ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA DOPRAVNÍ

Kateřina Pithartová

APLIKACE INOVATIVNÍCH METOD VÝPOČTOVÉ TOMOGRAFIE PRO POKROČILOU ANALÝZU ZKOUMANÝCH OBJEKTŮ

Bakalářská práce

2019

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K618 .....Ústav mechaniky a materiálů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Kateřina Pithartová

Kód studijního programu a studijní obor studenta:

B 3710 – ITS – Inteligentní dopravní systémy

Název tématu (česky): **Aplikace inovativních metod výpočtové tomografie pro pokročilou analýzu zkoumaných objektů**

Název tématu (anglicky): Application of Innovative Techniques of Computed Tomography for the Advanced Analysis of the Irradiated Objects

Zásady pro vypracování

Při zpracování bakalářské práce se řiďte osnovou uvedenou v následujících bodech:

- Konvenční výpočtová tomografie (CT) je v současnosti zavedenou metodou pro vizualizaci a analýzu zkoumaných objektů. V současnosti jsou metody výpočtové tomografie předmětem rozsáhlého výzkumu zahrnujícího zdokonalování existujících metod i vývoj metod zcela nových. Na spolupracujícím Ústavu teoretické a aplikované mechaniky AV ČR vznikne v rámci výzkumné činnosti nové zařízení pro vývoj inovativních metod výpočtové tomografie, s nímž bude náplň práce spojena.
- Bakalářská práce bude zaměřena na zprovoznění CT zařízení pro vývoj inovativních tomografických metod, včetně spolupráce na jeho návrhu, vývoji a implementaci měřicích procedur. S využitím zařízení bude možné provádět pokročilou analýzu materiálů včetně metod in-situ zatěžování.
- Výstupy práce budou vedle samotného zprovoznění zařízení pro inovativní tomografická měření zahrnovat zpracování technické dokumentace včetně podrobného popisu, schémat elektrického zapojení a diagramů funkčních bloků. Výstupem práce bude rovněž sada nástrojů (např. instruktážní pokyny, softwarové nástroje apod.) pro provádění vybraného tomografického měření. Zvolená funkcionalita zařízení bude v práci vhodným způsobem demonstrována (např. ukázkou měření).
- Cílem práce je vývoj a zprovoznění vybraných funkcí tomografického zařízení, jež bude dále sloužit pro výzkum v oblasti inovativních tomografických metod, vč. metod nedestruktivního testování a in-situ testování materiálů.

Rozsah grafických prací: není stanoven

Rozsah průvodní zprávy: minimálně 35 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury:

Carmignato S. et al. (2017), Industrial X-Ray Computed Tomography. Technology & Engineering, Springer, ISBN 978-3-319-59571-9

Wang M. et al. (2015), Industrial Tomography - Systems and Applications, Woodhead Publishing, ISBN 978-1-782-42118-4

Margui E., (2013), X-Ray Fluorescence Spectrometry and Related Techniques: An Introduction, Momentum Press

Vedoucí bakalářské práce: **Ing. Jan Šleichrt**

Ing. Tomáš Fíla

Datum zadání bakalářské práce: **10. srpna 2018**

(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání bakalářské práce: **26. srpna 2019**

a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta dopravní -3-

L. S.

prof. Ing. Ondřej Jiroušek, Ph.D.
vedoucí
Ústavu mechaniky a materiálů

doc. Ing. Pavel Hrubeš, Ph.D.
děkan fakulty

Potvrzuji převzetí zadání bakalářské práce.

Kateřina Pithartová
jméno a podpis studenta

V Praze dne 10. srpna 2018

Poděkování

Tato bakalářská práce by nemohla vzniknout bez pomoci a podpory řady lidí v mém okolí. Touto cestou bych chtěla poděkovat vedoucím své práce Ing. Janu Šleichrtovi a Ing. Tomáši Fílovi, za vynikající odborné vedení při zpracovávání i psaní této práce. Dále bych chtěla poděkovat Ing. Tomáši Musilovi Ph.D, za vstřícnost a pomoc při porozumění dané problematice. V neposlední řadě bych chtěla poděkovat své rodině a přátelům za podporu během celého studia na Fakultě dopravní.

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na ČVUT v Praze Fakultě dopravní.

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Nemám žádný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 21. srpna 2019 .....

podpis

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta dopravní

Aplikace inovativních metod výpočtové tomografie pro pokročilou analýzu zkoumaných objektů

bakalářská práce

srpen 2019

Kateřina Pithartová

ABSTRAKT

Výpočtová tomografie je zavedenou metodou umožňují zkoumat vnitřní strukturu objektu bez jeho porušení. Bakalářská práce Aplikace inovativních metod výpočtové tomografie pro pokročilou analýzu zkoumaných objektů je spojena s experimentálním zařízením pro vývoj inovativních metod výpočtové tomografe.

Tato bakalářská práce se zabývá sestavením a uvedením do provozu výpočtového tomografu vzniklého na spolupracujícím Ústavu teoretické a aplikované mechaniky, AV ČR. Dále obsahuje technickou dokumentaci a schémata zapojení.

KLÍČOVÁ SLOVA

výpočtová tomografie, programovatelné hradlové pole, řízení experimentálních zařízení, konfigurace

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of transportation sciences

Application of Innovative Techniques of Computed Tomography for the Advanced Analysis of the Irradiated Objects

bachelor thesis

August 2019

Kateřina Pithartová

ABSTRACT

Computed tomography is a common method that allows analyzing inner structure of an object without corrupting it. The aim of this bachelor thesis 'Application of Innovative Techniques of Computed Tomography for the Advanced Analysis of the Irradiated Objects', developed at Institute of Theoretical and Applied Mechanics of the Czech Academy of Sciences, is to assemble and to put into operation a custom developed experimental tomograph.

Written part of this thesis describes the work carried out and necessary steps; includes technical documentation and wiring diagrams.

KEY WORDS

computed tomography, field-programmable gate array, configuration and controlling of experimental devices

Obsah

Seznam použitých zkratek **8**

1 Úvod **9**

2 Seznámení s problematikou **11**

2.1 Výpočtová tomografie 11

2.2 Rentgenová fluorescenční analýza 12

2.3 Popis zařízení 12

2.4 Způsob řešení 13

2.4.1 Cíle práce 14

3 Metody **16**

3.1 Programovatelné hradlové pole 16

3.1.1 Základní popis hradlových polí 16

3.1.2 Obecná architektura FPGA 17

3.1.3 Konfigurace 18

3.1.4 Xilinx Spartan 3 19

3.2 Programovací jazyk VHDL 22

3.2.1 Úvod do jazyka VHDL 22

3.2.2 Architektura 22

3.2.3 Knihovny a balíčky 23

3.2.4 HostMot2 . . . 24
3.3 Řídicí karty . . . 25
3.3.1 Mesa Electronics . . . 25
3.4 LinuxCNC . . . 26
3.5 Xilinx ISE Design Suite . . . 26
3.5.1 Práce v ISE Design Suite . . . 26
3.5.2 Uživatelské rozhraní . . . 27

4 Návrh a kompilace souboru pro Mesa FPGA I/O cards 28

4.1 Návrh konfiguračního souboru . . . 28
4.1.1 Deklarace knihoven a balíčků . . . 28
4.1.2 Inicializační prostředí . . . 29
4.1.3 Piny . . . 30
4.2 Kompilace konfiguračního souboru . . . 32
4.2.1 Stažení podpůrného software . . . 32
4.2.2 Příprava pracovního projektu . . . 33
4.2.3 Syntéza a implementace . . . 35
4.2.4 Generování konfiguračního bit souboru . . . 35

5 Realizace 37

5.1 Popis jednotlivých komponent . . . 37
5.1.1 Lineární osy . . . 37
5.1.2 Driver . . . 37
5.1.3 Enkodér . . . 38
5.1.4 Rotační stolek . . . 38
5.1.5 Mesa karty . . . 39
5.2 Fyzická realizace . . . 41

5.2.1 Pilotní zapojení a testovací provoz 41

5.2.2 Finální zapojení 41

6 Referenční měření **45**

7 Závěr **48**

Seznam použité literatury **50**

Seznam obrázků **54**

A Moduly a piny **55**

B Konfigurační soubor **60**

C Šablona pro vytvoření uživatelského konfiguračního souboru **65**

D Schéma zapojení řídících karet **70**

Seznam použitých zkratek

ÚTAM	Ústav teoretické a aplikované mechaniky AV ČR
CT	Výpočetní tomografie
HAL	Hardware abstraction layer
RTG	Rentgenový
XRF	Rentgenová fluorescenční spektrometrie
LUT	Look up table
FPGA	Programovatelné hradlové pole
HSIC	Very High Speed Integrated Circuits
VHDL	HSIC Hardware Description Language
I/O	Input / output
PWM	Pulzně šířková modulace
SSI	Synchronous Serial Interface
BISS-C	Bidirectional serial synchronous interface
KS	Koncový spínač
CPR	Counts per revolution

1. Úvod

Výpočtová tomografie umožňuje detailní zkoumání vnitřní struktury objektu bez nutnosti jeho porušení. Díky technickému pokroku v této oblasti se metoda stala konvenčně používanou. V sektoru dopravy se jedná například o defektoskopii a inspekci dopravních prostředků. V současnosti dochází k rozsáhlému vývoji metod existujících i zcela nových. S využitím výzkumu probíhajícího na Ústavu teoretické aplikované mechaniky Akademie věd České republiky vznikne nové zařízení přinášející inovace v této oblasti.

Bakalářská práce si klade za cíl sestavení, zapojení a uvedení do provozu experimentálního zařízení, které kombinuje výpočtovou tomografii s rentgenovou fluorescenční analýzou. Díky tomu může být provedena komplexnější analýza zkoumaných objektů umožňující zkoumání prvkového složení objektu. Řízení nestandardního zařízení s sebou nese řadu rizik. Zařízení obsahuje vzájemně nekompatibilních rozhraní, které je nezbytné jednotně zapojit a řídit. Řešením tohoto problému bylo použití programovatelných řídících karet, které umožňují připojení celé řady zařízení, v závislosti na své konfiguraci. Speciální pozornost při zprovozňování bude zaměřena na vytvoření unikátního konfiguračního souboru navrženého dle specifických požadavků CT.

Vytváření uživatelských konfigurací pro řídící karty není běžnou praxí. Vzhledem k absenci návodů a komentářů bude mít pochopení této problematiky velký přesah i mimo aplikaci tohoto zařízení. Doposud byl návrh řízení nových zařízení na ÚTAM limitován dostupnými konfiguračními soubory. CT bude prvním zařízením s funkcionalitou řídících karet vytvořenou dle konkrétních požadavků na řízení.

Dalším výstupem práce bude sepsání návodu a dokumentace, která bude uživateli vysvětlovat postup od návrhu po kompilaci. Ten bude umožňovat komunitě pracující s tímto typem karet vytváření uživatelských konfiguračních souborů se širokým spektrem využitelnosti.

Práce s radiologickými zařízeními s sebou však nese riziko časové náročnosti při uvedení do provozu, protože jejich použití musí být nejprve schváleno Státním úřadem pro jadernou bezpečnost.

2. Seznámení s problematikou

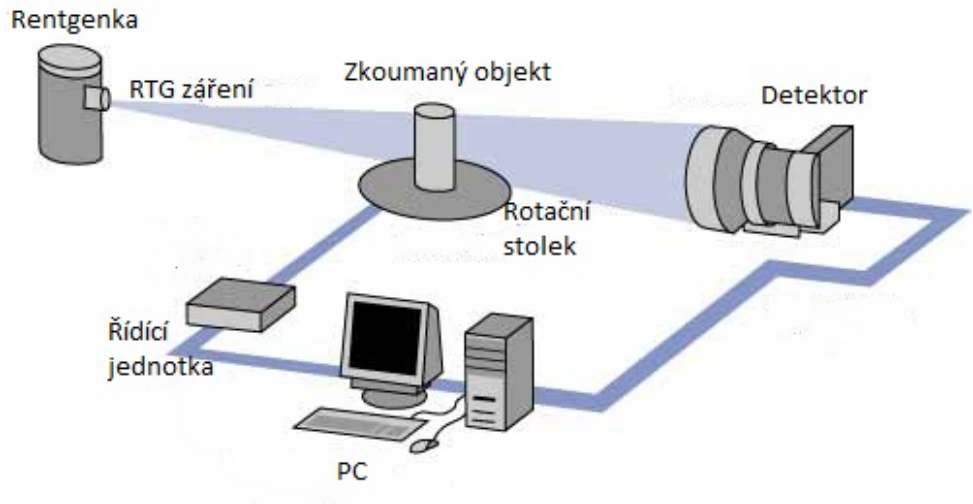
2.1 Výpočtová tomografie

Výpočtová tomografie je zobrazovací metoda, umožňující na základě rentgenových snímků objektů, pořízených z různých úhlů, sestavit 3D model vnitřní a vnější struktury zkoumaného objektu bez toho, aby došlo k porušení daného objektu. Díky tomu může být CT použito k detailnímu zkoumání velmi cenných objektů [1].

V běžné radiografii je zkoumaný předmět umístěn mezi zdroj rentegového záření a citlivý rentgenový film. Při průchodu záření objektem dochází k jeho absorpci v závisloti na materiálu či materiálech, ze kterých je objekt tvořen, přesněji na jejich absorpčních koeficientech. Detekcí intenzity záření za objektem je získán radiograf, tedy snímek o stupních šedi právě v závislosti na útlumu záření v objektu. Nahrazením rentgenového filmu detektorem záření umožňuje digitalizaci snímku a jeho následné zpracování [2].

Principem výpočtové tomografie je získání sady snímků stejného objektu pořízených s úhlovým inkrementem. Ze získané sady snímků je následně matematickým algoritmem dopočítán 3D model zkoumaného objektu, tzv. rekonstrukce [2].

Základní sestava laboratorního tomografu se skládá ze zdroje rentgenového záření, detektoru rentgenového záření a polohovacího systému. Standardní CT zařízení používané v medicíně jsou založena na principu staticky umístěného zkoumaného objektu - například lidského těla - okolo něhož se otáčí zdroj záření a detektor. Ve vědecké sféře se častěji používá principu opačného, kde zdroj záření a detektor jsou statické a otáčí se zkoumaným objektem. Kvůli nežádoucím účinkům ionizačního záření musí být místnost, ve které se CT nachází, odstíněna vhodným materiálem, absorbujícím rentgenové záření [2].



