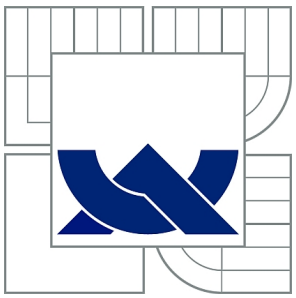
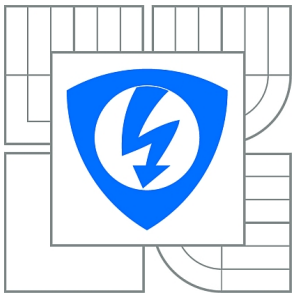



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY


FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV MIKROELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF MICROELECTRONICS


# GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ PRO NÁVRH FILTRŮ

GRAPHICAL USER INTERFACE FOR FILTER DESIGN

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS


AUTOR PRÁCE — JAN TESAŘÍK

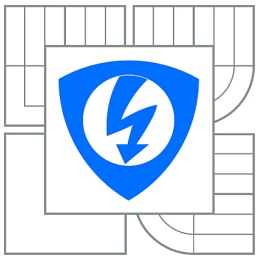
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARIÁN PRISTACH

SUPERVISOR



BRNO 2014



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**Fakulta elektrotechniky a komunikačních technologií**

**Ústav mikroelektroniky**

# Bakalářská práce

bakalářský studijní obor
**Mikroelektronika a technologie**

| | | | |
|---|---|---|---|
| ***Student:*** | Jan Tesařík | ***ID:*** | 146978 |
| ***Ročník:*** | 3 | ***Akademický rok:*** | 2013/2014 |

**NÁZEV TÉMATU:**

## Grafické uživatelské rozhraní pro návrh filtrů

**POKYNY PRO VYPRACOVÁNÍ:**

Seznamte se s možnostmi návrhu digitálních filtrů v programu Matlab. Vytvořte grafické uživatelské rozhraní pro návrh systémů založených na digitálních filtrech a podpůrných obvodech. Rozhraní bude sloužit jako grafická nadstavba na již vytvořený program umožňující generovat VHDL popis digitálních filtrů. Na tvorbu programu použijte programovací jazyk C++ a QT Framework. Program bude umožňovat načítání a ukládání návrhu systémů ve formátu XML a generování vrcholového popisu systémů v jazyce VHDL.

**DOPORUČENÁ LITERATURA:**

Podle pokynů vedoucího práce

| | | | |
|---|---|---|---|
| ***Termín zadání:*** | 10.2.2014 | ***Termín odevzdání:*** | 5.6.2014 |

***Vedoucí práce:*** Ing. Marián Pristach
***Konzultanti bakalářské práce:***

**doc. Ing. Jiří Háze, Ph.D.**
*Předseda oborové rady*

**UPOZORNĚNÍ:**

## ABSTRAKT

Práce se zabývá návrhem programu, který umožní generovat ze zapojení filtrů vrcholový popis v jazyce VHDL. Hlavním cílem je vytvořit grafickou nadstavbu na již existující program a propojit jej s generátorem VHDL. Program ukládá vytvořené zapojení do XML souboru a generuje z něj VHDL popis. Návrh probíhal v programovém prostředí Qt Framework a programovacím jazyce C++.

## KLÍČOVÁ SLOVA

filtr, FIR, CIC, IIR, Qt, XML, C++

## ABSTRACT

This work deals with design of program that allows to generate VHDL description from connection of filters. Its main aim is to create graphical extension on existing program and connect it to VHDL generator. Output from program will be XML file of created connection and generated VHDL description. Design of program was done by using Qt Framework environment and C++ language.

## KEYWORDS

filter, FIR, CIC, IIR, Qt, XML, C++

TESAŘÍK, J. *Grafické uživatelské rozhraní pro návrh filtrů*: Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 49 s. Vedoucí bakalářské práce Ing. Marián Pristach

# PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Grafické uživatelské rozhraní pro návrh filtrů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ..... .....
(podpis autora)

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Mariánu Pristachovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ..... .....
(podpis autora)

# OBSAH

# SEZNAM OBRÁZKŮ



# SEZNAM TABULEK

# ÚVOD

Tato práce se věnuje problematice návrhu systému s digitálními filtry, které stále častěji nahrazují analogové filtry. Rozšíření použití digitálních filtrů souvisí se zvyšováním hustoty integrace v integrovaných obvodech. Se zvyšováním složitosti systémů se zvyšuje potřeba vývoje prostředí, která by umožňovala efektivní návrh těchto systémů. Při návrhu zákaznických integrovaných obvodů (ASIC) je také potřeba přihlížet na požadavky kladené na vývoj systémů. Mezi hlavní požadavky většinou patří nízká výrobní cena, která je přímo úměrná ploše čipu, tj. kvalitě vygenerovaného HDL popisu celého systému. Výsledkem práce by mělo být prostředí, které by umožňovalo rychle a efektivně navrhnout systémy založené na digitálních filtrech.

Práce navazuje na program, který umožňoval pouze návrh kaskádního zapojení. Cílem je vytvořit program umožňující kaskádní i paralelní řazení filtrů s podpůrnými obvody a odstranit tak hlavní nevýhodu předchozího programu.

V teoretické části je rozebrána problematika filtrů FIR, CIC a IIR a jejich návrh v programovém prostředí MATLAB. Krátce se zabývá XML a jeho syntaxí. Nakonec teoretické části je zmínka o grafickém návrhovém prostředí Qt4 Framework a krátce popsány výhody jednotlivých podpůrných programů.

V další kapitole je rozebrán program, na který tato práce navazuje. Rozebírá jeho výhody a nevýhody. Dále je popsána funkce navrženého programu, který ukládá vytvořené zapojení do XML a poté ho předává programu pro generování VHDL. Poslední kapitola se zabývá ovládáním programu, vkládáním součástek a vytvářením zapojení.

# 1 TEORETICKÝ ÚVOD

## 1.1 Digitální filtry

Filtr v elektrotechnice obecně slouží pro změnu spektra výstupního signálu. U digitálních filtrů se jedná o změnu spektra diskrétního signálu, tzn. okamžitá hodnota se nemění spojitě v čase. Číslicové filtry mohou v určitých případech nahrazovat pasivní a aktivní analogové filtry.
Porovnání digitálních a analogových filtrů:

Vlastnosti digitálních filtrů

- vysoká přesnost,
- nemají drift,
- mohou mít lineární fázi (FIR),
- možnost adaptivní filtrace (tzv. samoučící filtrace),
- snadná simulace a návrh,
- výpočet musí proběhnout během periody vzorkování,
- filtrace nízkých frekvencí,
- nevhodné pro vf signály (omezeno výpočetní technikou).

Vlastnosti analogových filtrů

- menší přesnost,
- drift vlivem změn součástek,
- nelineární fáze,
- nelze použít adaptivní filtraci,
- obtížná simulace a návrh,
- vhodné pro vysoké frekvence.

Pokud nahrazujeme analogový filtr digitálním (číslicovým), je třeba použít A/D převodníky a poté případně D/A převodníky. Při převodu analogového signálu na digitální musíme signál vhodně navzorkovat (signál se stává diskrétním).

Vzorkování musí splňovat Shannonův-Kotělníkův teorém o dvojnásobné vzorkovací frekvenci:

$$f_s \geq 2 \cdot f_{MAX} \tag{1.1}$$

kde $f_\mathrm{s}$ je vzorkovací frekvence (s jako „sampling“) a $f_\mathrm{MAX}$ je maximální frekvence obsažená ve vzorkovaném signálu.

Pokud není splněn vzorkovací teorém, při zpětné rekonstrukci diskrétního signálu na analogový může dojít k aliasingu (periodizovaný signál se překrývá).

Číslicové filtry jsou zpravidla popisovány matematickými rovnicemi. Diferenční rovnice vyjadřuje závislost mezi posloupnostmi a jejich diferencemi:

$$b_s y(n+s)+b_{s-1}y(n+s-1)+\ldots+b_0 y(n) = a_r x(n+r)+a_{r-1}x(n+r-1)+\ldots+a_0 x(n) \tag{1.2}$$

kde $b_0, b_1, \ldots, b_s$ a $a_0, a_1, \ldots, a_r$ jsou koeficienty diferenční rovnice, $x(n)$ je známá vstupní posloupnost a $y(n)$ je hledané řešení diferenční rovnice

Provedeme-li Z transformaci diferenční rovnice, dostaneme užitečnější přenosovou funkci:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z + a_1 z^2 + \ldots + a_r z^r}{b_0 + b_1 z + b_1 z^2 + \ldots + b_s z^s} \tag{1.3}$$

kde $Y(z)$ je obraz výstupního signálu v Z transformaci, $X(z)$ je obraz vstupního signálu v Z transformaci, $a_0, a_1, \ldots a_r$ a $b_0, b_1, \ldots b_s$ jsou koeficienty diferenční rovnice a $z^r$ a $z^s$ jsou operátory vyjadřující zpoždění systému.

Z přenosové funkce lze zjistit:

- nulové body a póly,
- impulsní charakteristiku,
- stabilitu filtru,
- řád filtru.

**Nulové body a póly**

Polynomy v čitateli $Y(z)$ a jmenovateli $X(z)$ ze vzorce 1.3 můžeme rozložit na součin kořenových činitelů:

$$H(z) = \frac{Y(z)}{X(z)} = K\frac{(z-n_1)(z-n_2)+\ldots+(z-n_M)}{(z-p_1)(z-p_2)+\ldots+(z-p_N)} \tag{1.4}$$

kde $n_{1,2,\ldots,M}$ jsou nulové body a $p_{1,2,\ldots,N}$ jsou póly přenosové funkce.

Nulové body jsou tedy řešením rovnice $Y(z) = 0$ a póly jsou řešením $X(z) = 0$. Zakreslují se do komplexní roviny. Podle počtu pólů se určuje řád filtru.

**Impulsní charakteristika**

Impulsní charakteristika je vynucená odezva systému na jednotkový impuls a vychází ze součinu obrazů diskrétní konvoluce posloupností:

$$y(n) = \sum_{m=0}^{n} x(m)h(n-m) \tag{1.5}$$

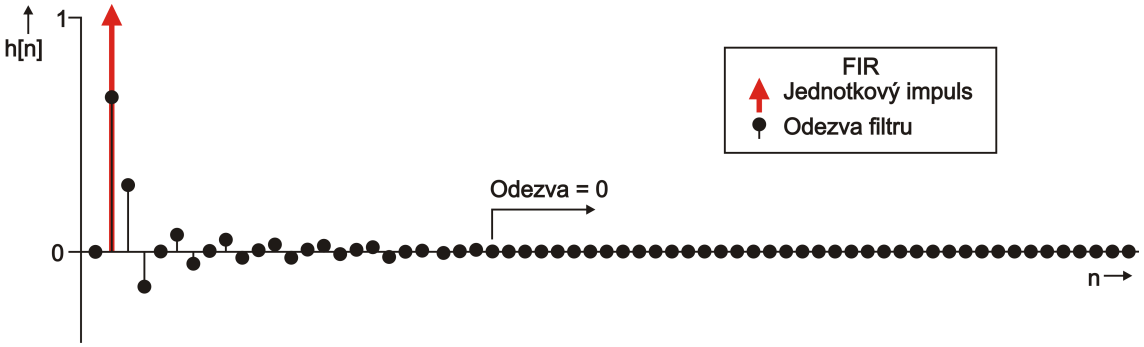
kde $y(n)$ je výstupní signál, $x(n)$ je vstupní signál a $h(n)$ je impulsní odezva (charakteristika) digitálního filtru.

Přenosová funkce $H(z)$ je obrazem impulsní charakteristiky digitálního filtru $h(n)$. Impulsní charakteristika $h(n)$ je dána jako zpětná Z transformace k přenosové funkci $H(z)$.

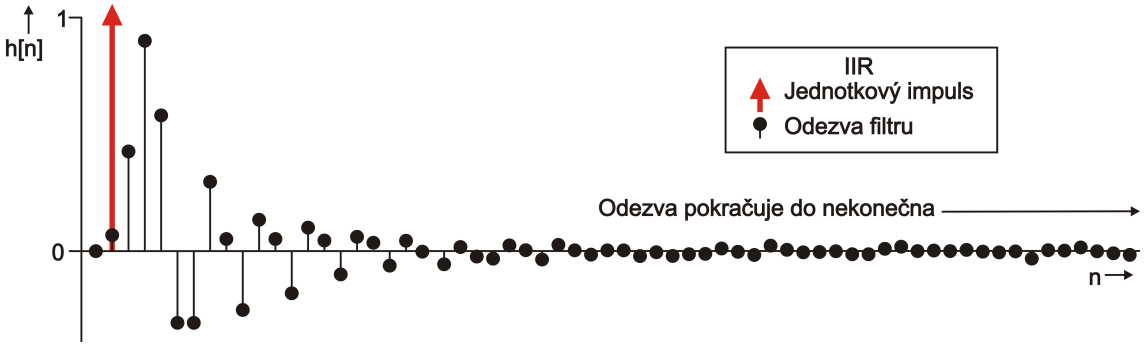
Podle délky impulsní charakteristiky (odezvy) rozdělujeme filtry na:

- diskrétní systémy s konečnou impulsní charakteristikou FIR (Finite Impulse Response),
- diskrétní systémy s nekonečnou impulsní charakteristikou IIR (Infinite Impulse Response).

