ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra elektrických pohonů a trakce

Možnosti využití FPGA pro řízení pohonů

Usage of FPGA for controlling electric drives

Diplomová práce

Studijní program: Elektrotechnika, Energetika a Management
Studijní obor: Elektrické pohony

Vedoucí práce: Ing. Jan Bauer, Ph.D.

Petr Zakopal

Praha 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zakopal** Jméno: **Petr** Osobní číslo: **483802**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektrických pohonů a trakce**

Studijní program: **Elektrotechnika, energetika a management**

Specializace: **Aplikovaná elektrotechnika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Oživení pracoviště s měničem DCM a PLC SIMATIC

Název bakalářské práce anglicky:

Workpalce with Rectifier DCM and PLC SIMATIC

Pokyny pro vypracování:

1) Seznamte se s měničem řady DCM firmy SIEMENS
2) Oživte základní regulační smyčky měniče (otáčkovou, proudovou)
3) Prostudujte možnosti záznamu průběhů z měniče pomocí PLC nebo dotykového panelu
4) Pomocí PLC SIMATIC S1200 a dotykového panelu realizujte vzdálené ovládání a monitoring měniče
5) Na dotykovém panelu vytvořte obrazovku pro nastavování otáček nebo momentu motoru napájeného měničem

Seznam doporučené literatury:

[1] Weidauer J., Messer R. Electrical Drives, Publics Erlangen, 2014
[2] SCE Training Curriculum. Siemens AG, 2016
[3] Durry B. The Control Techniques Drives and Controls Handbook 2nd ed., IeT, 2009
[4] Pavelka J., Kobrle P. Elektrické pohony a jejich řízení. 3. přepracované vydání. Praha: České vysoké učení technické v Praze, 2016. ISBN 978-80-01-06007-0.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Bauer, Ph.D., katedra elektrických pohonů a trakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **24.01.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Jan Bauer, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

PROHLÁŠENÍ

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne _____ _____

Petr Zakopal

PODĚKOVÁNÍ

Nam quis commodo justo. Mauris diam metus, mattis sed rutrum in, volutpat sit amet sem. Nam bibendum commodo porttitor. Quisque eget lectus rutrum, molestie tortor id, iaculis nunc. Sed et maximus ipsum. Vivamus vel facilisis nisl. Curabitur eu nibh nec erat mollis finibus at in sapien. Mauris viverra sapien neque, nec lacinia odio laoreet eu. Quisque consectetur eros ac orci interdum scelerisque.

ABSTRAKT

Nam quis commodo justo. Mauris diam metus, mattis sed rutrum in, volutpat sit amet sem. Nam bibendum commodo porttitor. Quisque eget lectus rutrum, molestie tortor id, iaculis nunc. Sed et maximus ipsum. Vivamus vel facilisis nisl. Curabitur eu nibh nec erat mollis finibus at in sapien. Mauris viverra sapien neque, nec lacinia odio laoreet eu. Quisque consectetur eros ac orci interdum scelerisque.

Klíčová slova: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis aliquam finibus sagittis. Nunc venenatis, augue quis luctus dictum, elit justo pharetra leo, nec viverra purus dui at quam.

ABSTRACT

Nam quis commodo justo. Mauris diam metus, mattis sed rutrum in, volutpat sit amet sem. Nam bibendum commodo porttitor. Quisque eget lectus rutrum, molestie tortor id, iaculis nunc. Sed et maximus ipsum. Vivamus vel facilisis nisl. Curabitur eu nibh nec erat mollis finibus at in sapien. Mauris viverra sapien neque, nec lacinia odio laoreet eu. Quisque consectetur eros ac orci interdum scelerisque.

Keywords: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis aliquam finibus sagittis. Nunc venenatis, augue quis luctus dictum, elit justo pharetra leo, nec viverra purus dui at quam.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 Úvod

V době, kdy byla od elektrických pohonů požadována spolehlivost, vysoká účinnost a nenáročné ovšem kvalitní řízení, byly k řízení využívány samotné digitální signálové procesory. Postupem času dochází ke zjištění, že výkon DSP není dostatečný a na některé aplikace, kde je vyžadováno provedení značné množství náročných výpočtů za co nejkratší čas, nejsou vhodné. Proto nastupuje éra logických programovatelných polí (FPGA), které jsou schopny tyto výpočty provést s velmi nízkými nároky na energii a za velmi krátký čas.