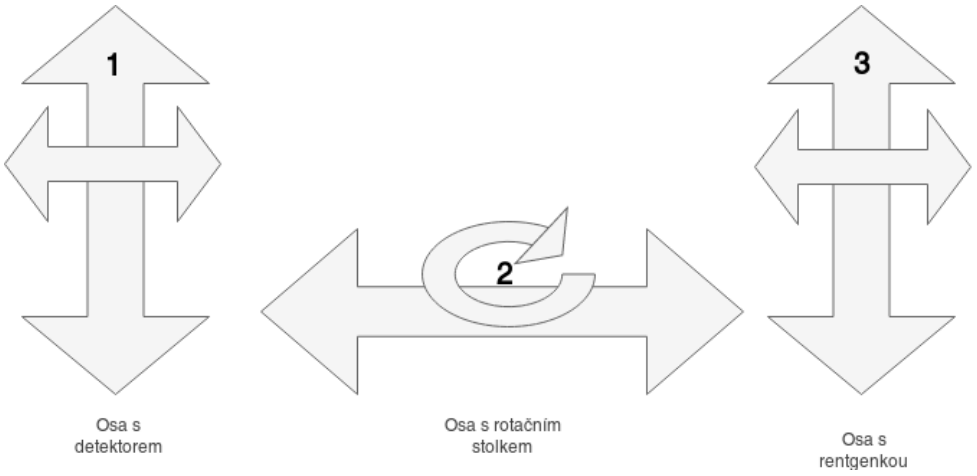
Obrázek 2.1: Schéma laboratorního CT [1]

2.2 Rentgenová fluorescenční analýza

Rentgenová fluorescenční spektrometrie je metoda, která má široké uplatnění při určování prvkové analýzy povrchu zkoumaného objektu. Hlavním principem je interakce částic nebo záření o vysoké energii s atomy zkoumaného vzorku. Při ionizaci dochází k vyrážení elektronů z vnitřních orbitalů. Vzniklé vakance jsou zaplňovány elektrony z jiných vrstev orbitalu. Přitom dochází k uvolňování energie, jejíž velikost se rovná energetickému rozdílu zúčastněných orbitalů. Tato energie je vyzářena jako fluorescenční tok fotonů sekundárního rentgenového záření. Složení vyzářeného fluorescenčního spektra je pro každý prvek charakteristické. Analýza detekovaného spektra umožňuje zjistit prvkové složení objektu. Použití rentgenové fluorescenční spektrometrie však není běžné v kombinaci s CT, přístroj vyvíjený na oddělení biomechaniky ji však implementuje. [3]

2.3 Popis zařízení

Zařízení je v základním sestavě tvořeno třemi na sobě nezávislými sestavami, jejichž schéma je zobrazeno na obrázku 2.2. Jedna sestava pohybuje rentgenkou, druhá detektorem a třetí zkoumaným objektem. Celé zařízení je tvořeno jedním rotačním stolkem a pěti lineárními osami, které jsou poháněny krokovými motory v kombinaci s enkodéry, které slouží k vyčítaní přesné polohy. Kombinace krokových motorů a inkrementálních enkodérů umožňuje dosáhnout přesnosti polohování v řádech jednotek mikrometrů.



Obrázek 2.2: Schéma směrů pohybu třech komponent

První sestava je navržena k polohování detektoru ve třech směrech. Je tvořena jednou hlavní vertikální osou, která pohybuje s druhou horizontální osou. Detektor je umístěn na speciální konstrukci připevněné na horizontání ose. Ta umožňuje manuální otočení detektoru v řádu jednotek stupňů.

Druhá sestava je umístěna naproti sestavě s detektorem. Je navržena pro potřeby polohování rentgenky ve dvou směrech. K tomu jsou využity dvě osy, horizontální a vertikální.

Třetí sestava slouží k polohování zkoumaného objektu, či zařízení, ve kterém je objekt umístěn. Tato sestava se nachází mezi rentgenkou a detektorem a je tvořena jednou horizontální osou s rotečním stolkem. Tento stolek umožňuje otáčení objektu o 360 stupňů, čímž může možné provádět skenování z několika úhlů pro následné sestavení 3D modelu objektu.

2.4 Způsob řešení

V rámci programu je vyvíjen tomograf kombinující tomografii a rentgenovou fluorescenční analýzu, což umožňuje detailněji analyzovat zkoumané objekty. Na tomto projektu pracuje celý tým pracovníků, který zpracovává jednotlivé části, přičemž o některých z nich pojednává tato bakalářská práce.

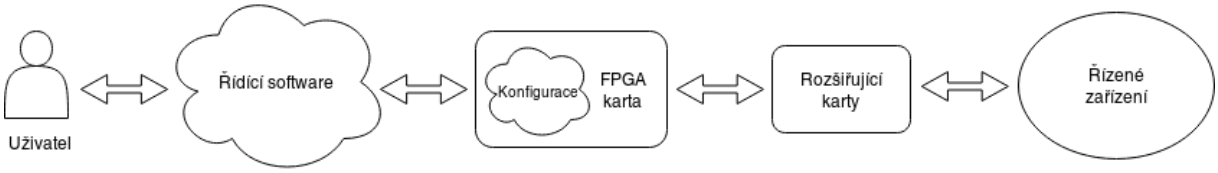
Řízení je navrženo pro požadavek připojení a řízení šesti na sobě nezávislých os, k jejichž polohování jsou využity krokové motory, koncové spínače, brzdy, enkodéry sloužící k přesnému vyčítání polohy objektu. Zařízení rovněž obsahuje mechanismus pro nouzové zastavení pohybu, tzv. E-STOP.

Při řešení polohování takového zažízení je nutné se vypořádat s několika problémy. Zařízení

obsahuje množství rozhraní, která se musí vhodně připojit, dále je nezbytné zařídit datovou komunikaci všech prvků, která bude spolehlivá a bez ztrát. Řízení je nezbytné provádět v reálném čase, aby nedocházelo k prodlevám mezi reakcemi jednotlivých prvků a vykonávané akce probíhaly okamžitě, najednou a bez prodlev (například motory).

Řešení tohoto problému nabízí na míru sestavená řídicí jednotka, která je na rozdíl od řídicích jednotek používaných v průmyslových CNC strojích založena na základové desce standardního počítače, do které jsou připojeny řídicí karty umožňující připojení periferie pro řízení krokových motorů, enkodérů a dalších komponent. Vhodné řešení tohoto problému nabízí kombinace karet společnosti Mesa Electronics, kterou tvoří hlavní karta osazená programovatelným hradlovým polem a rozšiřující karty, obsahující rozhraní pro připojení všech řízených komponent. Hlavní výhoda těchto karet spočívá v tom, že funkce rozšiřujících karet je dána mimo jiné i konfigurací programovatelného hradlového pole, na jejímž základě jsou přenášeny signály mezi řídicím softwarem a koncovým zařízením. Tento typ řídicích karet se na oddělení biomechaniky běžně používá, avšak doposud byly připravené konfigurační soubory při jejich aplikaci postačující. Kombinace koncových zařízení pro řízení tomografu je však na tolik specifická, že není možné jednodušše použít žádnou z jichž připravených konfiguracích. Ukázalo se jako nezbytné sestavit vlastní konfigurační soubor umožňujícího jednotné připojení všech komponent s ohledem na minimalizaci počtu použitých karet, čímž se značně zjednoduší řízení.

Všechny soubory určené ke konfiguraci karet společnosti Mesa Electronics jsou integrovány v balíčku HostMot2, který obsahuje rovněž sadu předpřipravené konfigurační soubory pro různé kombinace rozšiřujících karet a jejich koncových zařízení. Tyto soubory jsou uživateli dostupné a je možné je modifikovat, avšak díky absenci komentářů, popisu jednotlivých funkcí, návodů a manuálu není vytváření uživatelských konfiguračních souborů běžnou praxí.



Obrázek 2.3: Blokové schéma řízení

2.4.1 Cíle práce

Tato bakalářská práce se skládá z teoretické a praktické části, jejichž společným cílem je sestavení a uvedení zařízení do provozu. Rámcově se jedná o několik dílčích částí:

- otestování funkčnosti jednotlivých komponent
- sestavení a zapojení CT, včetně realizace kabeláže a jednotlivých propojení
- sjednocení pinoutu všech připojených komponent a vytvoření univerzálních propojovacích kabelů
- uvedení CT do zkušebního i finálního provozu
- vytvoření vlastních konfiguračních souborů pro řídící karty
- vytvoření návodu k vytváření uživatelských konfiguračních souborů
- sepsání návodu ke kompilaci konfiguračních souborů pro řídicí karty

Jedním z prvních kroků je funkční analýza konfiguračních souborů integrovaných v balíčku HostMot2 a analýza dostupných materiálů, zabývajících se touto problematikou. Jak již bylo zmíněno, přestože je HostMot2 open source projekt, neobsahuje dostatečnou dokumentaci pro návrh uživatelských konfigurací. Dostupné jsou pouze předvytvořené konfigurace, k jejichž úpravám je potřeba značná expertní znalost problematiky. Z tohoto důvodu je nezbytné provést funkční analýzu jednotlivých částí kódu již existujících a dostupných konfigurací, doplněnou konzultacemi s odborníkem v dané problematice.

Další fáze spočívá v sestavení a zapojení sestavy CT, což obnáší vytvoření kabeláže, která napřímo propojí koncová zařízení s řídicími kartami. Tento druh zapojení je určen pouze pro vývoj konfiguračního souboru a testování jeho funkčnosti. Během prvotního zapojení je možné dočasně připojit i další zařízení, na nichž může být testována funkčnost konfigurace pro budoucí aplikaci na jiném experimentálním zařízení.

Hlavním výstupem práce je sestavení finálního kompletního konfiguračního souboru, jenž bude navržen dle požadavků na řízení všech použitých kocových zařízení. Řešení této konfigurace je důležitým vstupem pro návrh interního zapojení finální řídicí jednotky, jenž systematicky integruje všechny komponenty prořízení systému včetně propojovacích prvků.

Dalším výstupem práce je sepsání sady instrukcí k vytvoření uživatelských konfiguračních souborů, včerně návodu na jejich následnou kompilaci.

3. Metody

Kapitola poskytuje úvod do problematiky technického vybavení a software používaného pro řízení experimentálních zařízení. V úvodu jsou popsána programovatelná hradlová pole, která jsou v této bakalářské práci použita jako prvek logiky na řídicích kartách, který umožňuje tento systém konfigurovat a řídit dle vlastních požadavků. Na programovatelná hradlová pole navazuje sekce zaměřená na teoretické seznámení s principy programovacího jazyka VHDL, ve kterém je psán firmware, který konfiguruje řídicí Mesa kartu. Třetí část kapitoly je zaměřena na bližší seznámení čtenáře s kartami společnosti Mesa Electronic, které jsou použity jako rozšiřující periferie pro řídicí počítač. Poslední sekce této kapitoly je věnována popisu LinuxCNC, který je použit jako hlavní softwarový nástroj pro řízení polohování celého zařízení a popisu software Xilinx ISE Design Suite - ten je používán ke kompilaci konfiguračních souborů.

3.1 Programovatelné hradlové pole

První část této sekce je zaměřena na seznámení s obecnými informacemi o hradlovém poli a výhodami jeho použití. Další část práce popisuje obecnou architekturu hradlového pole a dvě technologie, použité na jejich konfigurování. Tento popis hradlového pole je ve třetí části práce specifikován na popis konkrétní rodiny hradlového pole (Xilinx Spartan 3), jež je hlavním logickým prvkem řídicí karty experimentálního zařízení.

3.1.1 Základní popis hradlových polí

Programovatelná hradlová pole jsou integrované číslicové obvody navržené tak, aby mohly být opakovaně nakonfigurovány dle požadavků uživatele. Tato technologie vychází z programovatelných logických obvodů (PLD), které se začaly používat na počátku 70. let. V posledních třech desetiletích prošla FPGA velkým vývojem a vsoučasné době jsou nejrozšířenější architekturou programovatelných číslicových obvodů [4].

FPGA jsou dvoudimenzionální pole tvořena pravidelnou strukturou programovatelných logických bloků, propojovacích sítí a I/O buněk. Hlavní výhodou FPGA je vysoká přizpůsobitelnost, díky které mohou být úlohy optimalizovány a zpracovávány rychleji než procesorem [5, 6].

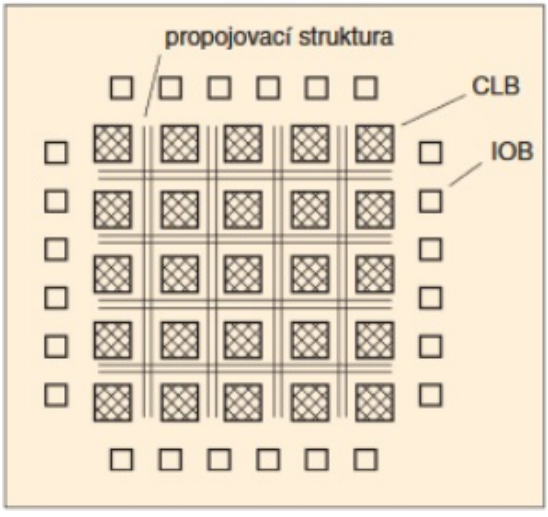
3.1.2 Obecná architektura FPGA

Logické buňky Základními prvky logických bloků jsou logické buňky. Tyto buňky pracují s jednoduchými funkcemi. Obecně jsou tvořeny konfigurovatelnými kombinačními logickými obvody a klopnými obvody. Nejčastěji se k implementaci konfigurace kombinačních logických obvodů používá metoda Look Up Table (vyhledávací tabulky) [5, 6].

Logické bloky Logické buňky se v obvodech FPGA obvykle sdružují do složitějších útvarů, které se nazývají logické bloky. Tyto bloky integrují funkce logických buněk, a tím poskytují logiku k řešení složitějších matematických a řídicích operací[5, 6].

I/O bloky I/O bloky jsou komponenty, které slouží jako rozhraní mezi logickými bloky a externě řízenými zařízeními. Fungují na principu převádění analogových hodnot na digitální a obráceně. Tyto bloky jsou technologicky nezávislé na budoucím použití [4].

Propojovací bloky Propojovací bloky obsahují vodiče, které lze variabilně propojit pomocí hardwarových přepínačů. Tyto konfigurovatelné přepínače se využívají k propojení jednotlivých logických buněk mezi sebou dle vnitřní struktury logických bloků, k propojení jednotlivých logických bloků mezi sebou a také k propojení logických bloků i s I/O bloky [5, 6].



Obrázek 3.1: Schéma obecné struktury FPGA [5]

3.1.3 Konfigurace

FPGA s volatilní konfigurací

Konfigurační data jsou ukládána do paměti typu SRAM (Static Random Access Memory). Tento typ paměti funguje na základě distribuce dat do paměti tvořené logickými buňkami, nebo do blokové paměti RAM, která je součástí struktury programovatelného hradlového pole. To znamená, že konfigurace není uložena na vedlejším médiu. Díky tomu jsou hradlová pole s nevolatilní konfigurací jednodušší, technologicky pokročilejší a umožňují zpracování úloh v kratších časových intervalech. Další výhodou je možnost snadné změny konfigurace, a to i během chodu systému [7].

Ukládání konfigurace do paměti SRAM s sebou však přináší i nutnost konfigurace při každém startu systému, neboť dojde k smazání původních dat. Konfigurační data musí být uložena mimo FPGA na externích paměťových nosičích. K datům v externí paměti FPGA přistupuje pouze při startu, když jsou nahrána do jeho vnitřní paměti. [7]

Další nevýhodou je nízké zabezpečení intelektuálního vlastnictví, protože do SRAM paměti je možno za chodu volně přistupovat a modifikovat její obsah, a slabá odolnost vůči ionizačnímu záření, při kterém může dojít k samovolné změně konfigurace. [7]

FPGA s nevolatilní konfigurací

Konfigurační data jsou uložena do nevolatilní paměti, která je součástí hradlového pole. Narozdíl od programovatelných hradlových polí s volatilní kofigurací jsou konfigurační data permanentně uložena v paměti a nemusí se opakovaně nahrávat při každém startu [7, 8].

Technologie používané k uložení konfigurace je možné rozdělit na 2 skupiny:

- Programovatelná hradlová pole s možností změny konfigurace - využívající elektricky smazatelné paměti například paměť typu EEPROM (Electrically Erasable Programmable Read-Only Memory) nebo flash [7, 8].
- Hradlová pole, která lze nakonfigurovat pouze jednou a další jejich změna není možná. Tato pole využívá technologii antipojistek, u které dochází k trvalému zapsání do struktury FPGA [7, 8].

Výhody tohoto typu konfigurace jsou značně vyšší odolnost vůči neautorizovanému přístupu,

ionizujícímu záření, a také nižší spotřeba energie. Tento typ je však konstrukčně náročnější a u FPGA méně častý [7].