Na obr. č. 1.1 a 1.2 jsou uvedeny charakteristiky výše zmíněných filtrů:




Obr. 1.1: Impulsní odezva FIR filtru na jednotkový impuls [1]




Obr. 1.2: Impulsní odezva IIR filtru na jednotkový impuls [1]

**Stabilita filtru**

Číslicový filtr je stabilní, pokud všechny jeho póly leží uvnitř jednotkové kružnice v komplexní rovině z.

### 1.1.1 FIR filtr

Filtry s konečnou impulsní charakteristikou (FIR - Finite Impuls Response) jsou nerekurzivními filtry (nemají zpětné vazby), tzn. výstupní signál záleží pouze na hodnotě vstupního signálu.

Diferenční rovnice FIR filtru

$$y[n] = h_0 x[n] + h_1 x[n-1] + \ldots + h_{N-1} x[n-N-1] = \sum_{k=0}^{N-1} h_k x[n-k] \qquad (1.6)$$

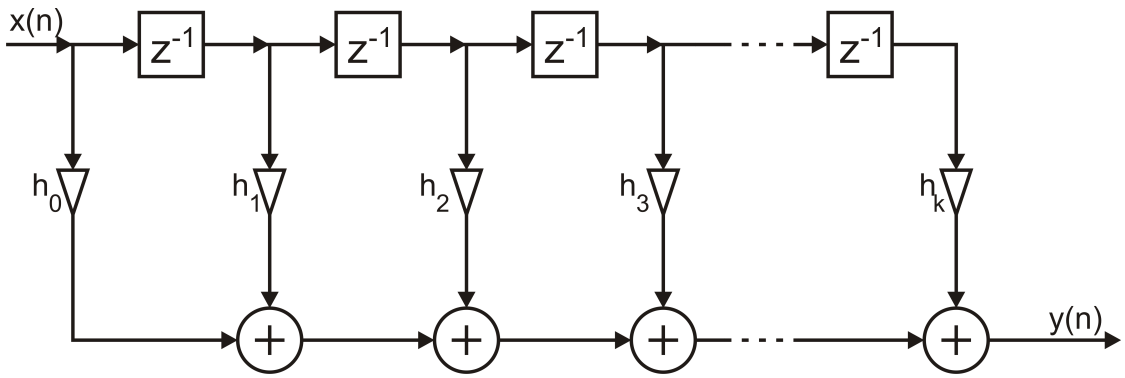
kde $y[n]$ je výstupní signál, $h_0, h_1, \ldots, h_{N-1}$ jsou hodnotami impulsní odezvy, $x[n]$ je vstupní signál, N-1 je řád filtru

Přenosová funkce FIR filtru

$$H(z) = \sum_{k=0}^{N-1} h_k \cdot z^{-k} \qquad (1.7)$$

kde $h_k$ jsou koeficienty filtru. Filtr má jeden N násobný pól $z = 0$, v počátku a systém je tedy vždy stabilní.

Přímá struktura FIR filtru je uvedena na obr. č. 1.3:




Obr. 1.3: Přímá struktura FIR filtru [2]

Výhody:

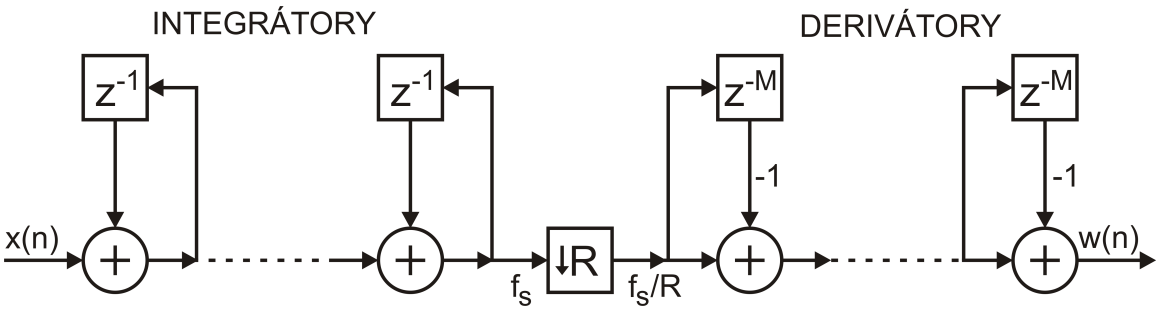
- vždy stabilní,
- matematicky jednoduše popsatelné,
- mohou mít lineární fázovou a kmitočtovou charakteristiku (vhodné pro signály s velkou šířkou pásma).

Nevýhody:

- nemají ekvivalent v analogových filtrech,
- pro dosažení velké strmosti nutno volit vysoký řád filtru,
- nevýhodou je zpoždění filtru při zpracování vstupního vzorku (velký počet zpožďovacích členů, násobiček),
- optimální iterační metody jsou výpočetně náročné.

### 1.1.2 CIC filtr

Speciálním typem FIR filtru je filtr CIC (Cascaded Integrating Comb – Hřebenový integrační filtr). Využívá decimace (snížení vzorkovací frekvence) a interpolace (zvýšení vzorkovací frekvence). Častěji se využívá decimace jako na obr. č. 1.4. Před samotným procesem decimace musí být splněn vzorkovací teorém (vzorec 1.1). Poté je snížen vzorkovací kmitočet decimačním kmitočtem. Decimační faktor označujeme R, vybere se tedy každý R-tý vzorek. Ostatní vzorky se neberou v úvahu. [3]




Obr. 1.4: Struktura filtru CIC - decimátor [4]

Signál x(n) vstupuje navzorkovaný frekvencí $f_s$, signál w(n) vystupuje navzorkovaný frekvencí $f_s/R$.

Příklad použití CIC filtru je veden na obr. č. 1.5:


Obr. 1.5: Blokové schéma filtru s použitím CIC

CIC filtr zpracovává signály s vysokými vzorkovacími frekvencemi a snižuje je. FIR filtr lépe pracuje právě na nižších kmitočtech (má strmější kmitočtovou charakteristiku). Dále kompenzuje útlum CIC filtru.

### 1.1.3 IIR filtr

Filtry s nekonečnou impulsní charakteristikou (IIR - Infinite Impulse Response) jsou rekurzivní filtry (mají zpětné vazby). IIR filtry mají obecně větší možnosti zapojení než FIR právě kvůli rekurzivní části.

Diferenční rovnice IIR filtru:

$$y[n] = \sum_{m=0}^{M} b_m x[n-m] - \sum_{l=1}^{L} a_l y[n-l] \tag{1.8}$$

kde $y[n]$ je výstupní signál, $b_m$ a $a_l$ jsou koeficienty zpětných vazeb, $M$ je počet zpoždění v nerekurzivní části, $L$ je počet zpoždění v rekurzivní části a udává řád filtru, $x[n-m]$ a $y[n-l]$ jsou vzorky vstupního a výstupního signálu zpožděné o m nebo l.

Přenosová funkce IIR filru:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots + a_L z^{-L}} \tag{1.9}$$

IIR filtry mají alespoň jeden pól mimo počátek souřadnic roviny z, proto může dojít k nestabilitě.

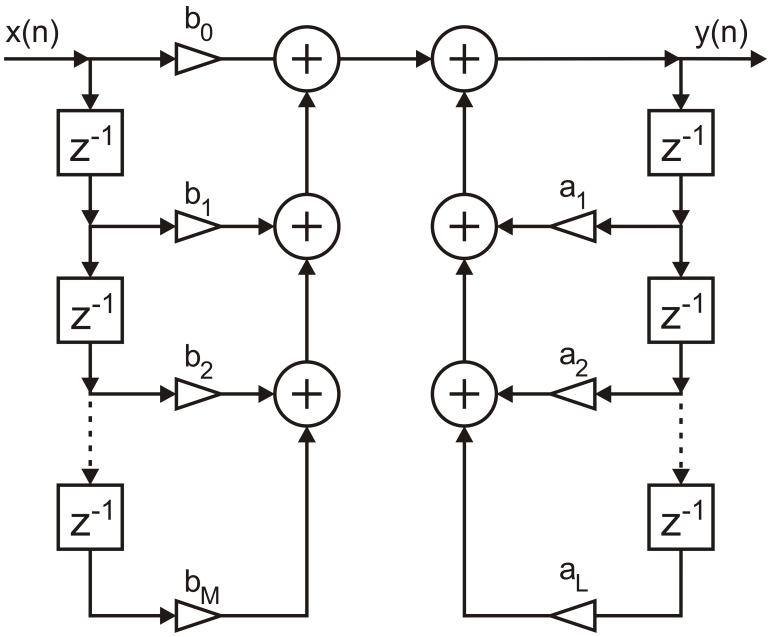
Výhody:

- malý řád přenosové funkce,
- malé zpoždění při zpracování vstupního vzorku (nižší řád filtru -> méně zpožďovacích členů),
- mají ekvivalent v analogových filtrech.

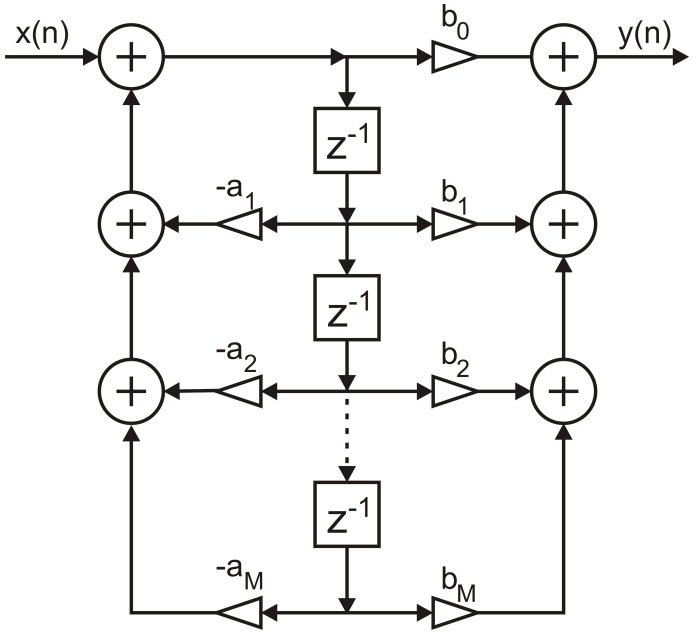
Nevýhody:

- nejsou vždy stabilní,
- nemohou mít lineární fázovou a kmitočtovou charakteristiku,
- velká citlivost na kvantování.

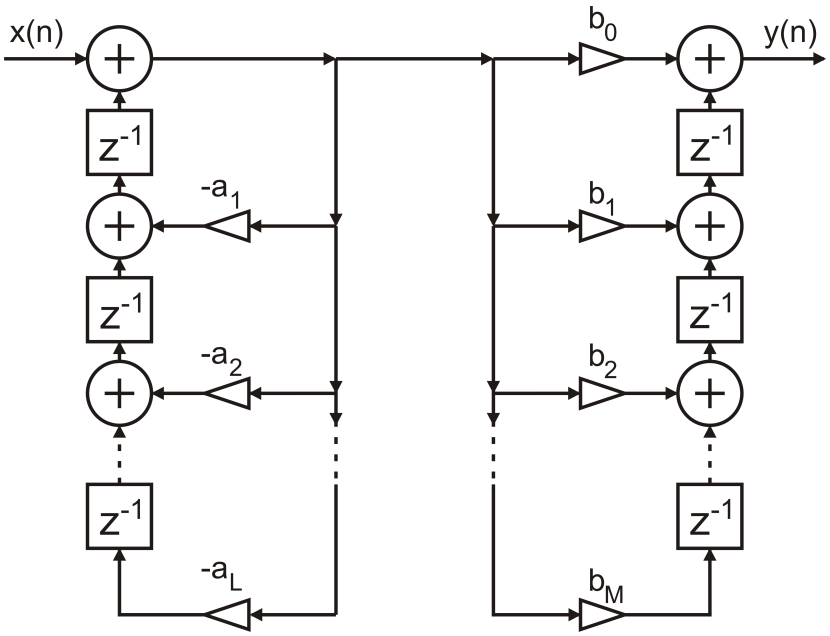
Přímé struktury IIR filtrů jsou uvedeny na obr. č. 1.6 a 1.8. Od nich odvozené (kanonické) struktury jsou uvedeny na obr. č. 1.7 a 1.9.