V mnoha odvětvích se již začíná využívat embedded systém s Application Spefified Hardware, který je určen pouze na využití v předem dané aplikaci. Tento hardware slouží v dané aplikaci k jedinému účelu, který vykonává a na který je optimalizován. Tím se liší od procesoru, který vykonává mnoho instrukcí a využít ho pouze jako samostatnou výpočetní jednotku je z hlediska energetické i finanční náročnosti nevýhodné. Implementace hradlových polí přináší nejen v řízení elektrických pohonů zvýšení výpočetního výkonu, ale také snižování energetické náročnosti řízení.

Perspektiva logických programovatelných polí a hardwaerově urychlovaných aplikací je podpořena jejich využíváním i mimo obor elektrických pohonů a trakce. Z důvodu jejich veliké propustnosti, vysokých výpočetních výkonů a nízké energetické náročnosti jsou využívány v AI, machine learningu, zpracování obrazu, těžení kryptoměn a jiných nepohonářských aplikacích.

Nevýhodou problematiky FPGA je jejich složitější programovatelnost z hlediska tvoření aplikace. Aplikace je tvořena určitým postupem (workflow), který kladne vysoké nároky na vzdělání a zkušenosti vývojářů. Většina FPGA je programována pomocí jazyků Verilog či VHDL, které mohou pro softwarově orientované programátory představovat značnou překážku. Proto bylo vyvinuto tvoření aplikací pomocí vyšší úrovně syntézy (HLS), kdy je možné tvořit programy ve vyšších programovacích jazycích jako je například C, C++ či Python. HLS umožnilo rapidní rozšíření a využití Embedded FPGA Accelerated Applications v mnoha aplikacích a značně vylepšilo vývojářský požitek (developer experience, DX) při tvorbě aplikací.

Protože může být náročné vytvořit vlastní architekturu, složenou z CPU a spolupracujícího FPGA, je vhodné při prvotním vývoji aplikace využít dostupné vývojové desky obsahující již předpřipravené propojení jednotlivých komponent. Součástí těchto vývojových desek bývá také mnoho vstupů a výstupů (I/O) pro snadnější využití při lazení a tvoření aplikace. V této práci je využívána vývojová deska Zybo od firmy Digilent. Ovšem autor v textu představuje další možnosti, které mohou být pro konkrétní aplikace a využití vhodnější.

Tato práce se zajímá o aplikace a možné využití FPGA při řízení elektrických pohonů. Autor v ní představuje základní principy Hardware Accelerated Applications, z jakého důvodu je tento přístup perspektivní a proč je vhodné se orientovat tímto směrem.

2 Embedded Systems

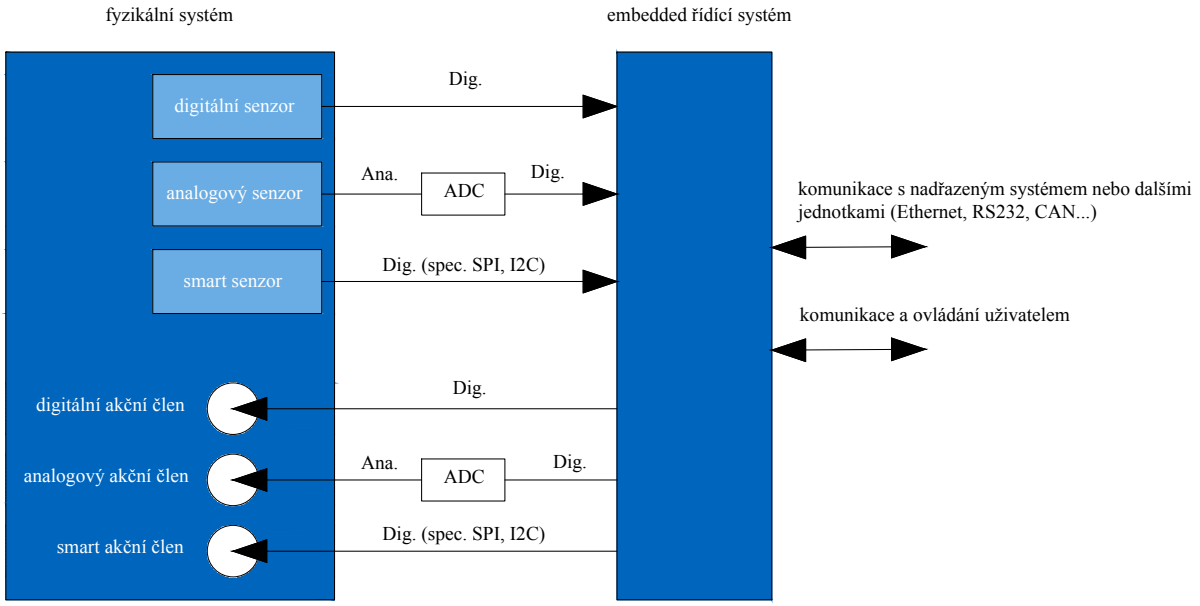
Embedded systémy je název pro skupinu zařízení, obecně systémů, které je možné charakterizovat jako specifické výpočetní zařízení, resp. počítače, které jsou určeny pro podporu funkce nebo řízení nějakého většího celku, produktu nebo fyzikálního systému. Oproti tomu osobní počítač je sice výpočetní zařízení, ale nelze mluvit o embedded systému, protože je určen pro mnoho univerzálních aplikací. [1]