3.1.4 Xilinx Spartan 3

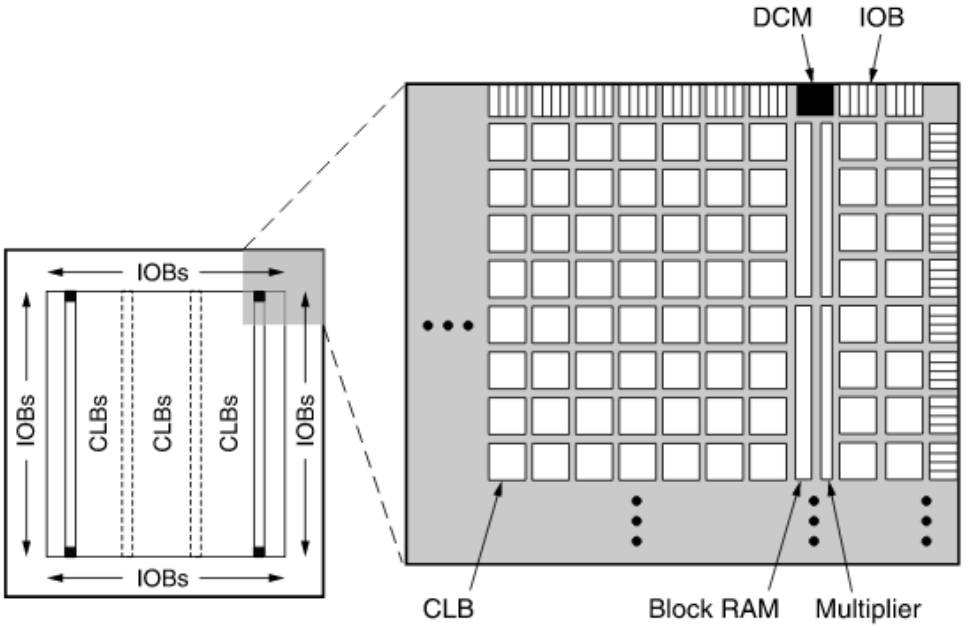
Spartan 3 je rodina hradlových polí, využitá při vývoji řízení polohovacího systému jednotlivých os tomografu. Jsou navrženy univerzálně - s ohledem na široké spektrum využitelnosti - při zachování nízkých pořizovacích nákladů.

Architektura Xilinx Spartan 3

Základní architektura Spartan 3 FPGA se skládá z následujících částí:

- Konfigurovatelné logické bloky (CLB) - Konfigurovatelné logické bloky jsou hlavním prvkem architektury vykonávající logiku hradlového pole [9].
- Input/Output Bloky (IOB) - Úkolem I/O bloků je zajištění obousměrného přenosu dat v rámci FPGA. Tyto bloky jsou bez konfigurace technologicky nezávislé a podporují široké spektrum přenosových standardů. Tyto standardy jsou specifikovány během konfigurace [9].
- Block RAM - Paměť RAM je tvořena 18kilobitovými bloky RAM, jež jsou v rámci architektury uspořádány do sloupců, umístěných po okrajích pole. Počet bloků tvořící sloupce a počet sloupců je závislý na konkrétní řadě hradlového pole. Tento prvek byl u FPGA firmy Xilinx poprvé použit u řady Spartan 3 [10].
- Dedicated Multipliers - násobiče obsahují prostředky pro násobení 18bitových signed a unsigned čísel. Pro lepší funkci násobičů jsou v rámci architektury pole umístěny do blízkosti blokové paměti RAM, která umožňuje ukládat vstupy a výstupy [9].
- Digital Clock Manager (DCM) - DCM jsou obvody řídicí hodinové signály. Umožňují generování nového signálu, jejich násobení a syntézu, eliminaci zpoždění nebo třeba změnu fázového posunu [11].

Rodina Spartan 3 obsahuje bohatou síť tras a přepínačů, které propojují všech pět funkčních prvků a přenášejí signály mezi nimi [9].



Obrázek 3.2: Schéma hradlového pole z rodiny Spartan 3 [9]

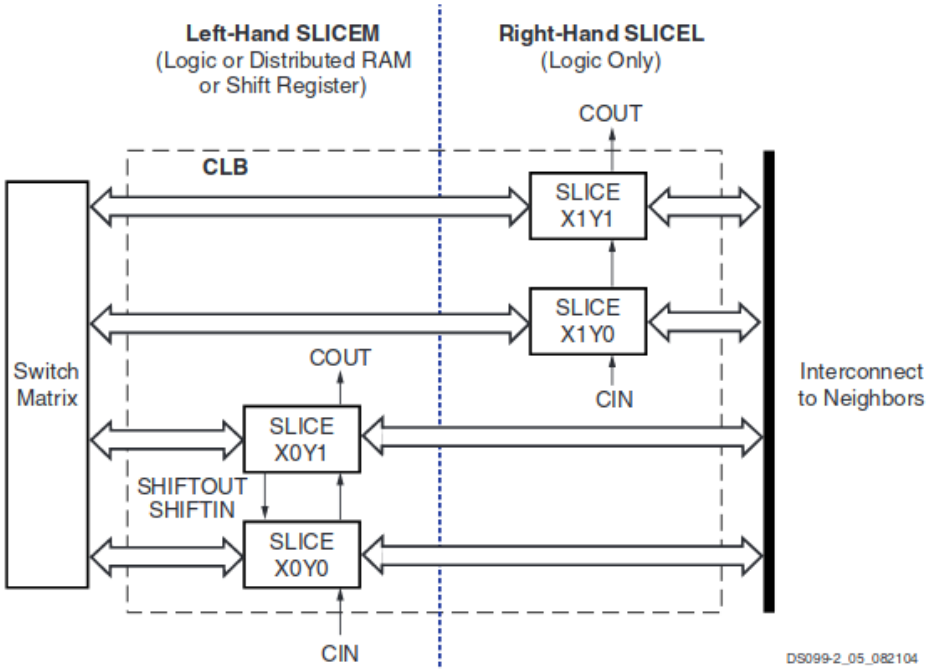
Look Up Tables Základním prvkem architektury Spartan 3 rodiny jsou LUT. Jedná se o kombinační obvod se čtyřmi vstupy a jedním výstupem. LUT jsou hlavním prostředkem pro implementaci logických funkcí a princip jejich fungování je založen na teorii Booleovy algebry, která zobecňuje pravidla pro operace s logickými funkcemi. Funkce s více vstupy jsou tvořeny kaskádovitým propojením několika LUT [9, 12].

Slice Slice je tvořen dvěma LUT, zastávajícími kombinační funkci, a dvěma klopnými obvody zastávající funkce sekvenční. Ve slices typu SLICEM mohou LUT použity jako 16x1bit RAM, nebo jako 16bitový posuvný registr. Klopné obvody realizují logické funkce tak, že každé kombinaci hodnot vstupních proměnných přiřadí určitou hodnotu výstupní proměnné [9].

Konfigurovatelné logické bloky Konfigurovatelné logické bloky jsou hlavním prvkem architektury FPGA poskytující implementaci sekvenčních i kombinačních obvodů. CLB jsou ve struktuře hradlových polí uspořádány do matice [9].

Každý konfigurovatelný logický blok obsahuje 4 tzv. slices, které jsou rozděleny do 2 párů podle své funkce. Tyto páry se nazývají SLICEL a SLICEM. [9]

- SLICEL může provádět pouze kombinatorické funkce [9]
- SLICEM může být nakonfigurován pro implementaci paměťových registrů [9]



Obrázek 3.3: Schéma rozdělení slices v CLB [9]

Konfigurace

Konfigurace je základní proces načítání dat do vnitřní paměti zařízení, která slouží k nastavení přepínačů, propojek, proměnných a parametrů. Rodina Spartan 3 (vyjma Spartan 3 AN) je tvořena FPGA s volatilní konfigurací, což znamená, že konfigurační data jsou uložena v konfiguračních západkách CMOS. Z toho plyne, nutnost konfigurace pole při každém zapnutí. Konfigurační data mohou být uložena na externím paměťovém nosiči, který může být umístěn na desce nebo mimo ní, nebo mohou být načtena pomocí externího zařízení, jako je například mikroprocesor, mikrokontrolér či počítač. K načtení těchto dat do zařízení jsou používány speciální konfigurační piny [6, 9].

3.2 Programovací jazyk VHDL

Tato sekce je zaměřena na stručné seznámenní s jazykem VHDL a úvod do architektury tohoto jazyka. Další část práce se zaměřuje na základní popis důležitých knihoven a balíčku Hostmot2, který je používán ke konfiguraci řídicích karet používaných v rámci této bakalářské práce.

3.2.1 Úvod do jazyka VHDL

VHDL je programovací jazyk používající se k popisu digitálních obvodů od jednoduchých logických bran až po kompletní mikroprocesory a vlastní čipy. Název vznikl kombinací dvou zkratek VHSIC (Very High Speed Integrated Circuit) a HDL (Hardware Description Language). V roce 1987 byl jazyk VHDL definován jako standardní jazyk popisu hardwaru americkým ministerstvem obrany a IEEE [13, 14].

VHDL je programovací jazyk, který je case insensitive - to znamená, že textový řetězec „VHDL“ je pro interpreter totožný s textovým řetězcem vhdl. Další charakteristickou vlastností je free-form, což umožňuje vkládat mezery a odsazení bez vlivu na funkčnost kódu. Jedná se o silně typovaný jazyk, tedy všechny objekty musí mít definované datové typy [6, 15].

Na základě VHDL kódu se při syntéze vygeneruje soubor popisující pole na úrovní jednotlivých hradel, z tohoto důvodu je nutné, aby syntézu prováděl program určený pro daný typ hradlového pole. Díky tomu je možné využít jeden programovací jazyk na velkém množství rozlišných platforem. Mimo programovatelných hradlových polí může být využit k návrhu obvodů pro technologii programovatelných logických obvodů (CPLD) nebo pro technologii integrovaných obvodů pro specifické použití (ASIC) [16].

3.2.2 Architektura

Systém návrhu hradlového pole je založen na skládání menších subsystémů pro vytvoření funkčního celku. Tento princip se nazývá strukturované programování [15].

K popisu funkce se využívají moduly, které se skládají z páru entita a architektura. Entita slouží k identifikaci komponenty a popisu jejího rozhraní. Deklaruje název příslušného obvodu, jeho popis, specifikace všech portů, vstupních a výstupních signálů. Architektura popisuje logické chování obvodu, definuje konstanty a interní signály. Při konfiguraci dochází k spárování architektury a příslušnou entitou.

3.2.3 Knihovny a balíčky

Knihovny jsou soubory obsahující předpřipravené funkce a procedury, sloužící například k definování entit, architektury, nebo balíčků. Samotný jazyk VHDL obsahuje dvě základní knihovny STD a WORK. Další důležitou knihovnou je IEEE.

Balíčky slouží k rozšíření a doplnění funkcí knihoven. Ve VHDL se často používají jen v kombinaci s unikátní knihovnou.

Knihovna STD

Knihovna STD je výchozí součástí jazyka VHDL. To znamená, že není nutné ji v kódu inicializovat. Obsahuje dva balíčky TEXTIO a STANDARD. Balíček STANDARD definuje základní funkce, datové typy a příslušné operátory. Balíček TEXTIO obsahuje funkce umožňující práci s textem [17].

Knihovna work

Nejedná se o knihovnu v pravém slova smyslu. Jedná se spíše o označení pracovního místa, ve kterém jsou uloženy všechny potřebné zdrojové kódy VHDL ke kompilaci [6].

Balíček Hostmot2 je při otevření projektu příslušné karty automaticky nahrán do knihovny work. Nově vytvořený konfigurační soubor však součástí původních dat Hostmot2 není, takže musí být přidán dodatečně.

Knihovna IEEE

Knihovna IEEE není součástí jazyka VHDL, takže je nezbytné ji inicializovat. Přesto však musí být součástí všech zdrojových kódů, protože slouží k rozšíření datových typů, funkcí a logiky. Knihovna IEEE patří standardu IEEE 1164, určenému k elektronické automatizaci návrhu. Z tohoto důvodu lze rozšiřující balíčky rozdělit na dvě skupiny: standardizované a průmyslově zavedené (nestandardizované). Příkladem nestandardizovaného balíčku může být například HostMot2 [18, 17].

3.2.4 HostMot2

HostMot2 je driver a zároveň balíček pro FPGA karty společnosti Mesa electronics, sloužící k řízení a konfiguraci karet řady Anything I/O FPGA. Jedná se o open source projekt psaný v jazyce VHDL. HostMot2 poskytuje moduly obsahující funkce a procedury k ovládání enkoderů, krokových motorů, pulzně šířkové modulace a dalších I/O zařízení. Součástí je také balíček pro Hardware Abstraction Layer, používaný k abstraktizaci karet Mesa Electronics Anything I/O FPGA cards [19, 20].

3.3 Řídicí karty

Sekce Řídicí karty je zaměřena na seznámení uživatele s kartami společnosti Mesa Electronic, ty jsou použity jako rozšiřující periferie pro řídicí počítač. Tyto karty umožňují komunikovat se všemi připojenými zařízeními. Jsou osazeny hradlovým polem,použitým pro implementaci logiky při řízení této periferie.

3.3.1 Mesa Electronics

Na základě kladných předchozích zkušeností s kartami společnosti Mesa Electronics na oddělení biomechaniky ÚTAM AV ČR byla k řízení polohování tomografu zvolena kombinace těchto karet, jež umožňují pokročilé řízení krokových motorů, servomotorů, enkodérů a dalších zařízeních používaných u CNC přístrojů. Princip Mesa karet se zakládá na kombinaci jedné hlavní konfigurovatelné řídicí karty osazené programovatelným hradlovým polem. Řídící karta obsahuje konektory s nezávislými I/O piny k nimž jsou připojeny rozšiřující karty. Typoskytují potřebné rozhraní pro komunikaci se všemi komponentami systému.

Řídicí karta

Mesa karty řady Anything I/O FPGA cards jsou osazeny programovatelným hradlovým polem rodin Spartan 3 a Spartan 6. Rodiny Spartan se vyznačují tím, že se jedná o karty s volatilní konfigurací. Řídicí karta je k počítači připojena pomocí sběrnic PCI nebo PCI express, umožňující paralelní přenos dat.

3.4 LinuxCNC

Požadavky na řízení polohování laboratorního tomografu jsou podobné jako požadavky na řízení průmyslových CNC strojů. Na základě zkušeností s ovládáním experimentálních zařízení zkonstruovaných na oddělení biomechaniky byl zvolen LinuxCNC i k řízení tohoto nově vzniklého experimentálního zařízení.

LinuxCNC je open source software navržený k ovládání servomotorů, krokových motorů, relé a dalších zařízení souvisejících s automatizací. Hlavní výhodou tohoto systému je možnost přizpůsobení uživatelského prostředí a funkcí dle vlastních funkčních požadavků. Na oddělení biomechaniky je vyvíjena jeho nadstavba využita k řízení exerimentálního zařízení.

3.5 Xilinx ISE Design Suite

ISE Design Suite je software vyvinutý společností Xilinx, sloužící k návrhu a syntéze programovatelných hradlových polí. Po registraci na oficiálních stránkách je pro uživatele poskytován zdarma [21].

V říjnu 2013 byla vydána podlední verze ISE Design Suite (verze 14.7) a další již nejsou plánovány. Stávající verze ISE podporuje zařízení rodin Spartan 6, Virtex 6 a Coolrunner a jejich předchozí generace. Pro novější rodiny byl vytvořen nový software Vivado Design Suite [21].

3.5.1 Práce v ISE Design Suite

Navrhování konfiguračních souborů pro hradlová pole v programu ISE Design Suite je provedeno prostřednictvím projektu, který obsahuje všechny soubory týkající se návrhu. Samotný návrh může být vytvořen dvěma způsoby:

- vytvořením zdrojových kódů pomocí hardwarových jazyků: VHDL, Verilog, ABEL
- schematicky

Po vytvoření kompletního návrhu umožňuje software následnou syntézou, během které dochází k vytvoření netlistu na základě návrhu. Netlist je textový popis obvodu, součástek obvodu a jejich propojení, využívaný jako vstup pro nástoje určené k tvorbě rozvržení čipu [22, 23].