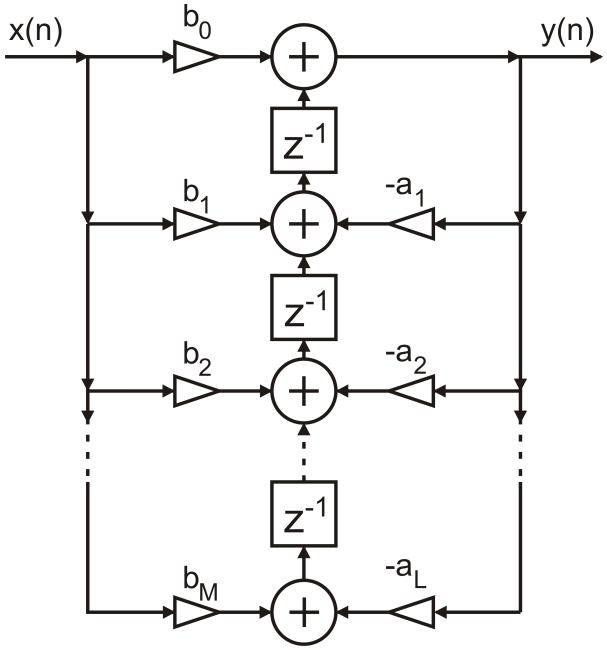

Obr. 1.6: Přímá struktura IIR filtru (Direct Form I) [2]




Obr. 1.7: Kanonická struktura IIR filtru (Direct Form II) [5]




Obr. 1.8: Přímá transponovaná struktura IIR filtru (Transposed Direct Form I) [5]

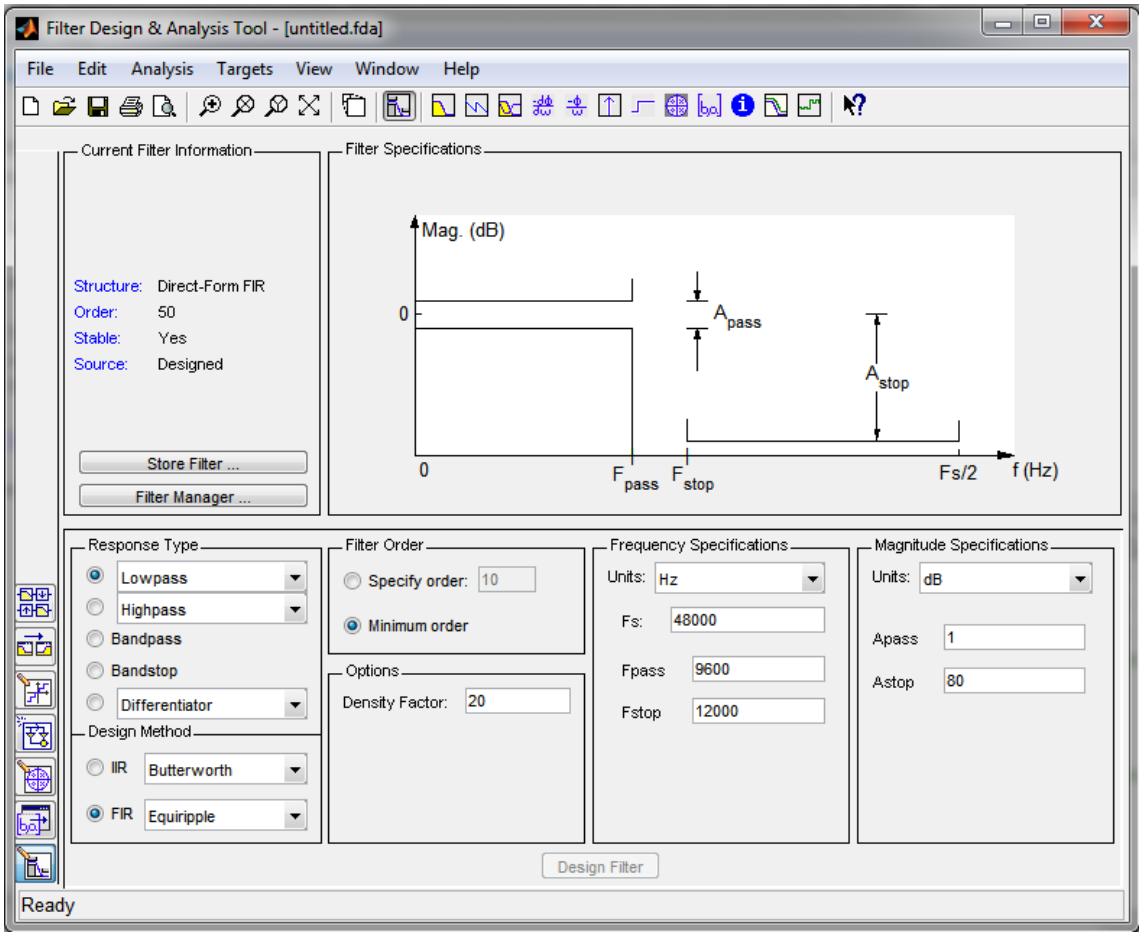



Obr. 1.9: Kanonická transponovaná struktura IIR filtru (Transposed Direct Form II) [5]

Problémy s citlivostí kvantování koeficientů a stability IIR řeší filtry SOS (Second-Order Sections – Filtry druhých řádů). Filtry vyšších řádů jsou nahrazovány kaskádou filtrů druhých řádů jedné z uvedených struktur.

### 1.1.4 Návrh filtrů

Programové prostředí MATLAB nabízí mimo jiné i program Filter Design&Analysis Toolbox (obr. č. 1.10), který slouží pro vytváření digitálních filtrů. V programu MATLAB se spouští příkazem FDATOOL.




Obr. 1.10: Filter Design&Analysis Toolbox - náhled

Možnosti nastavení:

- typ modulové frekvenční charakteristiky (dolní propusť, horní propusť, pásmová propusť, pásmová zádrž, apod.),
- druh filtru z hlediska použité metody – FIR nebo IRR a k nim příslušné aproximace (např. Butterworth, Chebyshev),
- frekvence - Fs - vzorkovací frekvence, Fpass - mezní frekvence, Fstop - frekvence potlačení,
- útlum - Apass - zvlnění v propustním pásmu, Astop - útlum v závěrném pásmu,
- řád filtru - možnost nastavit manuálně nebo nejmenší možný řád filtru,
- možnost nastavení kaskády filtrů,
- kvantovací parametry:

  - výstup koeficientů - fixed point (nastavení pevné desetinné čárky), double-precision floating point, single-precision floating point (formáty s plovoucí desetinnou čárkou),
  - šířka a rozsah koeficientů, vstupních a výstupních dat,
  - možnost nastavení kaskády filtrů,
  - kvantování vstupů, výstupu a koeficientů (může vést k nestabilitě IIR filtrů).

Datový typ s pevnou desetinnou čárkou (fixed point) umožňuje explicitní určení polohy desetinné tečky, tzn. nastavení přesnosti prováděných výpočtů.

Program má mnoho užitečných funkcí:

- zobrazení modulové i fázové frekvenční charakteristiky, skupinového a fázového zpoždění, impulsní a skokové odezvy,
- zobrazení nulových bodů, pólů a koeficientů filtru,
- možnost přidávat nulové body a póly,
- transformaci filtru na jiný typ (např. dolní propusť → horní propusť),
- možnost realizace filtru jako bloku do programu Simulink (další z programů MATLABU),
- export filtru do dalších podpůrných programů MATLABU,
- generování filtru ve VHDL nebo v hlavičkovém souboru v jazyce C,
- výpočet koeficientů filtrů a jejich uložení v desítkové, binární i hexadecimální soustavě,
- umožňuje faktorizovat IIR filtry vysokých řádů na kaskádu IIR filtrů druhého řádu. Důvodem je nestabilita IIR vysokého řádu.

Vytvořený filtr můžeme vyexportovat do souboru s příponou „*.fcf“. V souboru jsou vypočteny koeficienty přenosové funkce ve zvolené číselné soustavě. Dále také informace o typu filtru, bitové šířce koeficientů, rozsahu vstupních a výstupních dat, stabilitě filtru, atd.

## 1.2 XML

Zkratka XML je odvozená od eXtensible Markup Language (rozšířitelný značkovací jazyk). Značkovací jazyky využívají tagů - značek, které přidávají k vlastnímu textu další informace (např. význam a vzhled jednotlivých částí textu). Původně se značky používaly pro korekturu při formátování knih. Značkovací jazyky jsou určené pro tvorbu stránek publikovatelných na internetu, pro poskytování dat mezi aplikacemi a publikování dokumentů, u kterých jsou kladeny nároky na jednoduchou editaci.

Vývoj značkovacích jazyků začal již v roce 1964. XML vychází ze značkovacího jazyka SGML (Standard Generalized Markup Language – Standardizovaný zobecněný značkovací jazyk), který byl však příliš obecný, obsahoval mnoho výjimek a nepravidelností, proto bylo obtížné napsat program, který kód v jazyce SGML zpracuje.

Konsorcium pro World Wide Web se v roce 1996 rozhodlo vytvořit skupinu, která dostala za úkol vytvořit nový standard mezi značkovacími jazyky. Skupina byla později nazvána XWG (XML Working Group – Pracovní skupina pro XML). SGML bylo pracovní skupinou zjednodušeno kvůli své obecnosti. Užitečné prvky SGML ponechala a řadu komplikovaných a nadbytečných věcí vynechala anebo zjednodušila (např. předem definovala). První oficiální verze XML byla vydána 10. února 1998 a jmenovala se XML 1.0 Recommendation. [6]

### 1.2.1 Pravidla XML

Pro správné pochopení syntaxe XML je dobré popsat jednotlivé pojmy a jejich pravidla používání. [7]

**XML deklarace**
Jedná se o první řádek dokumentu. Obsahuje informace o XML verzi a kódování. V následujícím případě se jedná o XML 1.0 a kódování je implicitně nastaveno na ISO 10646, tzn. kódování UTF-8 (prvních 128 znaků je kompatibilních s ASCII).

```
<?xml version="1.0" ?>
```

Pro použití češtiny je možno použít ISO-8859-1 nebo Windows-1250. Potom by zápis deklarace XML dokumentu vypadal takto:

```
<?xml version="1.0" encoding="ISO-8859.1" ?>
```

Stejně jako kódování (encoding) je nepovinné i povolení používání externích souborů (standalone).

**Element**
Každý element se skládá z počátečního a koncového tagu (značky), mezi kterými se nachází text:

```
<jmeno>Eva</jmeno>
```

Deklarace XML dokumentu není element, tudíž nemusí mít koncovou značku.
V případě, že mezi tagy není text, jsou možné tyto dva zápisy:

```
<prodej datum="1.1.2000"></prodej>
<prodej datum="1.1.2000"/>
```

Takový element je nazývaný jako prázdný. Jeho používání by nemělo smysl, pokud bychom do něj nezapsali nějakou informaci - atribut (viz níže). Jednotlivé elementy se nesmí křížit. U značek se rozlišují malá a velká písmena.

**Kořenový element**
V XML dokumentu je to nejvyšší element, který může být jen jeden a nazývá se „root element“. Dá se říct, že „obaluje“ celý dokument.

**Vztahy mezi elementy**

```
<otec>
<jmeno>Vladislav</jmeno>
    <sestra>Anna</sestra>
    <bratr>Jan</bratr>
</otec>
```

Otec je kořenový element. Bratr a sestra jsou děti otce a navzájem jsou sourozenci.

**Atribut**
Skládá se z názvu a hodnoty atributu. Hodnoty atributů musí být v uvozovkách nebo v apostrofech. Častěji se však používají uvozovky. Použití je možné v klasickém elementu, tak i v prázdném elementu.

**Instrukce**
Slouží pro začlenění nestandartních dat. Např. připojení CSS stylu k dokumentu, ruční zlom řádku nebo stránky, apod. Intrukce používá znaky „<?“ a „?>“.

**Komentář**
Pro zakomentování části kódu nebo pro samotný komentář se používá této syntaxe:

```
<!-- Toto je komentář -->
```

Díky velké volnosti při tvorbě vlastních značek v XML si každý tvůrce může pojmenovat svou značku podle svého uvážení (např. <cena>, <castka>, <hodnota>, apod.). Dokumenty jsou sice validní, ale nastává problém s vyhledáváním. Proto začaly být schémata standardizovány. Existují schémata například pro matematické vzorce, elektronické obchody, technické dokumentace, chemické vzorce. [8]

### 1.2.2 Výhody a nevýhody XML

**Výhody**

- XML je standardizován,
- platformová nezávislost,
- čitelnost člověkem i strojem,
- podpora Unicode (asi 110 000 znaků),
- reprezentace nejběžnějších datových struktur (záznam, seznam, strom, pole),
- pevná pravidla pro syntaxi umožňují jednoduché a efektivní zpracování.