Dalším důležitým rozdílem mezi *Embedded System* a obecným výpočetním zařízením je ten, že v případě embedded systému je interakce mezi systémem a uživatelem uměle omezena na základní ovládání či kontrolu funkce. Není předpokládáno, že by uživatel, jež aplikaci embedded systému využívá, výrazným způsobem zasahoval do jeho funkce. Naopak obecný výpočetní systém je uzpůsoben na podstatné zásahy uživatele. [1] [2]

Do embedded systému obecně vstupují vstupní signály, které jsou následně zpracovány a poté vybrané výsledky výpočtů jsou v podobě výstupní signálů výstupním produktem systému. Tyto produkty mohou pomocí akčních členů zasahovat do řízeného systému či aplikace. Vstupní signály většinou přicházejí ze speciálních snímačů, kompatibilních s embedded systémem (senzor teploty, senzor tlaku, senzor zrychlení, gyroskop apod.). Naopak jeho výstupem signály, vedoucí na konektory, na kterých se dle požadavků objevuje např. specifická hodnota napětí. Nebo mohou na výstupních pinech být připojené LED signalizace, komunikační sběrnice některých komunikačních systémů nebo výstupní LDC displaye. Způsob, kterým jsou kódovány vstupní a výstupní signály je většinou specificky určený aplikaci daného systému. [1]

K obecnému výpočetnímu systému je možné připojit vstupní periferie klasických osbních počítačů – myš, klávesnice, mikrofon. Jako hlavní výstupní periferie obecného systému je monitor. Jeho komunikace s periferiemi je většinou standardizována tak, aby bylo možné periferie libovolně zaměňovat bez změny funkčnosti. [1].

Na obrázku 2 - 1 je zobrazeno názorné blokové schéma řízení fyzikálního systému pomocí embedded systému. Tyto bloky mezi sebou komunikují pomocí digitálních signálů. Pokud tyto signály nejsou digitální, musí se před zpracováním v embedded systému zdiskretizovat.



Obr. 2 - 1 Blokové schéma Embedded systému a řízeného fyzikálního systému. (převzato a upraveno z [2])

2.1 Application Specific Integrated Circuit

S tématem embedded systémů se pojí pojem hardware, který je určen pro jedinou aplikaci. Tato skupina zařízení se nazývá *Application Specific Integrated Circuits, popř. Hardware* (ASICs, ASHW). V této oblasti je opět využíváno přesvědčení, že pokud je architektura HW přímo specializovaná na jednu aplikaci, je vysoká pravděpodobnost, že ji bude vykonávat bezchybně, kvalitně a rychle.

Tyto aplikace jsou využívaný v širokém spektru oborů jako je např. zpracování zvuku, videa, výpočtů apod. Tyto ASIC mohou také vykonávat potřebné rychlé výpočty pro matematické modely elektrických strojů, které jsou využívány např. pro HIL.

Než je tento specifický obvod vytvořen, je nutné jej navrhnout, vyzkoušet a odladit. K tomu slouží logická programovatelná pole, ve kterých je možné požadovaný HW navrhnut a odladit před velko produkcí ASIC. Pokud velko produkce není z ekonomických důvodů možná, jsou FPGA využívány i v produkční oblasti, kde je HW struktura, která by byla přítomna na ASIC, vytvořena přímo na FPGA.