Následuje proces implementace, při které dochází k převedení logického návrhu na fyzický formát

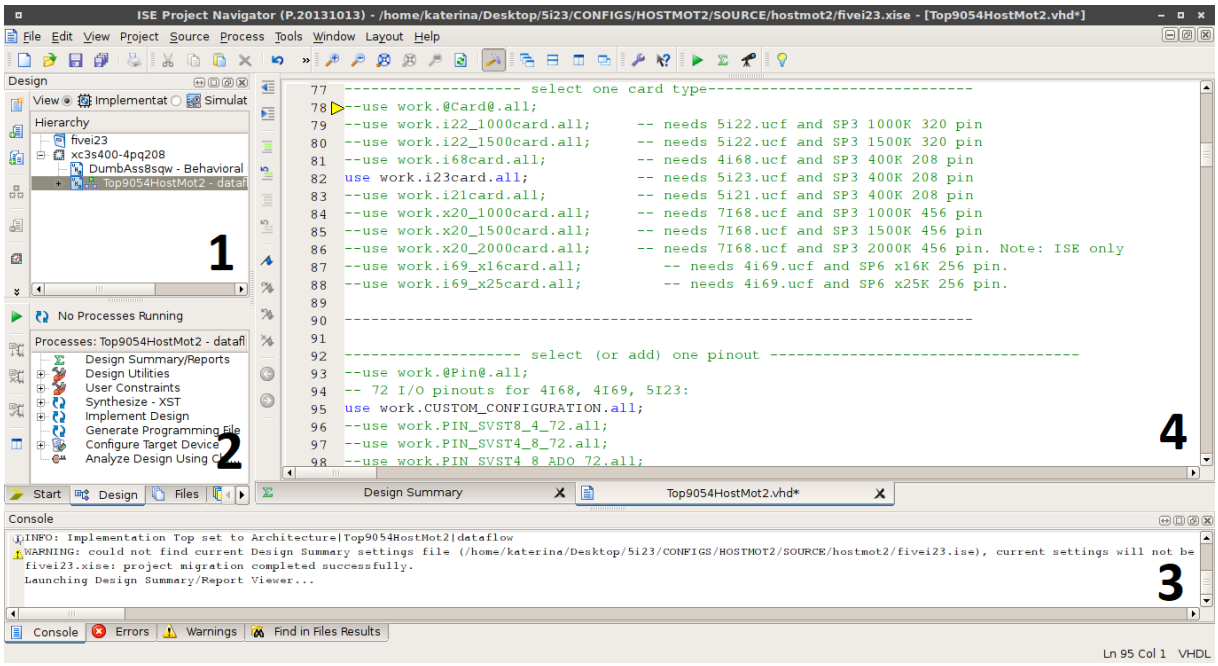
souboru. V aplikaci Project Navigator lze spustit proces implementace v jednom kroku, nebo jednotlivé implementační procesy spouštět samostatně [22].

Posledním krokem k vytvoření finálního souboru pro konfiguraci cílových zařízení je generování bitfile, při čemž dochází k vytvoření souboru s formátem .bit, který může být nahrán do příslušné paměti hradlového pole.

3.5.2 Uživatelské rozhraní

Výchozí okno uživatelského rozhraní softwaru ISE Design Suite se skládá z několika pracovních oken:

- okno se zdroji - hierarchicky zobrazuje soubory obsažené v projektu
- okno s procesy - zobrazuje list procesů, které je aktuálně v projektu možné spustit
- konzole - zobrazuje stav systému, probíhající procesy, hlášení a chyby
- hlavní pracovní okno - slouží k editování souborů a zobrazení Design summary



Obrázek 3.4: Označení jednotlivých oken v programu ISE Design Suite určených pro práci a syntézu.
1 - okno se zdroji, 2 - okno s procesy, 3 - konzole, 4 - hlavní pracovní okno

4. Návrh a kompilace souboru pro Mesa FPGA I/O cards

4.1 Návrh konfiguračního souboru

První část kapitoly popisuje tvorbu konfiguračního souboru dle vlastních požadavků. Při návrhu je třeba vycházet z výstupního rozhraní rozšiřujících karet, které musí odpovídat konfigurovaným funkcím. Konfigurační soubory mohou být vytvořeny použitím jednotlivých příkazů popsaných v této kapitole, nebo vyplněním připravené šablony v příloze C.

Konfigurační soubory se skládají ze 4 základních částí:

- inicializace knihoven
- vytvoření prostředí pro inicializaci modulů a pinů
- inicializace modulů
- definice jednotlivých pinů

4.1.1 Deklarace knihoven a balíčků

V první části souboru je třeba deklarovat knihovny a balíčky. Popis základních knihoven jazyka VHDL se nachází v podkapitole 3.2.3.

Následující řádky uvádí inicializaci knihoven, které mohou být vloženy do prvních řádků konfiguračního souboru.

```
library IEEE;
```

IEEE je základní knihovna jazyka VHDL.

```
use IEEE.std_logic_1164.all;
```

Balíček `IEEE.std_logic_1164` je určený k rozšíření logických hodnot knihovny IEEE. Obsahuje definice datových typů, podtypů a funkcí, které rozšiřují VHDL logiku.

```
use IEEE.STD\_LOGIC\_ARITH.ALL;
use IEEE.STD\_LOGIC\_UNSIGNED.ALL;
```

Tyto balíčky poskytující pokročilé funkce pro numerické výpočty.

```
use work.IDROMConst.all;
```

Balíček IDROMConstant umožňuje využívat funkce HostMot2.

4.1.2 Inicializační prostředí

Moduly a funkce pro piny se inicializují v prostředí balíčku, který je vytvořen a specifikován v konfiguračním souboru.

Začátek inicializačního prostředí

Začátek inicializačního prostředí je uveden vytvořením nového balíčku, jehož název odpovídá názvu konfiguračního souboru.

```
package NAZEV-SOUBORU is
```

Inicializace modulů

V druhé části kódu jsou inicializovány moduly. Tyto obsahují funkce a procedury, které implementují ovládání řídicích prvků hardwaru.

Definice každého modulu v konfiguračním VHDL souboru se skládá z několika částí, které jsou odděleny čárkou.

Příklad inicializace modulu

Narozdíl od knihoven jsou moduly inicializovány ve speciálním prostředí, které se vytvoří pomocí příkazu:

```
constant ModuleID : moduleIDType :=(
);
```

Do tohoto prostředí se vkládají řádky kódu inicializující jednotlivé moduly. V každém návrhu je třeba inicializovat právě 32 modulů. Tento příklad je znázorněn a popsán na modulu Smart Serial:

```
(SSerialTag, x"00", ClockLowTag, x"02", SSerialCommandAddr & PadT,
SSerialNumRegs, x"10", SSerialMPBitMask),
```

Popis jednotlivých prvků kódu:

1. GTag `( SSerialTag)` - název modulu
2. Version `(x"00")` - verze modulu
3. ClockTag `(ClockLowTag)` - typ clock signálu
4. Instances `(x"02")` - počet instancí
5. BaseAddr `(SSerialCommandAddr&PadT)`
6. NumRegisters `(SSerialNumRegs)` - počet registrů příslušných modulu
7. Strides `(x"10")`
8. MultRegs `(SSerialMPBitMask)`

Při vkládání těchto řádků je třeba brát zřetel hlavně na definici počtu instancí modulu. Ten by měl odpovídat počtu zařízení využívající tento funkční modul.

4.1.3 Piny

V části inicializace pinů jsou aplikovány jednotlivé funkční prvky modulů na konkrétní I/O piny řídicí karty. Tyto funkce jsou následně přenášeny do rozšiřujících karet, obsahující periferie pro

přenos signálů na koncová zařízení. Při návrhu pinoutu je třeba brát v potaz typ rozhraní a směr přenosu signálu. Ty jsou přiřazeny hardwarovýn pinům v pořadí, ve kterém jsou inicializovány.

Stejně jako moduly jsou i funkce pinů inicializovány ve separátním bloku kódu:

```
constant PinDesc : PinDescType :=(
);
```

Příklad inicializace pinů je znázorněn na příkladě funkce StepGenDirPin modulu StepGenTag. Jednotlivé parametry odděluje symbol "&". Počet inicializovaných pinů závisí napočtu I/O pinů na konfigurované řídicí kartě.

```
IOPortTag & x"00" & StepGenTag & StepGenDirPin,
```

1. `IOPortTag` - syntaxe udávající, že se jedná o I/O port
2. `x"00"` - indexování instance modulu
3. `StepGenTag` - název modulu
4. `StepGenDirPin` - funce pinu

Při návrhu je třeba dbát na indexování instancí modulu, kdy jedno zařízení musí být připojeno k pinům modulu které jsou označeny stejným indexem.

Ukončení inicializačního prostředí

Následující blok kódu je nezbytnou součástí k ukončení inicializace pinů:

```
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
```

Tento blok je v kofiguračním souboru určeném pro kartu 5i23, nebo pro jiné FPGA karty obsahující právě 72 I/O pinů, vložený právě devětkrát.
V samotném konci souboru se celé inicializační prostředí uzavře následujícím řádkem:

```
end package NAZEV-SOUBORU;
```

4.2 Kompilace konfiguračního souboru

Sekce popisuje pracovní postup kompilace od přípravy pracovního prostředí po generování souborů ve formátu .bit určených pro nahrání do paměti hradlového pole. Kompilace se provádí v prostředí programu Xilinx ISE Design Suite.

Postup pro vytvoření finálního konfiguračního souboru se skládá z několika kroků:

1. stažení podpůrného software
2. příprava pracovního projektu
3. syntéza a implementace
4. generování bit souboru

4.2.1 Stažení podpůrného software

Podpůrné software nabízí zdarma ke stažení na svých webových stránkách společnost Mesa Electronics. Je nezbytné stáhnou soubor příslušný k danému typu Anything I/O FPGA card, ten se pro kartou 5i23 nazývá `5i23.zip`.

Po stažení následuje extrahování souboru `5i23.zip` a následné extrahování souboru `hostmot2.zip`, ten je uložený v archivu na cestě `/5i23/CONFIGS/HOSTMOT2/SOURCE`. Archiv `hostmot2.zip` nabízi již sadu předpřipravených konfiguračních soborů, jež mohou být rovněž využity. Uživatelem vytvořené konfigurační soubory musí být před zahájením kompilace nahrány do složky obsahující všechny extrahované soubory z archivu `hostmot2.zip`. V případě pracovního postupu použitého v této práci se nacházejí na cestě
`/5i23/CONFIGS/HOSTMOT2/SOURCE`.

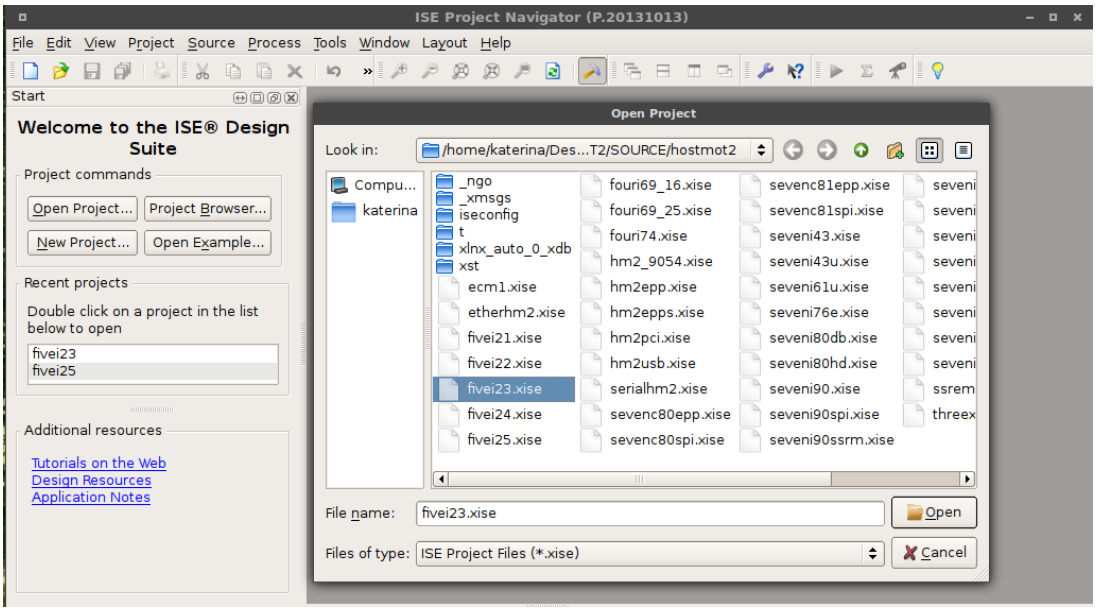
Výše popsaný proces je možné zpracovat v Linux příkazové řádce pomocí následujících příkazů. Před zahájením práce v příkazové řádce je důležité se v adresáři nacházet na místě na kterém je uložen soubor `5i23.zip`.

```
# lokalizace balíčku hostmot2
unzip 5i23.zip
cd 5i23/CONFIGS/HOSTMOT2/SOURCE
unzip hostmot2.zip
```

4.2.2 Příprava pracovního projektu

Procesy kompilace se provádí v prostředí programu Xilinx ISE Design studio. Jak bylo již popsáno v podkapitole 3.5.1, Xilinx ISE Design studio pracuje na bázi projektů, ve kterých jsou nahrány všechny soubory potřebné pro následnou syntézu a implementaci. Pro práci s FPGA I/PO cards jsou připraveny hotové projekty, které jsou součástí podpůrného software. Ty se společně s ostatními konfiguračními soubory nachází ve složce s exrahovanými soubory z archivu `5i23.zip`.

1. Po spuštění programu ISE Design Suite se projekt otevře v menu *File* příkazem *Open project*. Tento příkaz spustí nové dialogové okno, sloužící k vyhledávání projektu v adresáři počítače. Po otevření adresáře s extrahovanými soubory z archivu `hostmot2.zip` je třeba vybrat projekt pro příslušný řídící kartě (viz obrázek 4.1). Pro práci s kartou 5i23, byl zvolen projekt s názvem *fivei23.xise*.