**Nevýhody**

- XML soubory bývají velké,
- náročnost na výpočetní techniku při zpracovávání větších XML dokumentů,
- není definováno omezení na počet vnoření,
- jiné datové typy než text je nutno definovat pomocí schémat,
- s XML lze manipulovat jen jako s celkem. [9]

## 1.3 Návrh grafického prostředí

### 1.3.1 Qt4 Framework

Qt prošlo za poslední dobu mnoho majiteli (Trolltech, Nokia, Digia) a tím se měnil i samotný název, ve kterém však „Qt“ vždy zůstalo. Qt je komplexní C++ framework pro vývoj multi-platformních aplikací. Zastává přístupu „napsat jednou, sestavit (zkompilovat) kdekoliv“. Qt umožňuje programátorům používat jediný zdrojový kód pro aplikace, které poběží na Windows, Mac OS X, Linux, Solaris, HP-UX a mnoho dalších verzí Unixu s X11. Qt knihovny a nástroje jsou také součástí Qt Utopia Core, produktu, který poskytuje vlastní systém oken nad platformou Embedded Linux. Podporuje SQL, zpracování XML, přístup k souborům, správu vláken, práci s grafikou (i 3D) a multimédii. Velkou výhodou Qt je rozsáhlá dokumentace. [10][11]

### Qt Creator

Qt Creator je kompletní integrované vývojové prostředí (IDE) pro vytváření aplikací Qt. Qt Creator je určen pro vývoj desktopových i mobilních aplikací. Je k dispozici jako samostatný balíček nebo v kombinaci s knihovnami Qt a vývojovými nástroji jako kompletní SDK. Qt Creator obsahuje editor kódu pro C++ a JavaScript, prostředí pro návrh grafického uživatelského rozhraní (Qt Designer), simulátor pro uživatelská rozhraní mobilních aplikací (Qt Simulator umožňuje rychlé testování a ladění aplikací, které se zaměřují na mobilní zařízení) a další. [11][12]

### Qt Designer

Qt Designer je nadstavbou pro Qt Creator. Aplikace mohou být napsané jen pomocí Qt Creatoru, tzn. i grafická podoba aplikace je popsána zdrojovým kódem. Qt Designer zjednodušuje proces navrhování a vytváření grafických uživatelských rozhraní (GUI), je intuitivní a k dispozici má rozsáhlé nápovědy i s tutoriály. Snižuje časovou náročnost při sestavování projektů. Qt Designer pracuje hlavně s tzv. widgety, což jsou prakticky ovládací prvky pro uživatele. Umožňuje jejich vkládání, rozmístění, tvoření skupin, tříd. Dá se říci, že v tomto programu je možnost nastavit velkou část vlastností týkající se zobrazení, které uvidí uživatel. Nastavuje ovládací prvky do určitých počátečních stavů a u některých umožňuje naplnit je obsahem přímo a ne v kódu (např. Label, Combo Box, Menu). Ze zobrazení hierarchie je přehledně vidět vztah podřízených a nadřazených ovládacích prvků. [11][12]

### Qt Assistant

S Qt Assistant se pracuje podobným způsobem jako ve webovém prohlížeči. Dokumentace obsahuje hypertextové odkazy, čímž usnadňuje vyhledávání a orientaci. Obsahuje kompletní dokumentaci celé knihovny Qt s řadou příkladů a ukázek. Jelikož využívá pokročilého fulltextového vyhledávání, lze hledat konkrétní slova i fráze. Propojení této dokumentace s Qt Designer a samotným kódem je pomocí funkční klávesy „F1“, kdy stačí pouze najet myší na daný objekt nebo slovo, stisknout klávesu „F1“ a Qt Assistant vyhledá dotazovanou informaci. [11][12]

### Qt Linguist

Qt Linguist je nástroj pro překládání aplikací do různých jazyků. Text, který chceme přeložit do jiných jazyků, označíme jednoduchou syntaxí. Poté stačí vygenerovat soubor s textem, který byl takto označen, přeložit ho a nahrát zpět. [11][12]

# 2 NÁVRH PROGRAMU

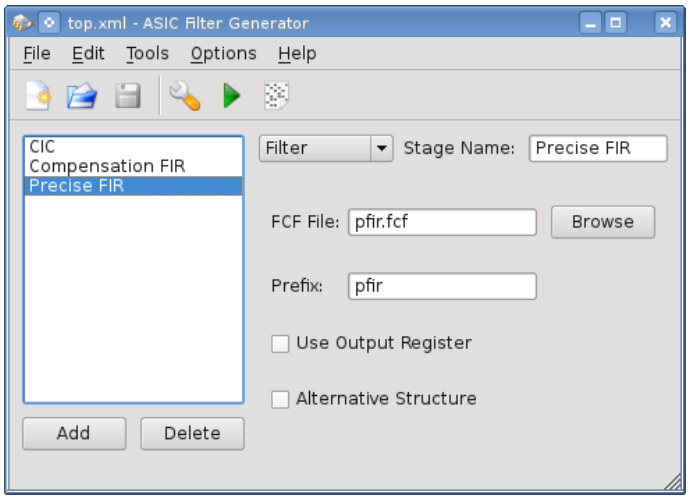
Cílem této práce je navázat na program (obr. č. 2.1), který umožňoval generovat popis kaskádního zapojení filtrů v jazyce VHDL (Very High Speed Integrated Circuits Hardware Description Language – Jazyk pro popis velmi rychlých integrovaných obvodů).
Výhody:

- vhodné pro jednoduchá kaskádní zapojení filtrů,
- jednoduché GUI,
- generátor VHDL.

Nevýhody:

- neumožňuje návrh složitějších zapojení filtrů,
- neumožňuje připojení dalších bloků (SPI, UART apod.).

Úkolem této práce je odstranit výše zmíněné nevýhody. Pro sériové i paralelní zapojení filtrů bylo nutno vytvořit program, kde je umístěna plocha pro návrh. Na této ploše může uživatel jednoduše rozmisťovat a propojovat filtry a podpůrné obvody.



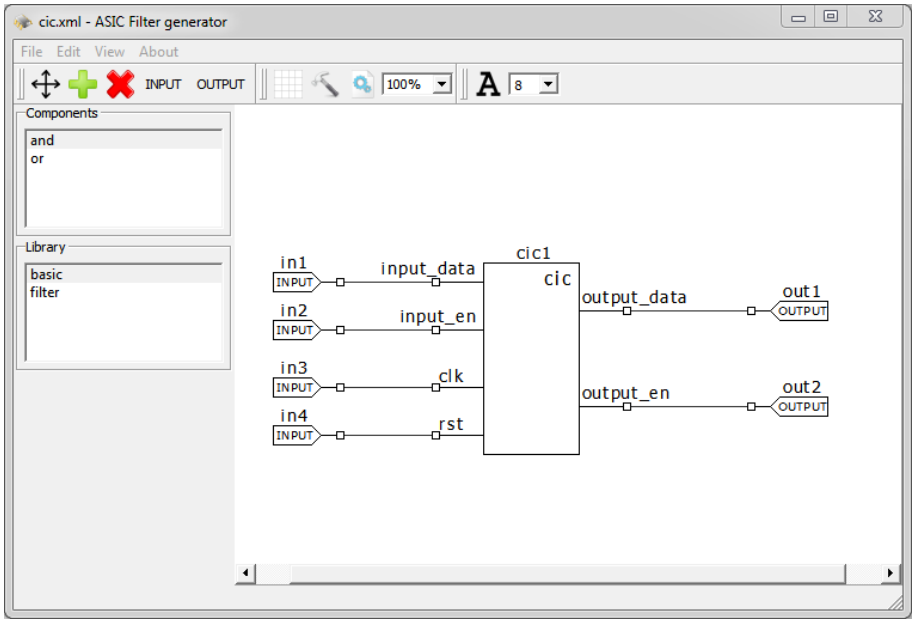

Obr. 2.1: Grafická podoba původního programu

## 2.1 Návrh grafické podoby programu

Program se skládá z několika dialogových oken. Hlavní okno s názvem *MainWindow* je potomkem třídy *QMainWindow*, která je určena k vykreslování hlavního okna aplikace. Tato třída umožňuje vložit menu, nástrojové lišty (toolbars), stavové lišty, ovládací prvky (widgets) apod. Další okna jsou potomky třídy *QWidget*, která umožňuje tvořit vlastní uživatelské dialogy.

### 2.1.1 Hlavní okno

GUI bylo na začátku navrhováno pouze v Qt Designeru pro lepší pochopení vlastností používaných objektů. Často bylo obtížné efektivně editovat návrh (jednalo se pouze o zdrojový kód), proto bylo přistoupeno k návrhu s pomocí nástroje Qt Creator a tím se podařilo zrychlit samotný proces návrhu (kompilace programu, jednodušší návrh grafiky). Některé ovládací prvky (např. ComboBox) nebylo možné umístit do panelů nástrojů, proto byly navrhnuty v Qt Designeru. GUI formulář je uložen v souboru mainwindow.ui. Jedná se o XML soubor.




Obr. 2.2: Grafická podoba hlavního okna

#### Plocha pro návrh

Hlavním a největším ovládacím prvkem je kreslící plátno (třída *QGraphicsScene*). Na plátno je možné vkládat součástky (Components) z knihoven (Library), vstupy, výstupy, propojovat je vodiči a vkládat text pro komentář k zapojení. Všechny

objekty vložené na plátno kromě textu se pohybují po mřížce. Je-li součástka přemístěna mimo kreslící plátno, je plátno zvětšeno. Díky tomu se nikdy nestane, že by se součástka dostala mimo kreslící plátno.

#### Hlavní nabídka - Menu

Menu obsahuje klasické uživatelské funkce (Nový, Otevřít, Uložit, Zavřít, Exportovat, Ukončit) a funkce nastavení a generování VHDL. V dalších záložkách uživatel nalezne možnosti vypnout a zapnout jednotlivé panely nástrojů a viditelnost mřížky, možnost nastavení pro uložení do XML souboru.

#### Panely nástrojů

K dispozici jsou tři panely nástrojů. V prvním se nachází funkce pro pohyb, vkládání a mazání součástek (vstupů, výstupů nebo z knihoven). Ve druhém panelu nástrojů je možnost zapnout viditelnost mřížky a vyvolat dialogové okno pro nastavení aktuálního projektu. Třetí panel nástrojů je určen pro jednoduchou práci s textem (vložení textu a editace velikosti). Funkce pro změnu fontu, tučný text, podtržení a kurzívu nebyly implementovány. Důvodem byla nutnost ukládat o každém textu všechny potřebné informace do XML souboru, proto byly tyto funkce vynechány. Uživatel může panely nástrojů přesouvat podle svého uvážení.

#### Součástky (Components)

V ovládacím prvku třídy *QListWidget* se zobrazuje seznam jednotlivých součástek z aktuálně načtené knihovny. Označenou součástku je možno vložit pomocí tlačítka a nebo dvojklikem na označenou součástku.

#### Knihovny (Library)

Při spuštění se načtou knihovny, se kterými se bude pracovat. Uživatel si vybere knihovnu a z ní se mu poté načtou součástky do okna „Components“.

#### Panel nápovědy

Pro jednoduchou nápovědu v programu slouží stavový řádek umístěný v dolní části okna. Nápověda se mění podle polohy kurzoru myši.

### 2.1.2 Další dialogová okna

Funkce podpůrných dialogových oken bude popsána v dalších podkapitolách. Další používaná dialogová okna jsou standardní vestavěné dialogy Qt. V programu jsou používány dialogy třídy *QFileDialog* pro ukládání a načítání souborů, vybírání

adresářů a dialogy třídy *QMessageBox* typu informace, dotaz, varování, o programu a o Qt Framework.

