tlačítko myši.

Pokud chceme vrátit úpravy zpět, klikneme v nástrojové liště na tlačítko *Zpět*. Pokud naopak chceme přejít v úpravách vpřed, klikneme v nástrojové liště na tlačítko *Vpřed*.

7.5. Simulace schemat s číslicovými obvody

Pro simulaci sestaveného schematu klikneme v nástrojové liště na tlačítko *Spustit simulaci*, viz obrázek 22. V režimu simulace je editace schemat zakázána. V režimu simulace je možno zapínat a vypínat generátory a spínače ve schematu. Pro ukončení simulace klikneme na tlačítko *Zastavit simulaci*.

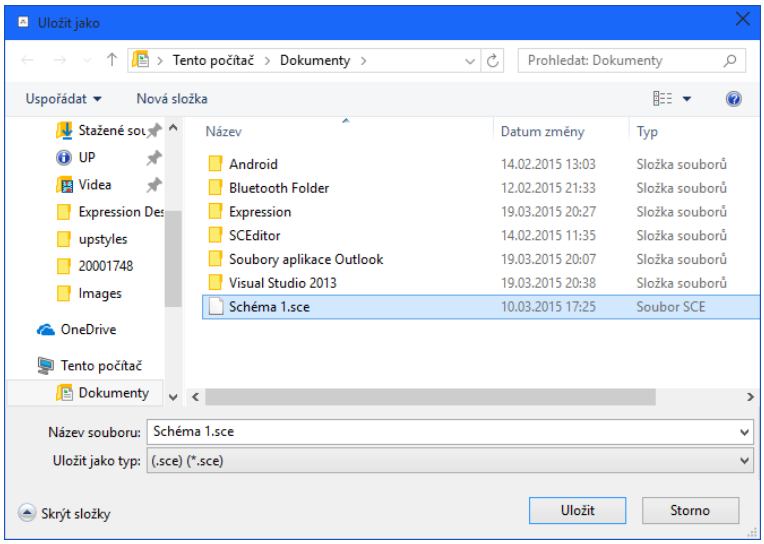


Obrázek 22.: Spuštění simulace schematu

7.6. Práce se soubory

Pro uložení sestaveného schematu v aplikaci *SCEditor* klikneme v nástrojové liště na tlačítko *Uložit*. Pokud jsme ještě dané schema neukládali, zobrazí se nové dialogové okno, viz obrázek 23. V dialogovém okně si vybereme umístění a název souboru, do kterého bude schema uloženo, a klikneme na tlačítko *Uložit*. Pokud jsme již schema měli uložené, při kliknutí na tlačítko *Uložit* se schema uloží do již vytvořeného souboru. Pokud chceme schema uložit do jiného souboru, klikneme na tlačítko *Uložit jako*, a otevře se dialogové okno pro výběr souboru pro uložení.

O úspěšném uložení aplikace *SCEditor* informuje pomocí nového dialogového okna.



Obrázek 23.: Dialog pro uložení schematu

Pokud chceme načíst již uložené schema, klikneme v nástrojové liště na tlačítko *Otevřít*. Otevře se nové dialogové okno, kde si vybereme soubor, ve kterém máme uložené schema, a klikneme na tlačítko otevřít. O úspěšném načtení schematu aplikace *SCEditor* informuje pomocí nového dialogového okna.

7.7. Export nakreslených schemat

Pro uložení schematu jako obrázku, klikneme v nástrojové liště na tlačítko *Exportovat*. Otevře se nové dialogové okno. V dialogovém okně vybereme umístění a název souboru s příponou .png, do kterého bude obrázek schematu uložen.

Závěr

V rámci práce vznikla aplikace *SCEditor*, která slouží pro práci s kombinačními a sekvenčními číslicovými obvody. Aplikace *SCEditor* umožňuje návrh schemat s číslicovými obvody a následnou simulaci funkce sestavených schemat. Sestavená schemata je možno exportovat do souboru PNG. Aplikace *SCEditor* podporuje možnost uložení schematu pro pozdější načtení a editaci.

Nejdůležitější část aplikace *SCEditor* je knihovna *GateLogic*, která implementuje výpočty booleovských funkcí číslicových obvodů a samotnou simulaci sestavených schemat. Dále implementuje komunikaci mezi číslicovými obvody a zasílání výstupních logických hodnot všem připojeným číslicovým obvodům.

Prezentační vrstva byla implementována pomocí technologie WPF. Poskytuje interaktivní rozhraní mezi uživatelem a knihovnou *GateLogic*.

Jelikož je aplikace *SCEditor* primárně určená pro studenty, poskytuje veškeré druhy číslicových obvodů, které jsou probírány na elektrotechnicky zaměřených středních školách.

Conclusions

The thesis deals with creating *SCEditor* application, that is used to work with combinational and sequential digital circuits. *SCEditor* application allows designing digital circuits diagrams and subsequent simulation of the designed circuits. Designed circuit diagrams can be exported to a PNG file. Application *SCEditor* also supports the option for saving designs for later retrieval and editing.

The most important part of the *SCEditor* application is a *GateLogic* library that implements the computation of Boolean functions of digital circuits and the simulation itself. It also implements the communication between digital circuits and sending output logic values to all connected digital circuits.

The presentation layer was implemented by using WPF. It provides an interactive interface between the user and the *GateLogic* library.

Since the *SCEditor* application is primarily designed for students, it provides all kinds of digital circuits, which are discussed at electro-technically oriented secondary schools.

Reference

[1] Matoušek, David. *Číslicová technika.* BEN, 2002. ISBN: 80-7300-025-3.

[2] Wakerly, John F. *Digital Design: Principles and Practices.* Prentice Hall, 2005. ISBN: 0131733494.

[3] Tony R. Kuphaldt. *Lessons In Electric Circuits, Volume IV – Digital.* Elektronická publikace, 2007.

[4] Christian Moser. *WPF Tutorial.net.* Elektronická publikace, 2011.

A. Obsah přiloženého CD

`bin/`
: Spustitelný soubor `SCEditor.exe` včetně knihovny `GateLogic.dll`.

`doc/`
: Dokumentace práce `Sekvencni-obvody.pdf`, vytvořená dle závazného stylu KI PřF. Všechny soubory nutné pro bezproblémové vygenerování PDF souboru (včetně skriptu `generate.bat`) v souboru `doc.zip`.

`src/`
: Kompletní zdrojové texty aplikace `SCEditor` se všemi potřebnými zdrojovými texty, knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu. Dále je v adresáři `src` obsažen soubor `Požadavky.txt`, který obsahuje požadavky pro sestavení aplikace.

`ReadMe.txt`
: Instrukce pro instalaci a spuštění aplikace `SCEditor`, včetně požadavků pro jeho provoz.

Navíc CD obsahuje:

`data/`
: Soubory s příponou `.sce`, které obsahují uložená schemata s číslicovými obvody pro aplikaci `SCEditor`.

`install/`
: Instalátor aplikace `SCEditor.msi`, který nainstaluje aplikaci včetně knihoven potřebných pro provoz aplikace.

`diagrams/`
: Diagram tříd knihovny `GateLogic` a diagram případu užití aplikace `SCEditor`.