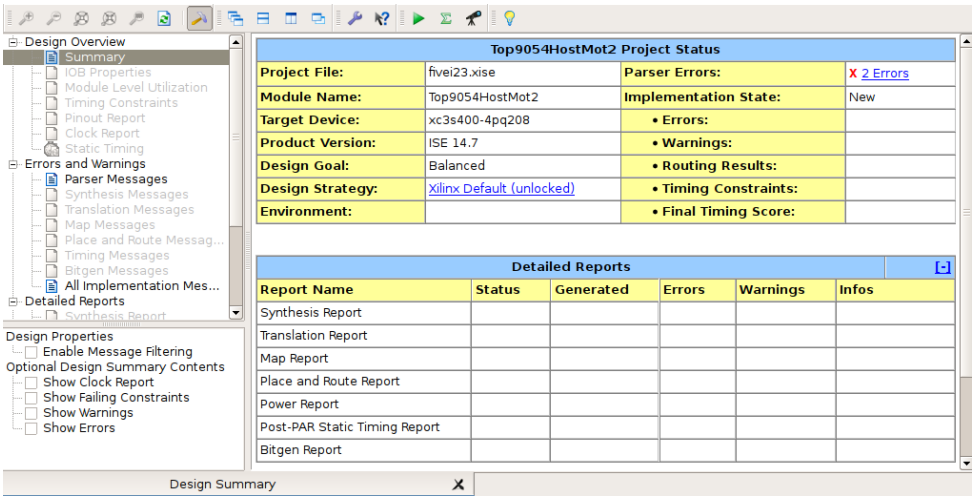


Obrázek 4.1: Screenshot programu ISE Design suite

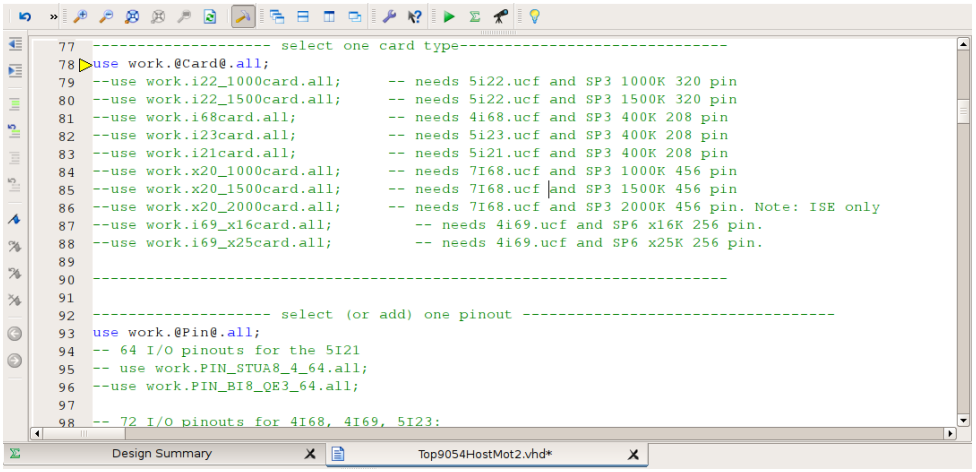
2. Po otevření projektu se v pracovním okně objeví tabulka s názvem *Design Summary*, znázorněna na obrázku 4.2. Tabulka obsahuje stručný přehled informací o projektu, včetně upozornění na dvě chyby. Tyto hlášení přímo odkazují na část souboru *Top9054HostMot2.vhd* ve kterém se volí typ motherboard a typ jejího konfiguračního souboru viz obrázek 4.3.

   Volba konfiguračního souboru spočívá v nahrazení řádků č. 78 (`use work.@Card@.all;`) a č. 93 (`use work.@Pin@.all;`) řádky s názvem karty tj `use work.i23card.all;` , resp.

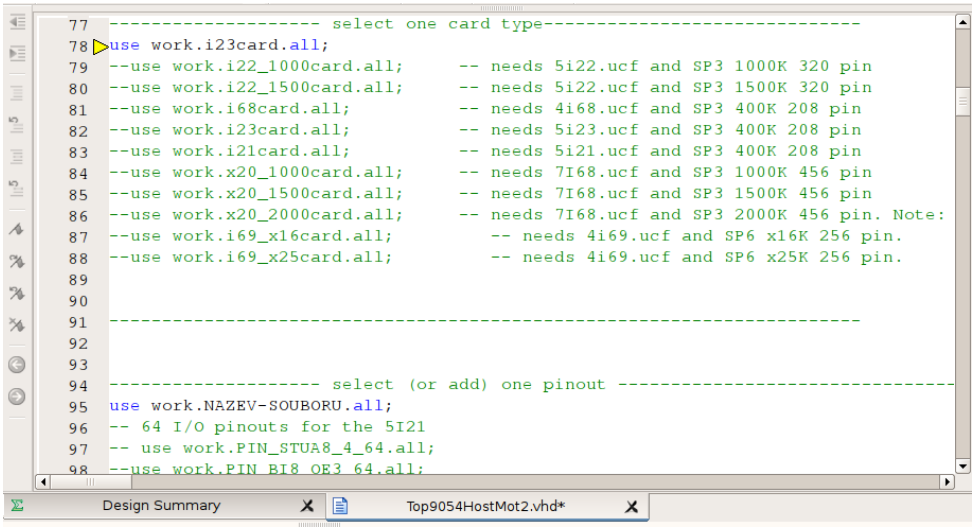
konfuguračního souboru tj. `use work.NAZEV-SOUBORU.all;`, jak je zobrazeno na obrázcích 4.3 a 4.4.



Obrázek 4.2: Pracovní okno zobrazující přehled projektu, který se zobrazí po otevření nového projektu včetně upozornění na 2 chyby



Obrázek 4.3: Původní chybové řádky 78 a 93 souboru TOP9054Hostmot2.vhdl



Obrázek 4.4: Řádky 78 a 93 souboru TOP9054Hostmot2.vhdl po volbě karty a konfiguračního souboru

3. Posledním krokem je práce s knihovnou work, jejíž popis se nachází v podkapitole 3.2.3. V rámci připravených projektů jsou pro kompilaci nezbytné soubory již součástí knihovny work. Při použití konfiguračního souboru, který není defaultně součástí podpůrného software, je nutné tento do knihovny dodatečně přidat.

   K přidání souborů do knihovny work slouží v menu *Project* příkaz *Add Source*, otevírající dialogové okno pro vyhledávání souboru v adresáři počítače. Při použití konfiguračního souboru, který není součástí knihovny, nemohou být provedeny následující kroky.

4.2.3 Syntéza a implementace

Principy syntézy a implementace VHDL kódu jsou stručně popsány v podkapitole 3.5.1.

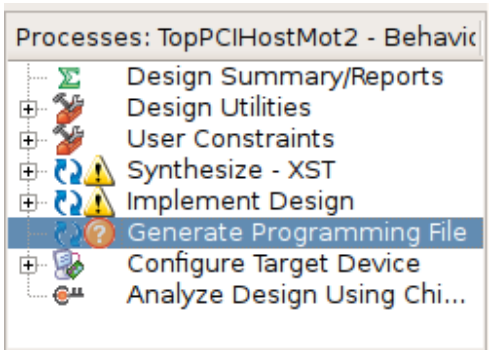
Syntézu i implementaci lze v prostředí Design ISE Suite provézt jedním příkazem pomocí menu *Process* příkazem *Implement Top module*, nebo postupně pomocí příkazů *Synthesize - XST* a *Implement Design* , které se nachází v okně s procesy.

4.2.4 Generování konfiguračního bit souboru

Posledním krokem v přípravě konfiguračního souboru ve formátu .bit určeného pro nahrání do paměti hradlového pole je jeho generování. Tento proces se spustí v okně s procesy, pomocí příkazu *Synthetize - XST* viz obrázek 4.5.

Finální konfigurační BIT soubor se nachází pod názvem `Top9054HostMot2.bit` na cestě, na které

jsou uloženy extrahované soubory z archivu `hostmot2.zip`.



Obrázek 4.5: Okno s procesy umožňující spuštění příkazu pro generování souboru ve formátu .bit

5. Realizace

Tato kapitola se zabývá technickým popisem komponent, se kterými se pracovalo v rámci této bakalářské práce. Dále popisuje vývoj realizace a zprovoznění zařízení od prvního zapojení několika komponent po kompletní zapojení a testovací provoz řídicí jednotky. Vzhledem k tomu, že se jedná o experimentální zařízení, je nezbytné pro jeho provoz používat řídicí jednotku s periferií umožňující řízení všech koncových zařízení.

5.1 Popis jednotlivých komponent

5.1.1 Lineární osy

Pro polohování byla zvolena kombinace několika druhů lineárních os litevské společnosti Standa, poháněných krokovými motory. Tyto motory jsou tvořeny rotorem - permanentními magnety a statorem - páry cívek . Po přivedení napětí na magnetické cívky dojde k otočení rotoru do nejbližší stabilní polohy. Souvislý pohyb motoru je zajišťěn přiváděním napětí na po sobě jdoucí cívky. Z toho vyplývá, že pohyb krokového motoru není spojitý, avšak umožňuje dosažení velké přesnosti polohování [24].

5.1.2 Driver

Drivery jsou zařízení používaná k řízení krokových motorů. Samotné řízení je realizováno prostřednictvím signálů STEP a DIR. Ty driver přijímá z řídicích karet a transformuje pro signály pro krokové motory. V tomto případě byly zvoleny digitální drivery leadshine DM422C, přizpůsobené k řízení dvoufázových a čtyřfázových krokových motorů [25].

Obrázek 5.1: Driver DM422C[25]

5.1.3 Enkodér

Magnetické enkodéry jsou jedním ze systémů používaných k přesnému měření polohy. Na tomto zařízení jsou použity inkrementální lineární enkodéry. Inkrementální znamená, že pracují s relativní polohou, kdy zaznamenávají přírůstek vzdálenosti od výchozího bodu na speciální magnetické pásce. Při realizaci CT byly zvoleny enkodéry a magnetické pásky značky Renishaw. Tato kombinace umožňuje měřit posun s rozlišením od 1 μm [26].

Obrázek 5.2: Inkrementální lineární enkodér LM13 [26]

5.1.4 Rotační stolek

Rotační stolek umožňuje otáčení umístěným objektem o 360 stupňů. V případě tohoto zařízení byl zvolen rotační stolek firmy Akribis řady DDR navržený pro přímé otáčení bez převodového mechanismu. Zařízení mimo jiné integruje i enkodér, což umožňuje dosáhnout rozlišení 183,552 CPR (counts per revolution). Dále bylo nezbytné využít driver společnosti Akribis určený k řízení rotačních stolů [27].

Obrázek 5.3: Rotační stolek Akribis ACD120 [27]

5.1.5 Mesa karty

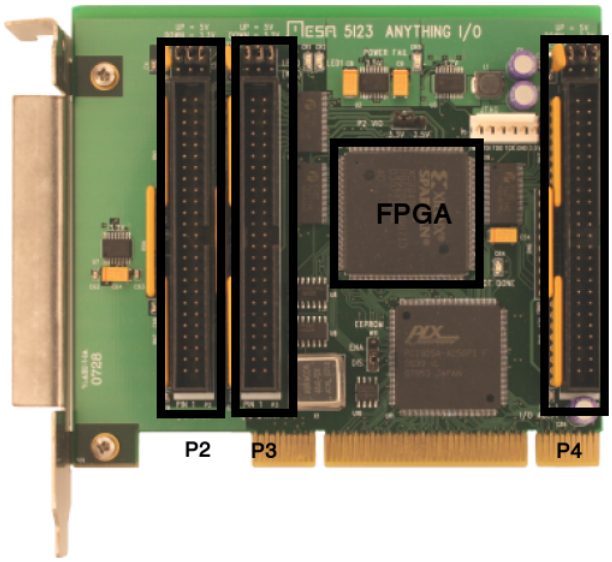
Jak již bylo zmíněno, na základě předchozích pozitivních zkušeností z dalších projektů realizovaných na oddělení biomechaniky byla k řízení polohování laboratorního tomografu zvolena karta 5i23 z řady Anything I/O FPGA cards a kombinace rozšiřujících karet 7i47, 7i52, 7i37TA. Tyto karty obsahují rozhraní jako jsou RS-422, multiplexované RS-422 a izolované I/O piny umožňující připojení všech komponent pro základní sestavu polohování a řízení rentgenky. Pro rozšířenou verzi zařízení je možné připojení druhé řídicí karty.

V příloze D je uvedeno schéma konfigurace a zapojení, v příloze B je kompletní konfigurační soubor a v příloze A se nachází kompletní seznam modulů použitých ke konfiguraci karet.

Řídicí karta 5i23

Tato karta disponuje hradlovým polem Spartan 3 XC3S400. Konfigurace je prováděna pomocí obvodu PLC9030 zajičťujícím přenos dat mezi FPGA a sběrnicí. Pro připojení rozšiřujících karet slouží tři 50pinové SCSI konektory s 24I/O piny, 24 GND piny a jeden 5 V napájecí pin. Dohromady tato karta poskyuje 72 nezávislých konfigurovatelných I/O pinů [28].

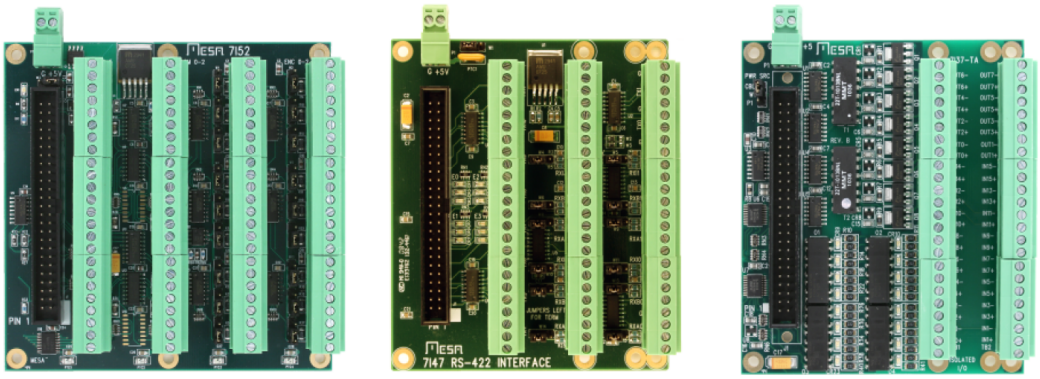
Karta 5i23 byla zvolena, protože disponuje dostatečným počtem konektorů (3x 50-pin) pro připojení všech potřebných rozšiřujících karet, zároveň disponuje vyspělejším čipem než ostatní karty v této kategorii.



Obrázek 5.4: Řídicí karta 5i23 [29]

Rozšiřující karty

Jako rozšiřující karty byly zvoleny 7i47, 7i52, 7i37TA plně pokrývající potřeby periferií pro připojení všech řízených komponent. Piny na kartě 7i52 jsou konfigurovány pro připojení šesti enkodérů. Karta 7i47 je přizpůsobena pro připojení 6 driverů určených k ovládání krokových motorů. Karta 7i37TA poskytuje nezávisle na konfiguraci sadu nezávislých I/O pinů, které mohou být využity například k připojení koncových spínačů, I/O pinů pro driver rotačního stolku a I/O pinů pro přenos dat mezi počítačem a rentgenkou.



Obrázek 5.5: Rozšiřující karty - zleva 7i52, 7i47, 7i37TA [29]

5.2 Fyzická realizace

Fyzické sestavení přístroje bylo provedeno rovněž v rámci praktické části této bakalářské práce. Konkrétně se jednalo o následující činnosti:

- sestavení a montáž zařízení
- zapojení všech komponent do karet
- realizace kabeláže pro zkušební zapojení
- návrh a osazení plošných spojů
- realizace kabeláže pro finální zapojení
- technická dokumentace konfigurace a zapojení

5.2.1 Pilotní zapojení a testovací provoz

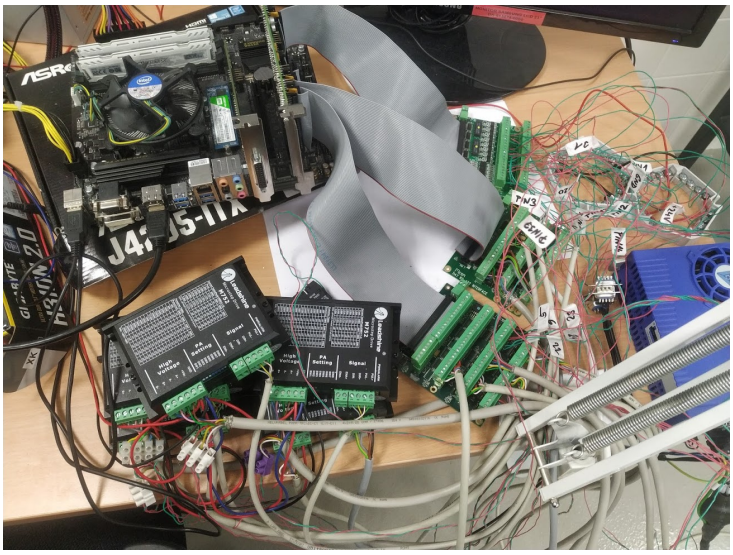
Po vytvoření prvního konfiguračního souboru proběhlo zkušební zapojení jednoho krokového motoru s koncovým spínačem a jednoho enkodéru pro ověření funkčnosti použitých modulů. Pilotní zapojení spočívalo ve vytvoření jednoho kabelu propojujícího I/O piny karet a konektory koncových zařízení.