## 2.2 Popis funkce programu

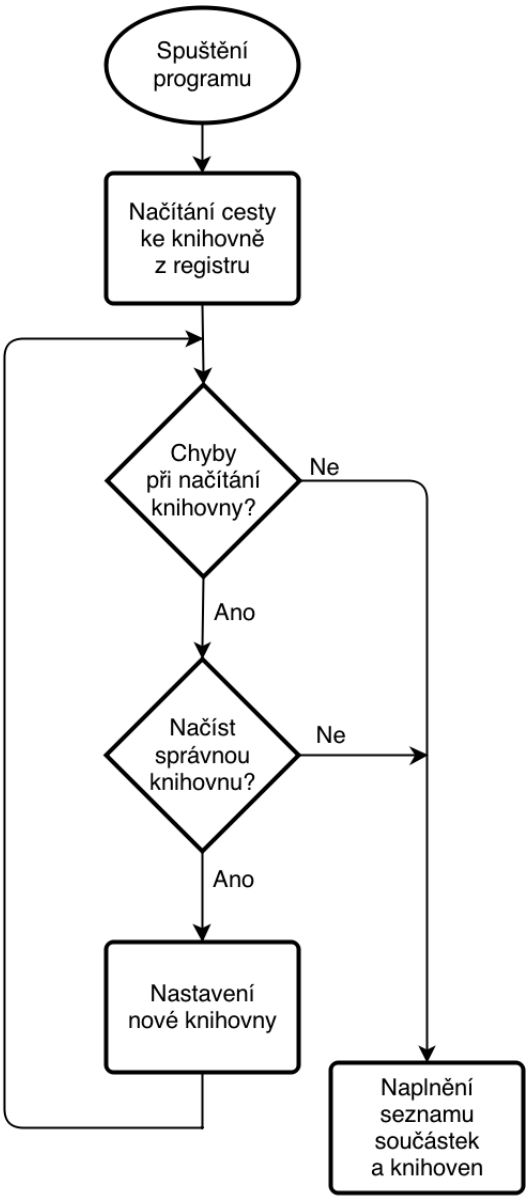
V následující kapitole je popsána funkce programu s ohledem na jednotlivé zdrojové soubory. V tabulce 2.1 jsou uvedeny použité soubory:

Tab. 2.1: Výčet zdrojových souborů

| Umístění | Název souboru | Popis |
|---|---|---|
| / | connection (*.h, *.cpp) | vodiče |
| / | connection_action (*.h, *.cpp) | práce s vodiči |
| / | defines (*.h) | definice hodin a resetu |
| / | fg01 (*.pro, *.qrc *.user) | hlavní soubor projektu v Qt |
| / | library (*.xml) | knihovna se součástkami |
| / | main (*.cpp) | hlavní funkce programu |
| / | mainwindow (*.h, *.cpp, *.ui) | hlavní okno |
| / | scene (*.h, *.cpp) | kreslící plátno |
| / | textitem (*.h, *.cpp) | práce s textem |
| /base/ | lib_structure (*.h, *.cpp) | čtení z knihovny |
| /base/ | top_structure (*.h, *.cpp) | zápis a čtení z XML |
| /base/ | function (*.h, *.cpp) | vytvořené funkce |
| /base/ | reports (*.h, *.cpp) | zprávy o načítání XML souborů |
| /base/tinyxml/ | tinyxml (*.h, *.cpp)<br>tinystr (*.h, *.cpp)<br>tinyxmlerror (*.cpp)<br>tinyxmlparser (*.cpp) | soubory knihovny TinyXML |
| /images/ | obrázky ve formátu PNG | ikony panelů nástrojů |
| /other_forms/ | mainsettingsdialog (*.h, *.cpp, *.ui)<br>newfiledialog (*.h, *.cpp, *.ui)<br>projectsettingsdialog (*.h, *.cpp, *.ui)<br>symbolsettingsdialog (*.h, *.cpp, *.ui) | ostatní formulářová okna |
| /symbol/ | symbol_main (*.h, *.cpp)<br>symbol_port (*.h, *.cpp) | vytvoření součástky |

### 2.2.1 Hlavní funkce programu

Funkce *main()* se nachází v souboru main.cpp a je volána automaticky po spuštění programu. V této funkci je vytvořen objekt typu *QApplication* - spravuje prostředky celé aplikace a je vyžadován pro funkčnost všech Qt programů mající GUI. Při první spuštění je uživatel vyzván, aby zvolil cestu ke knihovně a cestu ke generátoru VHDL. Na obr. č. 2.3 je zobrazen postup při načítání knihovny. Pokud je knihovna správně načtena, naplní se seznamy knihoven a součástek. Při dalším spuštění jsou cesty ke knihovně a generátoru načteny automaticky. Čtení z knihovny zajišťuje třída *LibStructure*.




Obr. 2.3: Funkce načítání knihovny

### 2.2.2 Hlavní okno - třída MainWindow

O propojení grafická podoby (obr. č. 2.2) hlavního okna, o celou funkčnost programu a komunikaci s ostatními třídami se stará třída *MainWindow*, která je popsána v souborech mainwindow.h a mainwindow.cpp.

#### Konstruktor třídy MainWindow

V konstruktoru třídy *MainWindow* je:

- propojeno uživatelské rozhraní (UI - user interface, obr. č. 2.2) s *MainWindow*,
- vytvořeny ukazatele na třídy kreslícího plátna a akce vodičů,
- propojení kreslícího plátna (třída *Scene*) se třídou *QGraphicsView*, která plátno zobrazuje,
- načtena cesta ke knihovně (viz Ukládání nastavení do systémových registrů),
- propojení akcí z GUI k jednotlivým funkcím,
- vytvoření panelů nástrojů - doplnění některých funkcí vytvářených mimo Qt Creator,
- nastavení výchozích parametrů pro ukládání zapojení do XML souboru (pracovní adresář, události hodinového signálu a resetu, popis zapojení),
- nastavena viditelnost a povolení některých funkcí (při spuštění jsou plátno a seznamy neviditelné, panely nástrojů a některé vybrané funkce z menu jsou zakázány),
- propojení signálů s funkcemi.

#### Menu

V menu „Soubor“ („File“) se nachází tyto funkce:

- Nový (New)- funkce pro vytvoření nového souboru. Zobrazí se dialogové okno pro vložení názvu souboru (*NewfileDialog*). V této funkci jsou vymazány případné vložené grafické objekty a jsou nastaveny výchozí parametry pro ukládání do XML souboru. Po vložení jména souboru jsou kreslící plátno, seznamy knihoven a součástek a všechny zakázané funkce povoleny.
- Otevřít (Open) - funkce pro načtení uloženého zapojení z XML souboru. Otevře se dialogové okno pro vybrání souboru, který chce uživatel načíst. Čtení z XML souboru obstarává třída *TopStructure* (viz 2.2.12). Nejdříve se načítají parametry z elementu „settings“. Následuje načítání vstupů, výstupů, součástek, vodičů, textů a čítačů.
- Uložit (Save), Uložit jako (Save As) - funkce pro ukládání vytvořených zapojení. Ukládání probíhá opět pomocí třídy *TopStructure*. Postupně se ukládá nastavení parametrů (pracovní adresář, hodiny, reset atd.), vstupy a výstupy, součástky, vodiče a texty. Výchozí adresář pro ukládání zapojení

je adresář, ze kterého je program spuštěn. Pokud je soubor uložen v jiném než pracovním adresáři, změní se tento adresář na pracovní.
- Zavřít (Close) - funkce pro zavření aktuálního souboru. Vrací celý program do počátečního stavu, tzn. jsou nastaveny výchozí parametry projektu, vymazány objekty z kreslícího plátna, upraveno zobrazení a povolení některých funkcí.
- Generovat (Generate) - funkce pro generování VHDL popisu z XML souboru. Funkce spustí program pro generování VHDL a předá mu potřebné parametry. Uživatel je upozorněn, zda vytváření VHDL popisu proběhlo v pořádku nebo nastala nějaká chyba. Chyby jsou zobrazovány v dialogovém okně v podrobnostech.
- Nastavení projektu (Project Settings) - funkce pro vyvolání nastavení aktuálního projektu.
- Hlavní nastavení (Main Settings) - funkce pro vyvolání hlavního nastavení programu.
- Export - funkce pro export celého kreslícího plátna. Exportovat lze do formátů BMP (Bitmap image file – Formát pro ukládaní rastrové grafiky), JPEG (Joint Photographic Experts Group – Formát pro ztrátovou kompresi grafiky) a PNG (Portable Network Graphics – Přenosný formát pro bezeztrátovou kompresi rastrové grafiky). Výchozím adresářem pro export zapojení je pracovní adresář.
- Export náhledu (Export View) - funkce pro export náhledu kreslícího plátna. Opět lze exportovat do BMP, JPEG a PNG.
- Ukončit (Exit) - funkce pro vypínání aplikace.

V menu „Edit“ se nachází funkce pro:
- pohyb s vloženými objekty,
- pro mazání,
- funkce pro vyvolání nastavení součástky.

V menu „Tools“ lze měnit:
- zobrazení panelů nástrojů,
- zobrazení mřížky.

V menu „About“ se nachází informace o programu a o Qt.

### Funkce pro chod programu

Výzvy k uložení mohou nastat při ukončení programu, vytváření nového projektu atd. V těchto a dalších případech se zavolá funkce *readyToContinue()*, která zjistí, zda byla provedena nějaká změna před posledním uložení. Změna je zjišťována pomocí otisku (hash) dvou souborů. Prvním je uložený soubor a druhým

je dočasný soubor, do kterého je uloženo aktuální zapojení. Pokud nebyla provedena změna, otisky se rovnají a není potřeba vyzývat uživatele k uložení.

Zobrazení panelů nástrojů (funkce *visibleToolbars()*) a změny zobrazení a povolení některých funkcí (*visibleWorkSpace()*) jsou jednoduché funkce, které mění zobrazení nebo povolují některé funkce.

Pro unikátní názvy vstupů, výstupů, součástek a vodičů je v názvu zavedena pořadová číslice. Výchozí názvy pro vstupy jsou „in“, pro výstupy „out“, pro součástky „unit“ a pro vodiče „net“. Funkce *findFreeNumber()* najde podle zadaného klíčového slova nejmenší volnou číslici, která se přiřadí k názvu součástky. Např. jsou vložené součástky s názvy „unit1“, „unit2“ a „unit3“. Prostřední součástka s názvem „unit2“ je smazána. Při vkládání další součástky se zavolá zmíněná funkce a předá se jí název „unit“. Podle předaného slova si načte názvy všech součástek. Pokud název začíná předaným slovem, načte číslo za tímto slovem. Všechna čísla názvů se uloží do pole, seřadí se a najde se nejmenší volné. V tomto případě funkce vrátí hodnotu „2“.

Uživatel má možnost zobrazit si mřížku, po které se pohybují součástky a vodiče. Pohyb po mřížce však nelze vypnout, aby se zamezilo chybám při připojování vodičů k součástkám. Funkce zobrazení mřížky *gridVisible()*) vytvoří obrázek 10x10 pixelů. Dvě jeho strany jsou vybarveny barvou mřížky, zbytek je pixelů je bílý. Pomocí funkce *setBackgroudBrush()* se obrázkem vyplní celé pozadí kreslícího plátna.

Funkcí usnadňující vytváření rozsáhlejších zapojení je lupa. Velikost kreslící plochy lze měnit v rozsahu 75-150%. Třída *QGraphicsScene* (rodič kreslícího plátna) má v souřadnicovém systému osu Y naopak než je obvyklé (záporné hodnoty jsou nad osou X, kladné pod osou X). Pro jednoduché vytváření součástek v knihovnách a nastavení obvyklého souřadnicového systému bylo nutné plátno, na které zobrazujeme kreslící plátno (třída *QGraphicsView*), zrcadlit podle osy X. Tato změna se však týká všech grafických objektů a exportu zapojení jako obrázku, které musí být také zobrazovány zrcadlově.

### Vkládání textu

Při stisknutí tlačítka pro vložení textu je změněn mód kreslícího plátna (třída *Scene*) na vložení textu. O samotné vložení se stará třída kreslícího plátna. Jakmile je text vložen, je mód kreslícího plátna změněn na pohyb s vloženými objekty. Dále je možno měnit velikost textu.

### Vkládání součástek

Stejně jako u vkládání textu se i o vkládání součástek stará třída *Scene*. Je nutné předat třídě *Scene* několik parametrů, které přesně definují, o jaký symbol se jedná. Předává se název knihovny, název součástky z knihovny (např. „fir“)

a jméno součástky (např. „unit1“). Jméno součástky je možné měnit. Třídě *Scene* se přepne mód na vkládání součástek. Kliknutí na kreslící plochu je součástka vložena. Mód třídy *Scene* se nastaví na pohyb s vloženými součástkami.

#### Vkládání vstupů a výstupů

Vstupy a výstupy se vkládají podobně jako součástky z knihovny. Třídě *Scene* se předá název knihovny „ports“ a název vstupu „input“ nebo výstupu „output“. Opět je vložen název ke konkrétnímu vstupu (např. „in1“), reps. výstupu (např. „out1“). Vstupy a výstupy nejsou uloženy v knihovnách a program je nenačítá, ale vytváří.