Po otestování funkčnosti tohoto zapojení byla sestavena konfigurace pro všechna zařízení. I v tomto případě byla koncová zařízení propojena do I/O pinů rozšiřujících karet pomocí realizovaných kabelů. Tyto kabely musely být vytvořeny pro všechna zařízení, neboť kabeláž, která byla součástí zařízení, nesplňovala požadavky pro zapojení.

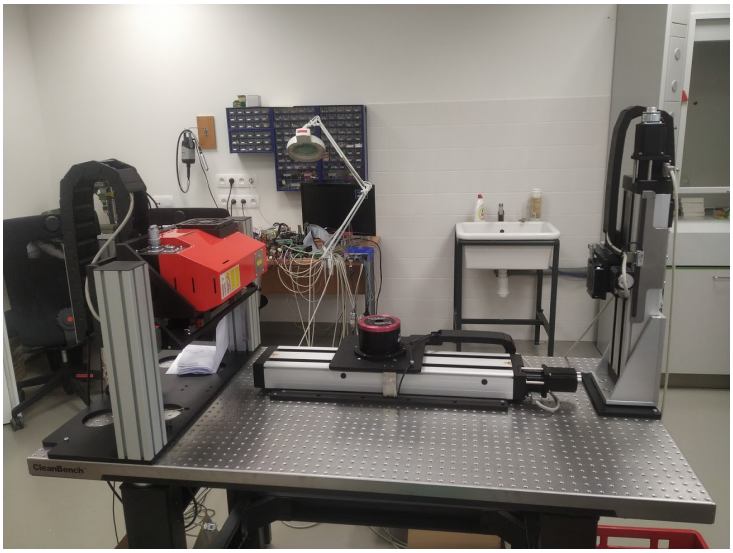
Po úspěšném ověření konfigurace byl tomograf sestaven a uveden do testovacího provozu - prozatím však bez dedikované řídicí jednotky. Během tohoto provozu bylo testováno připojení dalších zařízení využívajících různé moduly podporované HostMot2, ty mohou být použity v nově vzniklých zařízeních. Konkrétně se jedná o modul pro připojení enkodéru komunikující pomocí protokolu SSI - ten byl úspěšně zprovozněn - a modul pro protokol BISS-C, který se však nepodařilo zprovoznit z důvodu chybějící podpory v aktuální verzi LinuxCNC.

5.2.2 Finální zapojení

Po úspešném ukončení testovacího provozu byla navržena a sestavena řídicí jednotka integrující a propoující všechny prvky řídícího systému. Schéma finálního zapojení řídicí jednotky je

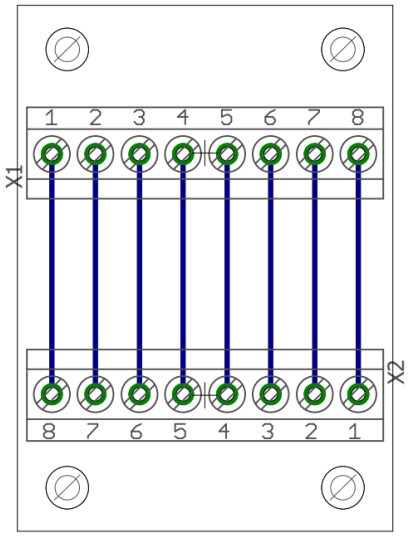


Obrázek 5.6: První zapojení všech komponent napřímo do řídicích karet

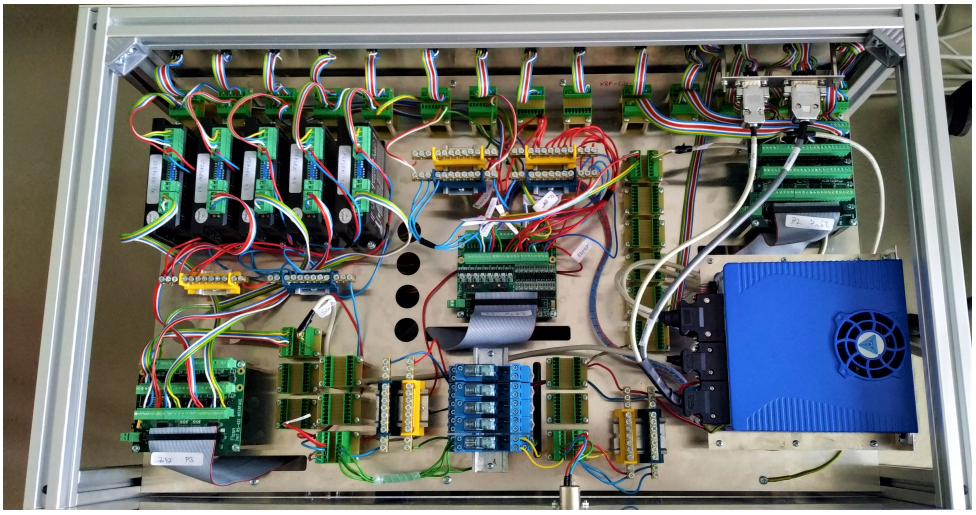


Obrázek 5.7: Sestavené zařízení se zapojením komponent napřímo do řídicích karet

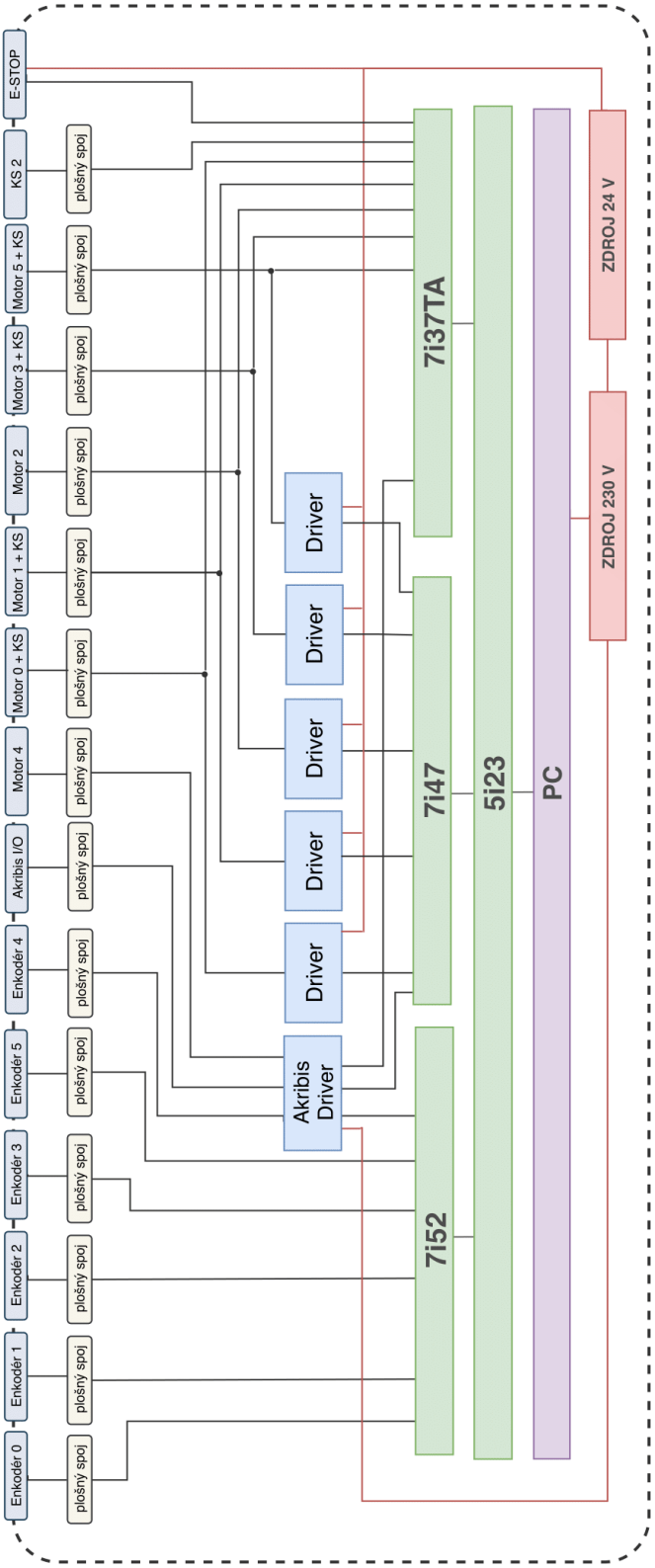
zobrzeno na obrázku 5.10, narozdíl od předchozích verzí bylo připojeno tlačítko pro nouzové zastavení všech motorů. Pro potřeby zařízení byl v rámci bakalářské práce vytvořen i jednoduchý plošný spoj, který je osazen dvojící násuvných konektorů. Ty zajišťují nepájené spojení vedených signálu, a tím umožňují jeho jednoduché rozpojení v případě potřeby.



Obrázek 5.8: Návrh vytvořeného plošného spoje



Obrázek 5.9: Fotografie části řídicí jednotky



Obrázek 5.10: Schéma zapojení řídící jednotky

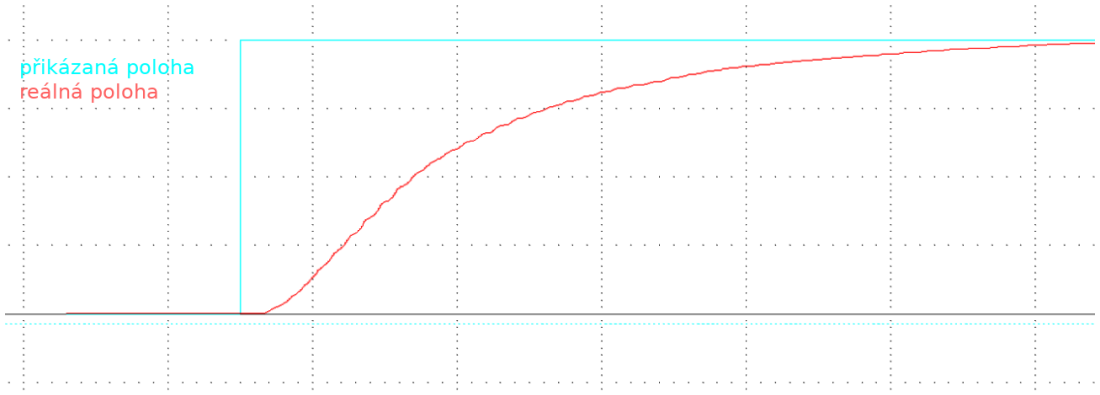
6. Referenční měření

Tato kapitola se zabývá popisem refenčního měření, které demonstruje funkci vytvořeného konfiguračního souboru. Funkčnost konfiguračního souboru je demonstrována na ose polohující rotačním stolkem, neboť na její přesnosti je závislá kvalita pořízených snímků. Dále je nezbytné v rámci konfigurace synchronizovat přenos dat pro rotační stolek, nebooť přesnost jeho polohování je rovněž klíčová pro kvalitu výstupu tomografie. Kompletní tomografické měření nemohlo být uskutečněno, neboť zprovoznění radiologických zařízení musí být nejprve schváleno Státním úřadem pro jadernou bezpečnost. Z tohoto důvodu bylo provedeno pouze měření demonstrující funkčnost polohování.

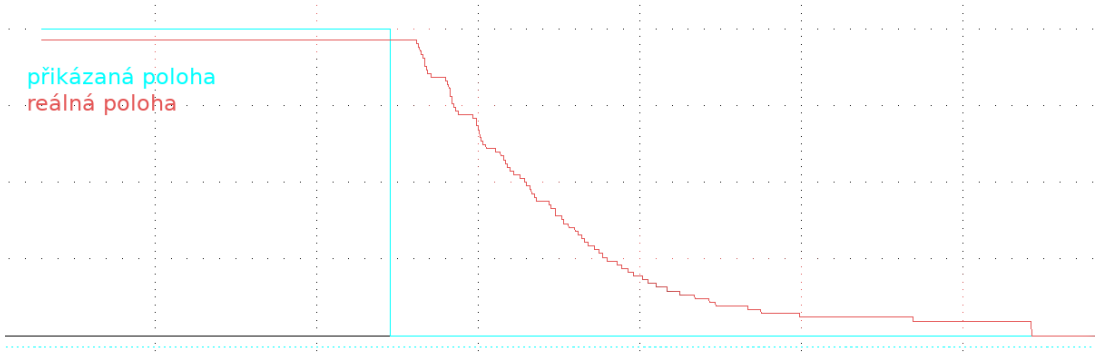
Přesnost ustavení polohy je mimo jiné dána vlastnostmi použitých zařízení. V případě os se jedná primárně o stoupání pohybového šroubu v kombinaci s použitým krokovým motorem. V případě enkodérů je přesnost měření dána typem použitého zařízení. Ani u zpětnovazebného řízení není možné dosáhnout shody přikazované pozice a reálné pozice vyčtené enkodérem. Z tohoto důvodu se před měřením stanovuje tolerance, se kterou bude polohování provedeno. V případě CT se jedná o toleranci $1\mu m$.

Testování pohybu lineární osy je demonstrováno časovým vývojem dat polohy snímané enkodérem. K provedení testu byla zvolena horizontální osa pohybující s rotačním stolkem. Měření bylo provedeno pouze na této ose, neboť výstupní data jsou ze všech os podobná. Výstupem měření jsou grafy vytvořené v prostředí Halscope, což je jeden z výchozích nástrojů LinuxCNC. Ty znázorňují časový vývoj přikazované polohy v porovnání s reálnou pozicí. Skokový vývoj modré křivky zachycuje zpracování příkazů pro pohyb osy. Červená křivka znázorňuje konvergentní vývoj polohy naměřené enkodérem. Z grafu je patrný plynulý přechod do cílové polohy, který byl, vzhledem k vyšší hmotnosti polohovaných zařízení, upřednostněn, aby se zabránilo setrvačnému pohybu a tím pádem zvýšenému namáhání zařízení.

Graf na obrázku 6.1 zobrazuje polohování osy z výchozí pozice o $20\mu m$. V grafu na obrázku 6.3 je znázorněno zadání vstupních parametrů do grafického rozhraní LinuxCNC. Graf na obrázku 6.2 demonstruje polohování osy zpět do výchozí pozice. Z tohoto obrázku je patrná přikázaná a skutečná hodnota polohy, která se liší o cca $0.5\mu m$ Tato hodnota je dostačující pro provádění přesného tomografického měření.

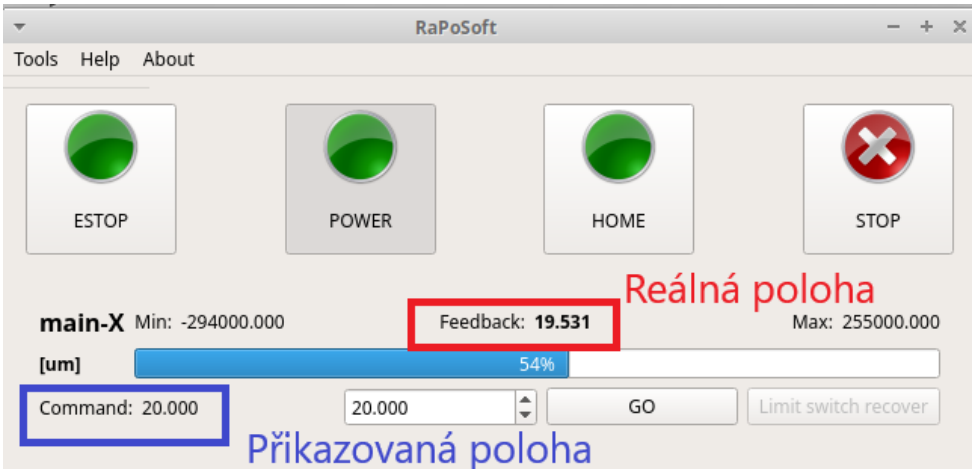


Obrázek 6.1: Výstup z Halscope znázorňující dopředný pohyb osy



Obrázek 6.2: Výstup z Halscope znázorňující zastavení pohybu

Požadavky na polohování jsou definovány v grafickém rozhraní vyvíjeném v rámci studentské práce na Fakultě dopravní. Na základě vstupů z grafického rozhraní jsou v LinuxCNC vytvářeny příkazy pro řídící kartu 5i23. Řídící karta zpracovává výstupy z LinuxCNC a na základě konfigurace generuje signály, které jsou do krokových motorů přenášeny prostřednictvím rozšiřujících karet. V případě CT bylo použito řízení se zpětnou vazbou, což umožňuje adaptování řídících parametrů (například rychlost a směr posunu) v závislosti na naměřené poloze z enkodéru. Náhled grafického rozhraní je k dispozici na obrázku 6.3.