#### Pohyb, mazání vložených objektů

Všechny vkládané grafické objekty mimo text se pohybují po mřížce. Pohyb zajišťuje další mód třídy *Scene*. Důležitou funkcí je mazání. Při mazání se prohledají všechny vložené objekty a ty, které se jsou označené, jsou vymazány. Každý port součástky, ke kterému je připojen vodič, ukládá název tohoto vodiče. Při mazání vodiče je nutno vymazat i tento název.

#### Ukládání nastavení do systémových registrů

Při ukončení programu jsou cesty ke knihovně a ke generátoru VHDL uloženy do systémového registru a při opětovném spuštění jsou načteny. Uživatel nemusí knihovnu a generátor zadávat při každém spuštění programu.

### 2.2.3 Grafická scéna - třída Scene

Třída *Scene* (potomek třídy *QGraphicsScene*) slouží pro vytvoření obrazu, který je poté zobrazen na zobrazovacím plátně třídy *QGraphicsView*. Zdrojový kód k této třídě je v souborech scene.h a scene.cpp. Pro jednodušší funkci třídy *Scene* byly vytvořeny tři módy: pohyb, vložení součástky a vložení textu. V konstruktoru třídy je nastaveno bílé pozadí, kontextové menu, velikost mřížky a výchozí mód třídy *Scene* je pohyb.

Při vkládání součástky se po kliknutí na kreslící plochu (mód „vkládání součástky“) vytvoří ukazatel na součástku třídy *SymbolMain*, kterému jsou předány pomocí funkcí všechny potřebné parametry:

- cesta ke knihovně, ze které byla součástka vložena,
- název knihovny,
- název součástky,
- pozice pro vložení (pozice, kam uživatel klikl je zaokrouhlena),
- kontextové menu, která má být použito na tuto součástku,

- editovatelné jméno součástky.

Třída *SymbolMain* vrátí vytvořenou součástku, která je vložena na kreslící plátno. Po vložení součástky se odešle signál zpět hlavní třídě *MainWindow*, která přepne mód scény na „pohyb“.

Při vkládání textu je nastaven mód „vkládání textu“ a je vytvořen ukazatel na třídu *TextItem*. Předává se jí pouze písmo, které obsahuje informaci o velikosti písma. Zbytek vlastností písma není možno měnit. Třída *QGraphicsTextItem*, která je rodičem třídy *TextItem*, má mnoho různých nastavení. Pro možnost editování textu stačí pouze změnit jeden parametr při vytváření textu.

Při načítání součástky nebo textu z uloženého souboru se postupuje podobně. Pozice se neodvozuje od kliknutí, ale je načítána ze souboru.

### 2.2.4 Vložení textu - třída TextItem

Třída *TextItem* je popsána v souborech textitem.h a textitem.cpp. v konstruktoru třídy jsou nastaveny vlastnosti pro možnost vybrání a pohybu textu. Text je obrácen podle osy X, aby se zobrazil správně (viz. 2.2.2 - odstavec Funkce pro chod programu). Ve třídě jsou funkce reagující na události textu - označení textu, konec upravování textu a dvojité kliknutí na text. Každá událost vhodně upravuje vlastnosti a vysílá signály např. když je text vybrán, aby bylo možné změnit jeho velikost.

### 2.2.5 Vytváření součástek

Při vytváření součástky se z knihovny načítají data o obrysu součástky, názvu a o portech (název, pozice, směr - vstup, výstup). O vytvoření součástky se starají tyto třídy:

#### Řídící třída - třída SymbolMain

V souborech symbol_main.h a symbol_main.cpp je popsána funkce třídy *SymbolMain*. V konstruktoru se nastavena velikost mřížky, vytvořeny ukazatele na název a editovatelné jméno, ukazatel na třídu *LibStructure*, ze které jsou načítány součástky. Grafickému objektu třídy *SymbolMain* je povolen pohyb a možnost vybrání objektu. Vytváření součástky se dá rozdělit do těchto kroků:

- z třídy *Scene* jsou poslány název, knihovna, jméno, pozice a další vlastnosti podle konkrétní součástky,
- podle názvu součástky se najde pozice v dané knihovně (tato pozice se používá v celé třídě *SymbolMain*),
- vytvoření obrysu součástky (obrys je v jedné vrstvě),
- vytvoření dalších čar,

- vytvoření názvu součástky a dalších textů,
- načtení parametrů součástek z elementu <param>,
- vytvoření portů - postupně jsou vytvořeny všechny porty součástky (podrobněji popsáno níže),
- vytvoření editovatelného jména.

Součástky typu vstup a výstup jsou přiřazeny ke knihovně „ports“, která však prakticky neexistuje. Jedná se pouze o odlišení, že se mají součástky vytvořit bez načítání z knihovny. Postup vytváření součástky se však příliš neliší. První se vytvoří obrys, název, jeden port a poté jméno součástky.

Události myši řídí pohyb po mřížce a vybírání součástky. Při pohybu myši jsou přepočítávány souřadnice tak, aby se součástka pohybovala po mřížce. Nejdříve se načte aktuální pozice součástky, která je vydělena velikosti mřížky. Tato hodnota je zaokrouhlena a zpětně vynásobena velikostí mřížky. Poté je součástka posunuta na pozici ležící na mřížce.

#### Porty - třída SymbolPort

Třídě *SymbolPort* je předán název knihovny, jméno portu, směr (vstup, výstup), pozice a ukazatel na součástku. Port je umístěn na pozici určenou v knihovně. Podle směru (vstup, výstup) se umístí název portu tak, aby nezasahoval do obrysu součástky. Dále třída obsahuje celou řadu metod vracejících hodnoty (tzv. getter) a metod pro zápis hodnot (tzv. setter). Třída je popsána v souborech symbol_port.h a symbol_port.cpp.

### 2.2.6 Práce s vodiči

Práci s vodiči třídy *Connection* obstarává třída *ConnectionAction*.

#### Vodiče - třída Connection

Třída pro vytváření vodičů se nachází ve zdrojových souborech connection.h a connection.cpp. V této třídě jsou vytvořeny ukazatele na porty. Dále proměnné pro pozice portů, jméno vodiče a dvě pole bodů pro práci s vloženou sítí vodičů. Při kreslení vodiče se načte počáteční port součástky a pozice, která se zaokrouhluje stejně jako u pohybu součástek. Každý další vložený bod na plátno (mimo port součástky) je ukládán do pole bodů. Podle počtu vložených bodů se liší algoritmus pro vykreslování vodičů. Pro ukončení vodiče je nutné kliknout na další port součástky. Tyto porty a pozice portů se ukládají, aby mohly být při přesunu součástky přepočítány a vodiče stále zůstávaly připojené k portům součástek.

#### Akce vodičů - třída ConnectionAction

Třída *ConnectionAction* je potomkem třídy *QObject*, které je vhodná pro práci se signály. Třída pomocí filtru událostí dostává všechny signály na události třídy *Connection* dříve než samotná třída *Connection*. Třída používá funkci *itemAt()* pro zjištění, zda se na pozici kliknutí nachází grafický objekt. Pokud se tam nachází port, začne vykreslování vodiče:

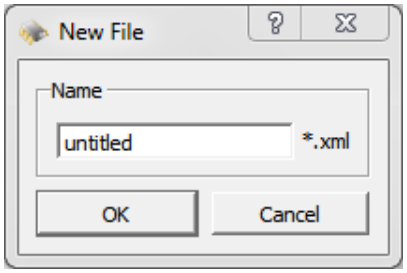
- vytvoří se vodič třídy *Connection*,
- načte se ukazatel na první port a pozice na plátně,
- při posunu myši se aktualizuje druhá pozice vodiče,
- při kliknutí se vytvoří bod zalomení,
- při připojení vodiče na druhý (koncový) port je načten ukazatel a pozice tohoto portu,
- dále je přiřazen název k vodiči a k připojeným portům.

Maximální počet zalomení je 50. Při větším počtu zalomení začíná být vykreslování vodiče pomalé. Při vložení 40 zalomení je uživatel varován, že pokud vloží ještě více vodičů, bude vkládaný vodič smazán.

Vytváření uzlů je zatím možné pouze přes porty součástek. Při vytvoření projení se přejmenují všechny připojené vodiče nacházející se ve stejné síti.

Název vodiče je buď odvozen od názvu součástky v případě vstupů a výstupů a nebo je přiřazen výchozí název „net“ + pořadová číslice.

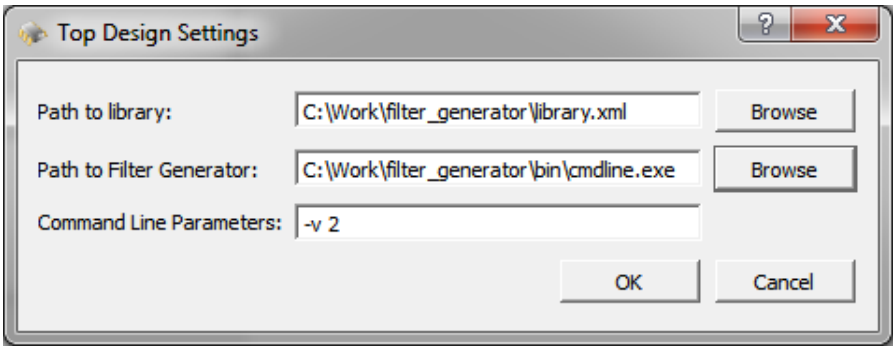
### 2.2.7 Dialogové okno pro vložení názvu souboru - třída NewfileDialog




Obr. 2.4: Dialogové okno - Nový soubor

Na obr. č. 2.4 je nejjednodušší dialogové okno pro vložení názvu. Okno je popsáno v souborech newfiledialog.h, newfiledialog.cpp a newfiledialog.ui. Textový ovládací prvek třídy *QLineEdit* je předvyplněn názvem „untitled“. Název musí být bez diakritiky, jelikož se z něj odvozuje název pro entitu ve VDHL.

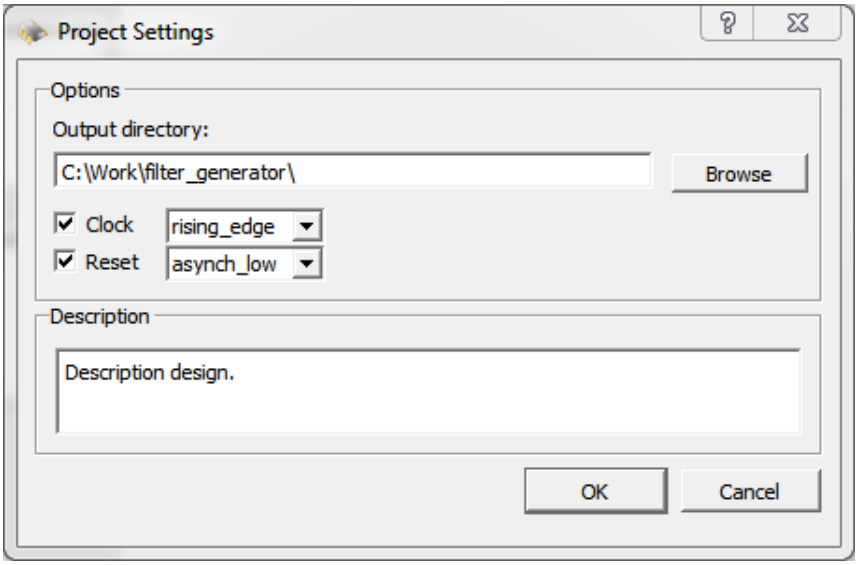
### 2.2.8 Dialogové okno hlavního nastavení programu - třída MainSettingsDialog




Obr. 2.5: Dialogové okno - Hlavní nastavení

Hlavní nastavení programu se nachází v souborech s názvy mainsettingsdialog.h, mainsettingsdialog.cpp a mainsettingsdialog.ui. V nastavení, které je zobrazeno na obr. č. 2.5, je možné změnit cestu ke knihovně a ke generátoru při změně knihovny se musí přepsat seznamy knihoven a součástek. Také se nastavují parametry pro generátor VHDL.

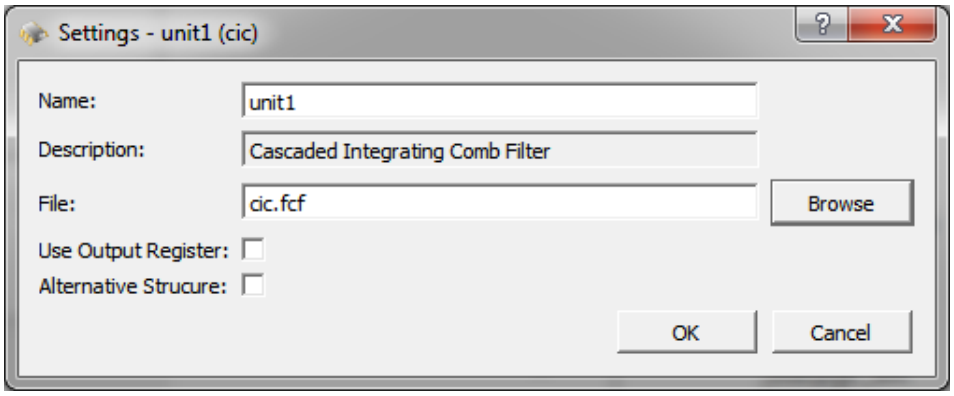
### 2.2.9 Dialogové okno pro nastavení projektu - třída ProjectSettingsDialog




Obr. 2.6: Dialogové okno - Nastavení projektu

Dialogové okno *ProjectSettingsDialog* obsahuje textový ovládací prvek třídy *QLineEdit* nastavující pracovní adresář. Samotné vybírání pracovního adresáře probíhá přes klasické dialogové okno pro vybírání souborů či adresářů. Dále jsou k dispozici dva rozbalovací seznamy pro nastavení události hodinového signálu a resetu, dvě zaškrtávací pole pro možnost vypnutí hodin a resetu a textový víceřádkový ovládací prvek třídy *QTextEdit*, do kterého může uživatel přidat popis zapojení. Vložený komentář může být s diakritikou. Nastavený pracovní adresář by měl být bez diakritiky. Popis dialogového okna na obr. č. 2.6 se nachází ve zdrojových souborech projectsettingsdialog.h, projectsettingsdialog.cpp a projectsettingsdialog.ui.

### 2.2.10 Dialogové okno pro nastavení součástek - třída SymbolSettingsDialog




Obr. 2.7: Dialogové okno - Nastavení součástky

Pro každou součástku se zobrazuje jiné nastavení podle toho, jaké parametry jsou u součástky potřeba nastavit. Např. u vstupů a výstupů nastavujeme pouze název a bitovou délku. Ostatní parametry jsou skryté, tudíž uživatelem nemožné nastavit. Na obr. č. 2.7 je zobrazeno nastavení pro filtry.

### 2.2.11 Čtení z knihovny

Čtení z knihovny (viz příloha P.1) zajišťuje třída *LibStructure*. Třídě se předává již několikrát zmiňovaná cesta ke knihovně, poté je volána funkce *readFile()*, která XML soubor přečte. Do příslušných vektorů si načte součástky typu „basic“, „filter“, „conversion“, „interface“ a „testbench“. Třída obsahuje další podpůrné podtřídy, např. každá načtená součástka uchovává název, popis součástky a čtyři pole ukazatelů na další třídy (parametry, obrys, porty a texty).

### 2.2.12 Čtení a zápis do XML

Zápis do XML a čtení z něj obstarává třída *TopStructure*. Stejně jako *LibStructure* obsahuje několik podpůrných tříd. Z přílohy P.3 jsou patrné třídy podle jednotlivých elementů a jejich vnoření. Např. třída *Design* zahrnuje pole ukazatelů na třídy *DesignPort* (vstupy a výstupy - porty entity), *DesignParts* (jednotlivé komponenty), *DesignNet* (vodiče) a *Plain* (texty). Některé tyto třídy zahrnují další ukazatele.

### 2.2.13 Generování VHDL popisu z XML souboru

Z jednotlivých elementů XML souboru (viz příloha P.3) se vygeneruje VHDL popis. Konkrétně ze vstupů a výstupů se vygeneruje entita, z vodičů (mimo vodiče připojené ke vstupům nebo výstupům) se generují signály s definovanou bitovou šířkou a ze součástek instance komponent nebo přímo VHDL popis dané součástky. Text se při generování nijak nevyužívá. Slouží pouze k popis při tvorbě zapojení. Pro generování zapojení s filtry je zapotřebí mimo XML, také soubor s příponou „*.fcf“, který může být vytvořen např. v programu Matlab. Obsahuje informace o bitové šířce vstupu, výstupu a jeho koeficienty. Pro každý filtr v návrhu je třeba jeden „*.fcf“ soubor.

## 2.3 Popis XML souboru

Navržený program generuje XML soubor s popisem nakresleného zapojení. V souboru musí být uloženy všechny informace pro zpracování programem, který bude generovat vrcholový popis ve VHDL, a informace pro uložení zapojení. V příloze P.3 je příklad XML souboru. V následují tabulce je srovnání názvů jednotlivých částí v XML souboru a VHDL:
Celý soubor je kódován v UTF-8, který je pro XML výchozí. Pro XML soubor je použita verze 1.0 a pro generátor VHDL verze 2.0. V elementu <settings> se nachází slovní popis daného zapojení, nastavení pracovního adresáře, události hodinového signálu a resetu.

V elementu <design> jsou atributy „name“ pro název zapojení a „library“ pro členění zapojení do knihoven. Potomky elementu <ports> jsou elementy <port>, ve kterých je popis jednotlivých vstupů a výstupů celého navrženého zapojení: jméno, směr toku dat (vstup/výstup), bitová šířka, případně jejich poloha.

Element <parts> v sobě zahrnuje jednotlivé součástky <part>. V každé součástce jsou atributy s jejím názvem („name“), polohou na kreslící ploše (souřadnice x, y), knihovnou, ve které se nachází, a jménem. Element <generic> slouží pro uložení parametrů součástky, např. nastavení bitové šířky, se kterou má

Tab. 2.2: Porovnání XML a VHDL

| XML | VHDL |
|---|---|
| ports | entita |
| port | porty entity |
| parts | výčet instancí komponenty |
| part | instance komponenty |
| pin | port komponenty |
| nets | výčet signálů |
| net | definice signálů |
| wire, text, plains | nemá význam pro VHDL |

součástka pracovat. Elementy <pin> jsou pak jednotlivé signály (vstupy a výstupy součástky - entity). Jejich atributem je název signálu, směr toku dat a název signálu, ke kterému jsou připojeny.

Dalším elementem je <nets>, kde jsou zapsány informace o propojení součástek. Název signálu je stejný jako v elementu <pin> a bitová šířka je opět stejná jako u součástek, který daný signál propojuje. Každý element <wire> obsahuje souřadnice o začátku a konci jednotlivých čar vodiče.

Posledním elementem je <plains>. Tento element je využívám pro ukládání informací, které program pro generování VHDL popisu nepotřebuje. Do elementu <text> se ukládají texty vložené uživatelem.

# 3 UŽIVATELSKÝ MANUÁL

S ohledem na standardy VHDL by se v celém programu neměla používat diakritika.

## 3.1 První spuštění

Při prvním spuštění je potřeba nastavit cestu ke knihovně („library.xml“), kterou chceme používat a ke generátoru VHDL („cmdline.exe“). Po načtení je zobrazeno hlavní okno programu. V menu „File“ lze založit nový projekt nebo otevřít rozpracovaný. Dále je možné změnit v nastavení „Main Settings“ nastavené cesty ke knihovně a generátoru VHDL.

## 3.2 Založení nového projektu

Při každém založení nového projektu je nutno zadat název, poté je zobrazeno kreslící plátno a seznamy s načtenými knihovnami a součástkami. Pro nastavení parametrů celého zapojení slouží okno „Project Settings“ přístupné v menu „Tools“ a nebo přímo z panelu nástrojů. V tomto nastavení je možné změnit zdrojový adresář, nastavit události hodinového signálu, reset a přidat krátký komentář k zapojení.

## 3.3 Práce se součástkami

Pro vložení součástky je nutné vybrat konkrétní součástku v seznamu, stisknout tlačítko „Insert“ a poté kliknout na místo na kreslící ploše, kam se součástka vloží. Postup lze urychlit dvojitým kliknutím na součástku v seznamu a umístěním na kreslící plochu. Vstupy a výstupy se vkládají pomocí tlačítek v panelu nástrojů. Po vložení součástky je automaticky zapnut režim pro pohyb. Součástky se pohybují po mřížce.

Mazání součástek je možné pomocí tlačítka „Delete“ v panelu nástojů, přes kontextové menu nebo pomocí klávesy „Delete“.

Nastavení součástky je možné vyvolat přes kontextové menu vybrané součástky. Lze měnit jméno součástky a parametry vyplývající z funkce součástky.

## 3.4 Práce s vodiči

Pro vložení vodiče je nutné kliknout na port součástky. V této chvíli se začne vykreslovat vodič. Levým tlačítkem myši lze přidat zalomení vodiče. Ukončit vodič lze opět na portu součástky. Mazání vodiče při kreslení je možné pomocí pravého

tlačítka myši. Nakreslený vodič se maže stejně jako součástka. Vytváření uzlů je možné pouze přes porty součástek.

## 3.5 Vkládání textu

Do zapojení je možné vkládat komentáře vložením textu o různých velikostech. S textem je možné libovolně pohybovat. Dvojitým kliknutím na text lze již vložený text editovat.

## 3.6 Ukládání

Ukládat lze pomocí funkcí „Save“ a „Save As“. Při použití funkce „Save“ se uloží projekt do pracovního adresáře. Pokud je třeba uložit projekt do jiného adresáře, jsou dvě možnosti. První možností je změnit zdrojový adresář v nastavení „Project Settings“ a uložit opět pomocí funkce „Save“. Druhou možností je použít funkci „Save As“ a vybrat si požadovaný adresář pomocí dialogového okna. Pro další práci v programu bude tento adresář nastaven jako pracovní.

## 3.7 Načítání

Při načítání uloženého XML souboru je nastavení projektu odvozeno od uložených parametrů. Jakmile je zapojení načteno, lze pokračovat v jeho editaci.

## 3.8 Generování VHDL

Pro generování zapojení do VHDL slouží funkce „Generate VHDL“. Po vygenerování je zobrazena zpráva, kde si lze prohlédnout podrobnosti o generování (informace, varování, chyby apod).

## 3.9 Další užitečné funkce

Dalšími funkcemi usnadňující práci je zobrazení mřížky a export zapojení do formátů BMP, JPEG nebo PNG. Exportovat lze buď celé kreslící plátno, nebo jen jeho aktuálně zobrazenou část.

## 3.10 Vypnutí programu

Při vypnutí programu jsou cesty ke knihovně a ke generátoru uloženy do registru, aby byly při dalším spuštění načteny a nemusely se zadávat při každém spuštění.

# ZÁVĚR

Cílem této práce bylo seznámit se s problematickou návrhu digitálních filtrů a jejich návrhu v programu MATLAB. Dále se seznámit s grafickým programových prostředím Qt4 Framework. Hlavním úkolem však byl návrh grafického uživatelského rozhraní pro návrh filtrů.

V teoretické části byla rozebrána problematika digitálních filtrů a jejich návrh pomocí programu FDATool, který je součástí programového prostředí MATLAB. Krátce byl rozebrán značkovací jazyk XML a jeho syntaxe. Poslední částí teoretického úvodu bylo seznámení s prostředím Qt4 Framework, ve kterém je celý program navržen.

V kapitole Návrh programu byl krátce zmíněn program, na který práce navazuje. Dále bylo popsáno navržené grafické uživatelské rozhraní (GUI) a funkce programu s ohledem na jednotlivé vytvořené třídy. Celý program byl propojen s již hotovým programem pro generování VHDL popisu. Byl rozebrán ukládaný XML soubor s ohledem na VHDL.

Vytvořený program umožňuje vytvářet zapojení filtrů, ukládat jej do XML souboru a generovat jeho VHDL popis. Parametry pro generování VHDL lze nastavovat v nastavení zapojení a nastavení jednotlivých součástek.

Ovládání programu je popsáno v kapitole Uživatelský manuál. V přílohách se nachází příklad načítané součástky, ukládaného XML souboru a VHDL popis zapojení.

V dalším vývoji aplikace se předpokládá vytváření vlastních symbolů z vloženého zapojení a následné vnořování zapojení do sebe. Dále je potřebné zjednodušit práci s vodiči a vytváření uzlů.

# LITERATURA


[1] Digital Filtering. *WaveMetric* [online]. [cit. 2013-12-06]. Dostupné z: http://www.wavemetrics.com/products/igorpro/dataanalysis/signalprocessing/digitalfilters.htm.

[2] Realizace filtrů FIR a IIR v programovacím jazyce C#. *Programujte.com* [online]. 2010 [cit. 2013-12-06]. Dostupné z: http://programujte.com/clanek/2010050400-realizace-filtru-fir-a-iir-v-programovacim-jazyce-c/.

[3] XILINX. *Cascaded Integrator-Comb (CIC) Filter V3.0* [online]. 2002 [cit. 2013-12-07]. Dostupné z: http://www.ux.uis.no/~karlsk/MIK200/dok/XilinxCICfilter.pdf/.