Obrázek 6.3: Grafické rozhraní s přikazovanou (modrá) a reálnou (červená) polohou (v mikrometrech)

7. Závěr

V rámci praktické části této bakalářské práce došlo k sestavení, zapojení a uvedení do zkušebního provozu výpočtového tomografu. První krok spočíval v kompletním sestavení všech komponent. K řízení CT byl využit systém LinuxCNC v kombinaci s řídicími kartami společnosti Mesa Electronics. Karty poskytují potřebné periferie pro zapojení a řízení všech použitých komponent polohování jako jsou krokové motory, koncové spínače, enkodéry.

Jednou z hlavních výzev aplikace těchto řídících karet byla absence dokumentace pro vytvoření konfiguračního souboru, určeného pro unikátní kombinaci koncových zařízení. S využitím dostupných materiálů zabývající se problematikou a funkční analýzou existujících konfiguračních souborů byl vytvořen pilotní konfigurační soubor. Ke kompilaci byl využit program Xilinx ISE Design Suite. Pro test funkčnosti konfigurace byla koncová zařízení připojena do řídících karet pomocí kabelů, které bylo nezbytné vytvořit, neboť dostupná kabeláž nebyla postačující pro specifický způsob zapojení. Tento typ zapojení byl určený pouze pro testovací provoz.

Vzhledem k absenci obecné dokumentace byl v rámci teoretické části této práce vytvořen návod k vytváření dalších vlastních konfigurací pro budoucí využití. Tímto je umožněno lépe využít možností stávajícího hardwaru - pro specifická využití nyní již není nutno kupovat speciální předkonfigurované karty, tyto je možné realizovat na stávajícím hardwaru s využitím vlastní konfigurace. Dále bylo uskutečněno testování dalších komunikačních protokolů používaných pokročilými polohovacími zařízeními, jejichž aplikací se zvyšuje přesnost polohování.

Po dokončení byla v rámci testovacího provozu zkoumána dostupnost jednotlivých funkcionalit konfiguračních souborů. Na základě výsledků testování se vytvořila finální konfigurace, k níž byly zpracovány podklady pro vytvářenou řídící jednotku. Na této jednoce jsem se podílela návrhem plošných spojů v programu Autodesk EAGLE, jejich osazením a realizací finální kabeláže.

Ke dni odevzdání práce je zařízení sestavené, zapojené a uvedené do provozu včetně dedikované řídící jednotky. Plánované měření však nemohlo být z legislativních důvodů provedeno, nicméně funkcionalita klíčových komponent byla v práci vhodně demonstrována. Vývoj zařízení bude probíhat i nadále, výhledově se počítá s implementací rentgenové fluorescenční analýzy.

Seznam použité literatury

1. *Computed Tomography*. Dostupné také z: https://www.nde-ed.org/EducationResources/CommunityCollege/Radiography/AdvancedTechniques/computedtomography.htm.
2. DAVID DOWSETT Patrick A Kenny, R Eugene Johnston. *The Physics of Diagnostic Imaging*. Second. CRC Press, 2006.
3. BECKHOFF B., Kanngießer B. & Langhoff N. (Eds.) *Handbook of Practical X-Ray Fluorescence Analysis*. Springer, 2006.
4. AGARWAL, Tarun. *Basic FPGA Architecture and its Applications*. Dostupné také z: https://www.edgefx.in/fpga-architecture-applications/.
5. KOLOUCH, Jaromír. Programovatelné logické obvody a hradlová pole – moderní stavebníprvky číslicových systémů. *Automatizace ,* 2009.
6. CHU, Pong P. *FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version*. Wiley-Interscience, 2008. ISBN 9780470185315.
7. ŠŤASTNÝ, Jakub. *FPGA prakticky Realizace číslicových systémů pro programovatelná hradlová pole*. BEN - technická literatura, 2011. ISBN 9788073002619.
8. MAXFIELD, Clive. Xilinx redefines the non-volatile FPGA landscape. *EE Times*. 2007.
9. XILINX. *Spartan-3 FPGA FamilyData Sheet*. 2018. Dostupné také z: https://www.xilinx.com/support/documentation/data_sheets/ds312.pdf.
10. XILINX. *Using Block RAM in Spartan-3 Generation FPGAs*. 2005. Dostupné také z: https://www.xilinx.com/support/documentation/application_notes/xapp463.pdf.
11. XILINX. *Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs*. 2006. Dostupné také z: https://www.xilinx.com/support/documentation/application_notes/xapp462.pdf.
12. *Boolevova algebra*. Dostupné také z: http://et-pocitacovesystemy.wz.cz/cislicova_technika/logfce/bool_algebra/booleova_algebra.html.

13. PELLERIN, David. *An Introduction to VHDL*. Dostupné také z: http://www.pldworld.info/_hdl/1/VHDL_courses/www.acc-eda.com/h_intro.htm.

14. ŠUSTA, Richard. *Příkladný úvod do VHDL*. 2013. Dostupné také z: dcenet.felk.cvut.cz/edu/fpga/doc/PrikladnyUvodDoVHDL.pdf.

15. KOŘÍNEK, Milan. *Implementace Logického Analyzátoru do FPGA*. 2012. Vysoké učení technické v Brně.

16. LIBOSVÁR, Jan. *Hodiny s hlasovým výstupem v FPGA*. 2008. České vysoké učení technické v Praze.

17. KUBÍČEK, Michal. *Úvod do problematiky obvůdů FPGA pro integrovanou výuku VUT a VŠB-TUO*. 2014. Disertační práce. Vysoké učení technické v Brně.

18. *VHDL MINI-REFERENCE*. Dostupné také z: https://www.ics.uci.edu/~jmoorkan/vhdlref/vhdl.html.

19. RADA, Václav. *Modular Multi-process Control Software for Experimantal Devices*. 2019. České vysoké učení technické v Praze.

20. *Mesa HostMot2 Driver*. 2019. Dostupné také z: http://linuxcnc.org/docs/html/drivers/hostmot2.html.

21. XILINX. *ISE Design Suite*. Dostupné také z: https://www.xilinx.com/products/design-tools/ise-design-suite.html.

22. XILINX. *Xilinx ISE Overview*. 2008. Dostupné také z: https://www.xilinx.com/support/documentation/sw_manuals/xilinx10/isehelp/ise_c_overview.htm.

23. *Netlist*. Dostupné také z: http://www.vlsiip.com/asic_dictionary/N/netlist.html.

24. VLASTNOSTÍMATERIÁLŮ, Modulární jednotka řízení experimentálníchzařízení pro měření mechanických. *Jan Šleichrt*. 2016. České Vysoké Učení Technické v Praze.

25. *DM422C - digitální driver pro 2-fázové krokové motory, 36V, 2.2A*. Dostupné také z: http://www.cncshop.cz/dm422c-driver-pro-2-fazove-krokove-motory-40v-2-2a.

26. *LM13 linear incremental magnetic encoder system*. Dostupné také z: https://www.rls.si/en/lm13-linear-magnetic-encoder-system.

27. *Direct Driven Rotary Tables*. Dostupné také z: http://www.akribis-sys.com/pro_detail/productId=24.html.

28. *5I23 ANYTHING I/O MANUAL*. Dostupné také z: http://www.mesanet.com/pdf/parallel/5i23man.pdf.

29. *Mesa Anything I/O Daughter Cards*. Dostupné také z: `http://www.mesanet.com/`.

Seznam obrázků

2.1 Schéma laboratorního CT [1] 12

2.2 Schéma směrů pohybu třech komponent 13

2.3 Blokové schéma řízení 14

3.1 Schéma obecné struktury FPGA [5] 17

3.2 Schéma hradlového pole z rodiny Spartan 3 [9] 20

3.3 Schéma rozdělení slices v CLB [9] 21

3.4 Označení jednotlivých oken v programu ISE Design Suite určených pro práci a syntézu.
1 - okno se zdroji, 2 - okno s procesy, 3 - konzole, 4 - hlavní pracovní okno 27

4.1 Screenshot programu ISE Design suite 33

4.2 Pracovní okno zobrazující přehled projektu, který se zobrazí po otevření nového projektu včetně upozornění na 2 chyby 34

4.3 Původní chybové řádky 78 a 93 souboru TOP9054Hostmot2.vhdl 34

4.4 Řádky 78 a 93 souboru TOP9054Hostmot2.vhdl po volbě karty a konfiguračního souboru 35

4.5 Okno s procesy umožňující spuštění příkazu pro generování souboru ve formátu .bit 36

5.1 Driver DM422C[25] 38

5.2 Inkrementální lineární enkodér LM13 [26] 38

5.3 Rotační stolek Akribis ACD120 [27] 39

5.4 Řídicí karta 5i23 [29] 40

5.5 Rozšiřující karty - zleva 7i52, 7i47, 7i37TA [29] 40

5.6 První zapojení všech komponent napřímo do řídicích karet 42

5.7 Sestavené zařízení se zapojením komponent napřímo do řídicích karet 42

5.8 Návrh vytvořeného plošného spoje 43

5.9 Fotografie části řídicí jednotky 43

5.10 Schéma zapojení řídící jednotky 44

6.1 Výstup z Halscope znázorňující dopředný pohyb osy 46

6.2 Výstup z Halscope znázorňující zastavení pohybu 46

6.3 Grafické rozhraní s přikazovanou (modrá) a reálnou (červená) polohou (v mikrometrech) 47

A. Moduly a piny

Příloha obsahuje popis všech použitých a testovaných modulů, příklad jejich inicializace a informace o pinech, jež mohou být v rámci modulů použity.

WatchDogTag

Watch Dog je ochranný modul, sloužící ke spolehlivé detekci chyby a její následné eliminaci. V případě detekování chyby, odpojí modul všechny I/O piny z jejich modulových instancí, které tím přejdou do stavu s vysokou impedancí, která zabrání toku veškerých signálů. V tomto případě nedojde k narušení samotných HostMot2 modulů, instance tedy nepřestanou generovat výstupní signály, ty však už nejsou přenášeny na koncová zařízení.

- Tento modul by měl být součástí každé konfigurace
- Pro tento modul nejsou dedikovány žádné piny

Inicializace modulu

```
(WatchDogTag, x"00", ClockLowTag, x"01", WatchDogTimeAddr&PadT, WatchDogNumRegs,
xx"00", WatchDogMPBitMask),
```

HM2DPLL

DPLL je algoritmus navržený k řešení problematiky NP-úplných problémů. Na jeho principech je založený modul HM2DPLL, sloužící ke spouštění cyklů a čtení jejich výstupů. Modul je nezbytný pro správný chod modulů, které pro správný chod potřebují hodinové signály. Takovým modulem může být například SSSITag (popis tohoto modul je součástí přílohy) .

- Tento modul by měl být součástí každé konfigurace
- Pro tento modul nejsou dedikovány žádné piny

Inicializace modulu:

```
(HM2DPLLTag, x"00", ClockLowTag, x"01", HM2DPLLBaseRateAddr &PadT, HM2DPLLNumRegs,
x"00", HM2DPLLMPBitMask),
```

PWMTag

PWMTag je modul, který slouží k aplikaci pulzně šířkové modulace signálu.

Inicializace modulu:

```
(PWMTag, x"00", ClockHighTag, x"06", PWMValAddr&PadT, PWMNumRegs, x"00",
PWMMPBitMask),
```

Piny:

`PWMAOutPin` - TX

`PWMCEnaPin` - TX

`PWMBDirPin` - TX

Inicializace pinů:

```
IOPortTag & x"00"& PWMTag & PWMAOutPin,
IOPortTag & x"00"& PWMTag & PWMCEnaPin,
IOPortTag & x"00"& PWMTag & PWMBDirPin,
```

StepgenTag

Stepgen slouží ke generování signálů na ovládání krokových motorů. Krokové motory jsou řízeny pomocí signálů STEP a DIR, kdy STEP generuje kroky a DIR určuje směr posuntí.

Inicializace modulu:

```
(StepGenTag, x"02", ClockLowTag, x"09", StepGenRateAddr&PadT, StepGenNumRegs,
x"00", StepGenMPBitMask),
```

Piny:

`StepGenDirPin` - TX

`StepGenStepPin` - TX

Inicializace pinů:

```
IOPortTag & x"00"& StepGenTag & StepGenDirPin,
IOPortTag & x"00"& StepGenTag & StepGenStepPin,
```

SSerialTag

Smart serial je rozhraní umožňující připojit více zařízení z řady "smart-serial remotes"prostřednictvím jednoho hardwarového portu.

Inicializace modulu:

```
( SSerialTag, x"00", ClockLowTag, x"01", SSerialCommandAddr&PadT, SSerialNumRegs,
x"10", SSerialMPBitMask),
```

Piny:

`SSerialTX0Pin` - TX

`SSerialRX1Pin` - RX

Inicializace pinů:

```
IOPortTag & x"00"& SSerialTag & SSerialTX0Pin,
IOPortTag & x"00"& SSerialTag & SSerialRX0
```

QCount

QCount slouží k přenosu dat z inkrementálních enkodérů.

Inicializace modulu:

```
(IOPortTag, x"00", ClockLowTag, x"02", PortAddr&PadT,IOPortNumRegs, x"00",
IOPortMPBitMask),
```

Piny:

`QCountQAPin` - TX

`QCountQBPin` - TX

`QCountIDXPin` - TX

Inicializace pinů:

```
IOPortTag & x"00"& QCountTag & QCountQAPin,
IOPortTag & x"00"& QCountTag & QCountQBPin,
IOPortTag & x"00"& QCountTag & QCountIdxPin,
```

MuxedQCount

Tento modul je určený na komunikaci s inkrementálními enkodéry. Narozdíl od jiných modulů určených pro enkodéry, tento možňuje komunikovat s více enkodéry pomocí jedné sady I/O hardwarových portů mother board tím, že využívá technologii multiplexování. Multiplexovaný přenos dat je dostupny pouze na pinech daughter cards označených jako MENCA, MENCB, MIDX.

Inicializace modulu:

```
MuxedQcountTag, MQCRev, ClockLowTag, x"08",MuxedQcounterAddr&PadT
MuxedQCounterNumRegs,x"00",MuxedQCounterMPBitMask),
```

Piny:

`MuxedQCountQAPin` - TX

`MuxedQCountQBPin` - TX

`MuxedQCountIDXPin` - TX

Inicializace pinů:

```
IOPortTag & x"00"& MuxedQCountTag & MuxedQCountQAPin,
IOPortTag & x"00"& MuxedQCountTag & MuxedQCountQBPin,
IOPortTag & x"00"& MuxedQCountTag & MuxedQCountIdxPin,
```

SSSITag

Modul SSSITag podporující komunikační rozhraní SSI (Synchronous serial interface), určený pro absolutní enkodéry musí být pro správný chod inicializován společně s modulem HM2DPLL.

Inicializace modulu:

```
SSSITag, x"00", ClockLowTag, x"24", SSSIDataAddr&PadT, SSSINumRegs, x"00",
SSSIMPBitMask),
```

Piny:

`SSSIClkPin` - TX

`SSSIDataPin` - RX

Inicializace pinů:

```
IOPortTag & x"00"& SSSITag & SSSIClkPin,
IOPortTag & x"00"& SSSITag & SSSIDataPin,
```

BISSTag

Modul BISSTag podporující komunikační rozhraní BiSS-C, (bidirectional, serial, synchronous) určené pro absolutní enkodéry, není podporovaný aktuální verzí LinuxCNC.

Inicializace modulu:

`(BISSTag, x"00", ClockHighTag, x"04", BISSDataAddr&PadT, BISSNumRegs, x"00", BISSMPBitMask),`

Piny:

`BISSClkPin` - TX

`BISSDataPin` - RX

Inicializace pinů:

```
IOPortTag & x"00"& BISSTag & BISSDataPin,
IOPortTag & x"00"& BISSTag & BISSClkPin,
```

NullTag

NullTag je modul bez funkcí.

`(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),`

Inicializace pinů: `IOPortTag & x"00"& NullTag & x"00",`

B. Konfigurační soubor

```
library IEEE;
use IEEE.std_logic_1164.all;  -- defines std_logic types
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

use work.IDROMConst.all;

package 7i52_7i47_v2 is
constant ModuleID : ModuleIDType :=(
(WatchDogTag, x"00", ClockLowTag, x"01", WatchDogTimeAddr&PadT, WatchDogNumRegs,
x"00",WatchDogMPBitMask),
(IOPortTag,x"00", ClockLowTag,x"01",PortAddr&PadT, IOPortNumRegs,x"00",
IOPortMPBitMask),
(LEDTag, x"00",ClockLowTag, x"01",LEDAddr&PadT, LEDNumRegs, x"00",
LEDMPBitMask),
(StepGenTag, x"02",ClockLowTag, x"09",StepGenRateAddr&PadT, StepGenNumRegs,
"00",StepGenMPBitMask),
(MuxedQcountTag, MQCRev, ClockLowTag, x"03", MuxedQcounterAddr&PadT,
MuxedQCounterNumRegs,x"00",MuxedQCounterMPBitMask),
(QcountTag, x"02", ClockLowTag, x"00", QcounterAddr&PadT, QcounterNumRegs, x"00",
QCounterMPBitMask),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
```

```
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000"),
(NullTag, x"00", NullTag, x"00", NullAddr&PadT, x"00", x"00", x"00000000")
);


constant PinDesc : PinDescType :=(
--  Base func  sec unit sec func   sec pin
-- 7i52 card
IOPortTag & x"00" & NullTag & x"00",                        -- I/O 00 PIN1
IOPortTag & x"00" & MuxedQCountTag & MuxedQCountQAPin,      -- I/O 01 PIN3 MA0
IOPortTag & x"00" & MuxedQCountTag & MuxedQCountQBPin,      -- I/O 02 PIN5 MB0
IOPortTag & x"00" & MuxedQCountTag & MuxedQCountIDXPin,     -- I/O 03 PIN7 MX0
IOPortTag & x"01" & MuxedQCountTag & MuxedQCountQAPin,      -- I/O 04 PIN9 MA1
IOPortTag & x"01" & MuxedQCountTag & MuxedQCountQBPin,      -- I/O 05 PIN11 MB1
IOPortTag & x"01" & MuxedQCountTag & MuxedQCountIDXPin,     -- I/O 06 PIN13 MX1
```

```
IOPortTag & x"02" & MuxedQCountTag & MuxedQCountQAPin,     -- I/O 07 PIN15 MA2
IOPortTag & x"02" & MuxedQCountTag & MuxedQCountQBPin,     -- I/O 08 PIN17 MB2
IOPortTag & x"02" & MuxedQCountTag & MuxedQCountIDXPin,    -- I/O 09 PIN19 MX2
IOPortTag & x"00" & NullTag & x"00",                       -- I/O 10 PIN21
IOPortTag & x"06" & StepGenTag & StepGenDirPin,            -- I/O 11 PIN23 Dir6
IOPortTag & x"00" & NullTag & x"00",                       -- I/O 12 PIN25
IOPortTag & x"06" & StepGenTag & StepGenStepPin,           -- I/O 13 PIN27 Step6
IOPortTag & x"00" & NullTag & x"00",                       -- I/O 14 PIN29
IOPortTag & x"07" & StepGenTag & StepGenDirPin,            -- I/O 15 PIN31 Dir7
IOPortTag & x"00" & NullTag & x"00",                       -- I/O 16 PIN33
IOPortTag & x"07" & StepGenTag & StepGenStepPin,           -- I/O 17 PIN35 Step7
IOPortTag & x"00" & NullTag & x"00",                       -- I/O 18 PIN37
IOPortTag & x"08" & StepGenTag & StepGenDirPin,            -- I/O 19 PIN39 Dir8
IOPortTag & x"00" & NullTag & x"00",                       -- I/O 20 PIN41
IOPortTag & x"08" & StepGenTag & StepGenStepPin,           -- I/O 21 PIN43 Step8
IOPortTag & x"00" & NullTag & x"00",                       -- I/O 22 PIN45
IOPortTag & x"00" & NullTag & x"00",                       -- I/O 23 PIN47

--7i47
IOPortTag & x"00" & StepGenTag & StepGenDirPin,      -- I/O 24 PIN1 Dir0
IOPortTag & x"00" & StepGenTag & StepGenStepPin,     -- I/O 25 PIN3 Step0
IOPortTag & x"01" & StepGenTag & StepGenDirPin,      -- I/O 26 PIN5 Dir1
IOPortTag & x"01" & StepGenTag & StepGenStepPin,     -- I/O 27 PIN7 Step1
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 28 PIN9
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 29 PIN11
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 30 PIN13
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 31 PIN15
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 32 PIN17
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 33 PIN19
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 34 PIN21
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 35 PIN23
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 36 PIN25
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 37 PIN27
IOPortTag & x"00" & NullTag & x"00",                 -- I/O 38 PIN29
```

```
IOPortTag & x"00" & NullTag & x"00",                -- I/O 39 PIN31
IOPortTag & x"02" & StepGenTag & StepGenDirPin,     -- I/O 40 PIN33 Dir2
IOPortTag & x"02" & StepGenTag & StepGenStepPin,    -- I/O 41 PIN35 Step2
IOPortTag & x"03" & StepGenTag & StepGenDirPin,     -- I/O 42 PIN37 Dir3
IOPortTag & x"03" & StepGenTag & StepGenStepPin,    -- I/O 43 PIN39 Step3
IOPortTag & x"04" & StepGenTag & StepGenDirPin,     -- I/O 44 PIN41 Dir4
IOPortTag & x"04" & StepGenTag & StepGenStepPin,    -- I/O 45 PIN43 Step4
IOPortTag & x"05" & StepGenTag & StepGenDirPin,     -- I/O 46 PIN46 Dir5
IOPortTag & x"05" & StepGenTag & StepGenStepPin,    -- I/O 47 PIN47 Step5

--7i37TA
IOPortTag & x"00" & NullTag & x"00",    -- I/O 48 PIN1
IOPortTag & x"00" & NullTag & x"00",    -- I/O 49 PIN3
IOPortTag & x"00" & NullTag & x"00",    -- I/O 50 PIN5
IOPortTag & x"00" & NullTag & x"00",    -- I/O 51 PIN7
IOPortTag & x"00" & NullTag & x"00",    -- I/O 52 PIN9
IOPortTag & x"00" & NullTag & x"00",    -- I/O 53 PIN11
IOPortTag & x"00" & NullTag & x"00",    -- I/O 54 PIN13
IOPortTag & x"00" & NullTag & x"00",    -- I/O 55 PIN15
IOPortTag & x"00" & NullTag & x"00",    -- I/O 56 PIN17
IOPortTag & x"00" & NullTag & x"00",    -- I/O 57 PIN19
IOPortTag & x"00" & NullTag & x"00",    -- I/O 58 PIN21
IOPortTag & x"00" & NullTag & x"00",    -- I/O 59 PIN23
IOPortTag & x"00" & NullTag & x"00",    -- I/O 60 PIN25
IOPortTag & x"00" & NullTag & x"00",    -- I/O 61 PIN27
IOPortTag & x"00" & NullTag & x"00",    -- I/O 62 PIN29
IOPortTag & x"00" & NullTag & x"00",    -- I/O 63 PIN31
IOPortTag & x"00" & NullTag & x"00",    -- I/O 64 PIN33
IOPortTag & x"00" & NullTag & x"00",    -- I/O 65 PIN35
IOPortTag & x"00" & NullTag & x"00",    -- I/O 66 PIN37
IOPortTag & x"00" & NullTag & x"00",    -- I/O 67 PIN39
IOPortTag & x"00" & NullTag & x"00",    -- I/O 68 PIN41
IOPortTag & x"00" & NullTag & x"00",    -- I/O 70 PIN43
IOPortTag & x"00" & NullTag & x"00",    -- I/O 71 PIN45
```

```
IOPortTag & x"00" & NullTag & x"00",    -- I/O 72 PIN47

emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin);

end package 7i52_7i47_v2;
```

C. Šablona pro vytvoření uživatelského konfiguračního souboru

```
library IEEE;
use IEEE.std_logic_1164.all;  -- defines std_logic types
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.IDROMConst.all;

package FILENAME is
constant ModuleID : ModuleIDType :=(
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
```

```
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000"),
(NullTag,x"00",NullTag,x"00",NullAddr&PadT,x"00",x"00",x"00000000")
);


constant PinDesc : PinDescType :=(
IOPortTag & x"00" & NullTag & x"00", -- I/O 00 PIN1
IOPortTag & x"00" & NullTag & x"00", -- I/O 01 PIN3
IOPortTag & x"00" & NullTag & x"00", -- I/O 02 PIN5
IOPortTag & x"00" & NullTag & x"00", -- I/O 03 PIN7
IOPortTag & x"00" & NullTag & x"00", -- I/O 04 PIN9
IOPortTag & x"00" & NullTag & x"00", -- I/O 05 PIN11
IOPortTag & x"00" & NullTag & x"00", -- I/O 06 PIN13
IOPortTag & x"00" & NullTag & x"00", -- I/O 07 PIN15
IOPortTag & x"00" & NullTag & x"00", -- I/O 08 PIN17
IOPortTag & x"00" & NullTag & x"00", -- I/O 09 PIN19
IOPortTag & x"00" & NullTag & x"00", -- I/O 10 PIN21
IOPortTag & x"00" & NullTag & x"00", -- I/O 11 PIN23
IOPortTag & x"00" & NullTag & x"00", -- I/O 12 PIN25
IOPortTag & x"00" & NullTag & x"00", -- I/O 13 PIN27
```

```
IOPortTag & x"00" & NullTag & x"00", -- I/O 14 PIN29
IOPortTag & x"00" & NullTag & x"00", -- I/O 15 PIN31
IOPortTag & x"00" & NullTag & x"00", -- I/O 16 PIN33
IOPortTag & x"00" & NullTag & x"00", -- I/O 17 PIN35
IOPortTag & x"00" & NullTag & x"00", -- I/O 18 PIN37
IOPortTag & x"00" & NullTag & x"00", -- I/O 19 PIN39
IOPortTag & x"00" & NullTag & x"00", -- I/O 20 PIN41
IOPortTag & x"00" & NullTag & x"00", -- I/O 21 PIN43
IOPortTag & x"00" & NullTag & x"00", -- I/O 22 PIN45
IOPortTag & x"00" & NullTag & x"00", -- I/O 23 PIN47


IOPortTag & x"00" & NullTag & x"00", -- I/O 24 PIN1
IOPortTag & x"00" & NullTag & x"00", -- I/O 25 PIN3
IOPortTag & x"00" & NullTag & x"00", -- I/O 26 PIN5
IOPortTag & x"00" & NullTag & x"00", -- I/O 27 PIN7
IOPortTag & x"00" & NullTag & x"00", -- I/O 28 PIN9
IOPortTag & x"00" & NullTag & x"00", -- I/O 29 PIN11
IOPortTag & x"00" & NullTag & x"00", -- I/O 30 PIN13
IOPortTag & x"00" & NullTag & x"00", -- I/O 31 PIN15
IOPortTag & x"00" & NullTag & x"00", -- I/O 32 PIN17
IOPortTag & x"00" & NullTag & x"00", -- I/O 33 PIN19
IOPortTag & x"00" & NullTag & x"00", -- I/O 34 PIN21
IOPortTag & x"00" & NullTag & x"00", -- I/O 35 PIN23
IOPortTag & x"00" & NullTag & x"00", -- I/O 36 PIN25
IOPortTag & x"00" & NullTag & x"00", -- I/O 37 PIN27
IOPortTag & x"00" & NullTag & x"00", -- I/O 38 PIN29
IOPortTag & x"00" & NullTag & x"00", -- I/O 39 PIN31
IOPortTag & x"00" & NullTag & x"00", -- I/O 40 PIN33
IOPortTag & x"00" & NullTag & x"00", -- I/O 41 PIN35
IOPortTag & x"00" & NullTag & x"00", -- I/O 42 PIN37
IOPortTag & x"00" & NullTag & x"00", -- I/O 43 PIN39
IOPortTag & x"00" & NullTag & x"00", -- I/O 44 PIN41
IOPortTag & x"00" & NullTag & x"00", -- I/O 45 PIN43
```

```
IOPortTag & x"00" & NullTag & x"00", -- I/O 46 PIN46
IOPortTag & x"00" & NullTag & x"00", -- I/O 47 PIN47



IOPortTag & x"00" & NullTag & x"00", - I/O 48 PIN1
IOPortTag & x"00" & NullTag & x"00", -- I/O 49 PIN3
IOPortTag & x"00" & NullTag & x"00", -- I/O 50 PIN5
IOPortTag & x"00" & NullTag & x"00", -- I/O 51 PIN7
IOPortTag & x"00" & NullTag & x"00", -- I/O 52 PIN9
IOPortTag & x"00" & NullTag & x"00", -- I/O 53 PIN11
IOPortTag & x"00" & NullTag & x"00", -- I/O 54 PIN13
IOPortTag & x"00" & NullTag & x"00", -- I/O 55 PIN15
IOPortTag & x"00" & NullTag & x"00", -- I/O 56 PIN17
IOPortTag & x"00" & NullTag & x"00", -- I/O 57 PIN19
IOPortTag & x"00" & NullTag & x"00", -- I/O 58 PIN21
IOPortTag & x"00" & NullTag & x"00", -- I/O 59 PIN23
IOPortTag & x"00" & NullTag & x"00", -- I/O 60 PIN25
IOPortTag & x"00" & NullTag & x"00", -- I/O 61 PIN27
IOPortTag & x"00" & NullTag & x"00", -- I/O 62 PIN29
IOPortTag & x"00" & NullTag & x"00", -- I/O 63 PIN31
IOPortTag & x"00" & NullTag & x"00", -- I/O 64 PIN33
IOPortTag & x"00" & NullTag & x"00", -- I/O 65 PIN35
IOPortTag & x"00" & NullTag & x"00", -- I/O 66 PIN37
IOPortTag & x"00" & NullTag & x"00", -- I/O 67 PIN39
IOPortTag & x"00" & NullTag & x"00", -- I/O 68 PIN41
IOPortTag & x"00" & NullTag & x"00", -- I/O 70 PIN43
IOPortTag & x"00" & NullTag & x"00", -- I/O 71 PIN45
IOPortTag & x"00" & NullTag & x"00", -- I/O 72 PIN47

emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
```

```
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,
emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin,emptypin);

end package FILENAME;
```

D. Schéma zapojení řídících karet

Schéma zapojení řídících karet