[4] E. B. Hogenauer, An economical class of digital filters for decimation and interpolation *IEEE Transactions on Acoustic, Speech and Signal Processing*. 1981 [cit. 2013-12-07].

[5] SMITH, Julius O. Introduction to digital filters. *Center for Computer Research in Music and Acoustics (CCRMA), Stanford University* [online]. [cit. 2013-12-07]. Dostupné z: https://ccrma.stanford.edu/~jos/fp/.

[6] Značkovací jazyky a jejich použití (2). In: *UbiQue Webs: Blog.UbiQue.cz* [online]. 2013 [cit. 2013-11-17]. Dostupné z: http://blog.ubique.cz/2013/04/26/znackovaci-jazyky-a-jejich-pouziti-2/.

[7] WALSH, Norman. A Technical Introduction to XML: What Do XML Documents Look Like?. In: *XML.com* [online]. 1998 [cit. 2013-12-03]. Dostupné z: http://www.xml.com/pub/a/98/10/guide0.html?page=3/.

[8] KOSEK, Jiří. XML schémata. *Vše o WWW* [online]. 2013 [cit. 2013-12-03]. Dostupné z: http://www.kosek.cz/xml/schema/.

[9] KEPRT, Aleš. XML. In: *Osobní stránky Aleše Keprta, Ph.D.* [online]. 2005 [cit. 2013-12-03]. Dostupné z: http://www.keprt.cz/texty/xml.pdf/.

[10] BLANCHETTE, Jasmin a Mark SUMMERFIELD. *C GUI programming with Qt 4* [online]. 2006, 537 s. [cit. 2013-12-06]. Dostupné z: http://www.qtrac.eu/C++-GUI-Programming-with-Qt-4-1st-ed.zip/.

[11] CHROBOCZEK, Martin. *Grafická uživatelská rozhraní v Qt a C++* Brno: Computer Press, 2013, 392 s. ISBN 978-80-251-4124-3.

[12] What is Qt?. *Qt Project* [online]. 2013 [cit. 2013-12-06]. Dostupné z: https://qt-project.org/doc/qt-4.8/.

# SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ASCII American Standard Code for Information Interchange – Americký standardní kód pro výměnu informací

BMP Bitmap image file – Formát pro ukládaní rastrové grafiky

CIC Cascaded Integrating Comb – Hřebenový integrační filtr

FIR Finite Impulse Response – Filtr s konečnou impulzní odezvou

GML Generalized Markup Language – Zobecněný značkovací jazyk

GUI Graphical User Interface – Grafické uživatelské prostředí

IIR Infinite Impulse Response – Filtr s nekonečnou impulzní odezvou

JPEG Joint Photographic Experts Group – Formát pro ztrátovou kompresi grafiky

PNG Portable Network Graphics – Přenosný formát pro bezeztrátovou kompresi rastrové grafiky

SGML Standard Generalized Markup Language – Standardizovaný zobecněný značkovací jazyk

SOS Second-Order Sections – Filtry druhých řádů

SPI Serial Peripheral Interface– Sériové periferní rozhraní

UART Universal Asynchronous Reciever/Transmiter – Asynchronní komunikační rozhraní

VHDL Very High Speed Integrated Circuits Hardware Description Language – Jazyk pro popis velmi rychlých integrovaných obvodů

XML Extensible Markup Language – Rozšiřitelný značkovací jazyk

XWG XML Working Group – Pracovní skupina pro XML

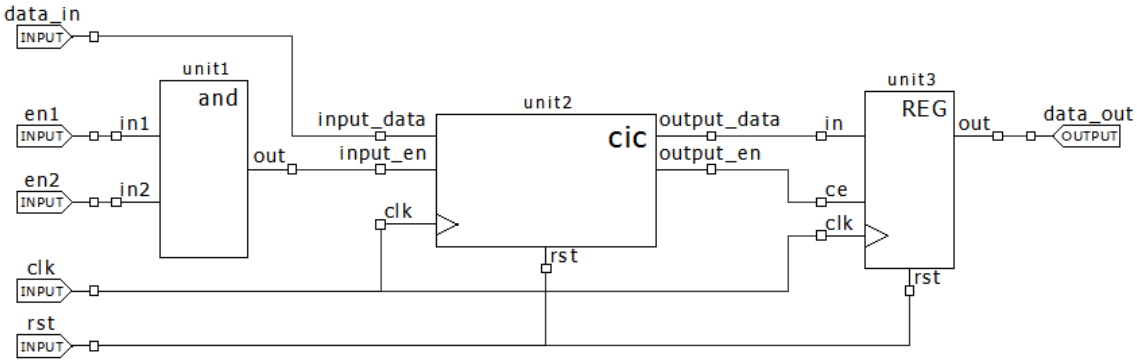
# SEZNAM PŘÍLOH

# P.1 Příklad součástky z knihovny

```
<?xml version="1.0" encoding="utf-8"?>

<library name="filter">
<symbol name="cic" description="Cascaded Integrating Comb Filter">
        <param name="file" type="string" default=""/>
        <param name="output_reg" type="string" default="no"/>
        <param name="alternative" type="string" default="no"/>
        <wire x1="-100" y1="60" x2="100" y2="60" width="3" layer="2"/>
        <wire x1="100" y1="60" x2="100" y2="-60" width="3" layer="2"/>
        <wire x1="100" y1="-60" x2="-100" y2="-60" width="3" layer="2"/>
        <wire x1="-100" y1="-60" x2="-100" y2="60" width="3" layer="2"/>
        <wire x1="-100" y1="-30" x2="-80" y2="-40" width="3" layer="1"/>
        <wire x1="-100" y1="-50" x2="-80" y2="-40" width="3" layer="1"/>
        <wire x1="-150" y1="40" x2="-100" y2="40" width="3" layer="1"/>
        <wire x1="-150" y1="10" x2="-100" y2="10" width="3" layer="1"/>
        <wire x1="-150" y1="-40" x2="-100" y2="-40" width="3" layer="1"/>
        <wire x1="150" y1="40" x2="100" y2="40" width="3" layer="1"/>
        <wire x1="150" y1="10" x2="100" y2="10" width="3" layer="1"/>
        <wire x1="0" y1="-60" x2="0" y2="-80" width="3" layer="1"/>
        <pin name="input_data" x="-150" y="40" direction="in" width="0"/>
        <pin name="input_en" x="-150" y="10" direction="in" width="1"/>
        <pin name="output_data" x="150" y="40" direction="out" width="0"/>
        <pin name="output_en" x="150" y="10" direction="out" width="1"/>
        <pin name="clk" x="-150" y="-40" direction="in" width="1"/>
        <pin name="rst" x="0" y="-80" direction="in" width="1"/>
        <text text="_NAME" x="100" y="60" size="20" layer="5"/>
        <text text="_VALUE" x="100" y="60" size="12" layer="6"/>
    </symbol>
</library>
```

# P.2 Exportované zapojení do PNG




Obr. P.2: Schéma zapojení CIC filtru

# P.4 Příklad ukládaného XML souboru

```
<?xml version="1.0" encoding="utf-8" ?>
<filter_generator format="2.0"/>

<settings>
    <description text=""/>
    <directory dir=""/>
    <general clk="rising_edge" reset="asynch_low"/>
</settings>

<design name="top" library="work">
    <ports>
        <port name="data_in" x="-400" y="140" direction="in" width="0"/>
        <port name="en1" x="-400" y="50" direction="in" width="1"/>
        <port name="en2" x="-400" y="-10" direction="in" width="1"/>
        <port name="clk" x="-400" y="-90" direction="in" width="1"/>
        <port name="rst" x="-400" y="-140" direction="in" width="1"/>
        <port name="data_out" x="550" y="50" direction="out" width="0"/>
    </ports>

    <parts>
        <part name="unit1" value="" x="-250" y="20" library="basic" device="and">
            <param W="0"/>
            <pin name="in1" direction="in" net="en1"/>
            <pin name="in2" direction="in" net="en2"/>
            <pin name="out" direction="out" net="net3"/>
        </part>
        <part name="unit2" value="" x="60" y="10" library="filter" device="cic">
            <param alternative="no" file="cic.fcf" output_reg="no"/>
            <pin name="input_data" direction="in" net="data_in"/>
            <pin name="input_en" direction="in" net="net3"/>
            <pin name="output_data" direction="out" net="net1"/>
            <pin name="output_en" direction="out" net="net2"/>
            <pin name="clk" direction="in" net="clk"/>
            <pin name="rst" direction="in" net="rst"/>
        </part>
        <part name="unit3" value="" x="390" y="10" library="basic" device="reg_ce">
            <param W="0"/>
            <pin name="in" direction="in" net="net1"/>
            <pin name="out" direction="out" net="data_out"/>
            <pin name="clk" direction="in" net="clk"/>
            <pin name="rst" direction="in" net="rst"/>
            <pin name="ce" direction="in" net="net2"/>
        </part>
    </parts>

    <nets>
        <net name="net1" width="0">
            <wire x1="210" y1="50" x2="310" y2="50"/>
            <wire x1="310" y1="50" x2="310" y2="50"/>
        </net>
```

```
        <net name="net2" width="0">
            <wire x1="210" y1="20" x2="280" y2="20"/>
            <wire x1="280" y1="20" x2="280" y2="-10"/>
            <wire x1="280" y1="-10" x2="310" y2="-10"/>
        </net>
        <net name="net3" width="0">
            <wire x1="-170" y1="20" x2="-90" y2="20"/>
            <wire x1="-90" y1="20" x2="-90" y2="20"/>
        </net>
        <net name="data_in" width="0">
            <wire x1="-350" y1="140" x2="-170" y2="140"/>
            <wire x1="-170" y1="140" x2="-170" y2="50"/>
            <wire x1="-170" y1="50" x2="-90" y2="50"/>
        </net>
        <net name="en1" width="1">
            <wire x1="-350" y1="50" x2="-330" y2="50"/>
            <wire x1="-330" y1="50" x2="-330" y2="50"/>
        </net>
        <net name="en2" width="1">
            <wire x1="-350" y1="-10" x2="-330" y2="-10"/>
            <wire x1="-330" y1="-10" x2="-330" y2="-10"/>
        </net>
        <net name="clk" width="1">
            <wire x1="-350" y1="-90" x2="-90" y2="-90"/>
            <wire x1="-90" y1="-90" x2="-90" y2="-30"/>
            <wire x1="3" y1="3" x2="3" y2="3"/>
            <wire x1="-350" y1="-90" x2="280" y2="-90"/>
            <wire x1="280" y1="-90" x2="280" y2="-40"/>
            <wire x1="280" y1="-40" x2="310" y2="-40"/>
        </net>
        <net name="rst" width="1">
            <wire x1="-350" y1="-140" x2="60" y2="-140"/>
            <wire x1="60" y1="-140" x2="60" y2="-70"/>
            <wire x1="3" y1="3" x2="3" y2="3"/>
            <wire x1="-350" y1="-140" x2="390" y2="-140"/>
            <wire x1="390" y1="-140" x2="390" y2="-90"/>
        </net>
        <net name="data_out" width="0">
            <wire x1="470" y1="50" x2="500" y2="50"/>
            <wire x1="500" y1="50" x2="500" y2="50"/>
        </net>
    </nets>
</design>
```

# P.4 Vygenerovaný VHDL popis zapojení

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity top is
    port (
        data_in : in std_logic_vector(1 downto 0);
        en1 : in std_logic;
        en2 : in std_logic;
        clk : in std_logic;
        rst : in std_logic;
        data_out : out std_logic_vector(11 downto 0)
    );
end top;

architecture rtl of top is

    signal net1 : std_logic_vector(11 downto 0);
    signal net2 : std_logic;
    signal net3 : std_logic;

begin

    -- instance 'unit1', device 'and' (bitwise AND)
    net3 <= en1 and en2;

    -- instance: unit2, device: cic, library: filter
    i_unit2 : entity work.unit2_filter(rtl)
        port map (
            input_data => data_in,
            input_en => net3,
            output_data => net1,
            output_en => net2,
            clk => clk,
            rst => rst
        );

    -- instance unit3, device 'reg_ce' (register with reset and clock enable)
    i_unit3: process (clk, rst) begin
        if (rst = '0') then
            data_out <= (others => '0');
        elsif rising_edge(clk) then
            if (net2 = '1') then
                data_out <= net1;
            end if;
        end if;
    end process;


end rtl;
```

## P.5 Obsah přiloženého CD

- **adresář fg01**
  - zdrojové soubory k programu
- **adresář fg01_exe**
  - soubor ke spuštění program
  - nezbytné knihovny (*.dll)
  - knihovna součástek („library.xml“)
  - generátor VHDL popisu („cmdline.exe“)
  - ukázky zapojení
- **elektronická verze bakalářské práce**