2.2 Hardware Accelerated Applications

V mnoha aplikacích, nejen při řízení elektrických pohonů, je vyžadováno, aby výpočty nebo zpracování dat probíhalo vysokou rychlostí. Tento problém nemůže být většinou vyřešen použitím běžného procesoru (CPU), který je optimalizován na provádění obecných komplexních funkcí řízení běhu programu, komunikace či přesunu dat. V moderním světě dochází k exponenciálnímu nárůstu množství dat, které je potřeba zpracovat. Aby tyto data bylo možné v požadovaném čase zpracovat, je třeba využít specifický HW, který bude schopen požadavky rychlosti a výkonu uspokojit. Tento proces se nazývá *Hardware Acceleration*. [3]

Princip hardwaerové akcelerace spočívá v přesunu výpočetně náročných aktivit na zvláštní oddělený hardware. Celkové řízení běhu aplikace a komunikace je ovšem stále přítomno na řídícím CPU. Oddělený hardware, na kterém dochází k akceleraci výpočtů je optimalizován na vykonávanou úlohu a jeho využití přináší zefektivnění běhu celkové aplikace. [3]

Struktura, ve které je využíváno více oddělených hardwarových procesorových a akceleračních jednotek, se často nazývá heterogenní. [3]

Hardwaerová akcelerace poskytuje rychlejší výpočty než CPU, protože využívá značné úrovně paralelismu výpočtů. Oproti tomu klasické CPU vykonává jednotlivé instrukce sériově. I v případě, že CPU má více jader a využívá více vláken, nemůže se úrovni paralelismu při dané energetické náročnosti HW vyrovnat.

Pro HW akceleraci je v mnoha oblastech využíváno několik druhů jednotek, které jsou optimální pro dané aplikace.

Graphics Processing Units (GPUs) jsou jednotky, které převážně slouží k akceleraci zpracování a renderování grafických úloh. V době rapidního rozvoje elektroniky a SW je možné využití GPUs v mnoha odvětví umělé inteligence (AI) či kreativních odvětích. GPUs jsou využívány v aplikacích, kde není kladen veliký důraz na nízkou odezvu (latenci). [3]

Tensor Processing Units (TPUs) jsou jednotky, které slouží k provádění algoritmů strojového učení (machine-learning, ML). Jejich přímé datové propojení umožňuje velmi rychlý a přímý přenost dat. Díky přímému připojení nevyžadují využití pamětí, kter= by přenos dat zpomalovali. [3]

Field Programmable Gate Arrays (FPGAs) jsou jednotky, ve kterých není při výrobě pevně daná HW struktura. To umožňuje vytvoření, resp. naprogramování HW dle požadavků akcelerované aplikace. FPGAs jsou využívány i v on-line výpočtech matematických modelů elektrických strojů. Při realizaci této práce je pro akceleraci využíváno právě těchto programovatelných polí.

3 Programovatelné hradlové pole – FPGA

3.1 Vývoj FPGA z PLD

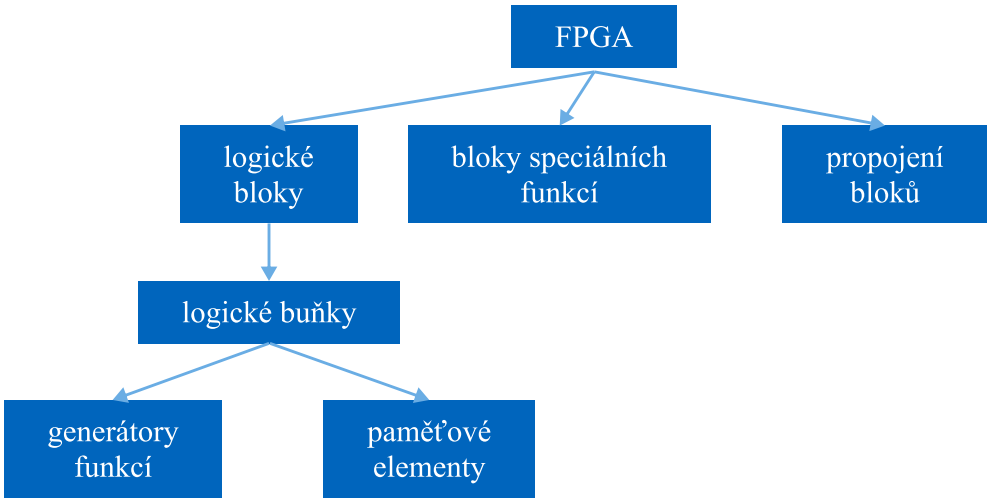
Programovatelné hradlové pole jsou zařízení, jejichž historický vývoj stojí na programovatelných logických zařízení (programmable logic devices, PLD). První PLD fungovala na principu Booleových funkcí součtu násobení (sum of products). Tyto zařízení obsahovala matici více vstupových bloků AND a OR. Programování požadované probíhalo pomocí přerušování vstupů do těchto logických bloků. Později byly do PLA přidány D klopné obvody s multiplexory. Díky těmto součástím bylo možné vytvářet logické kombinační a sekvenční obvody, resp. automaty. Posledním vylepšením PLA, které stálo před zrodem FPGA, spočívalo v umístění více PLA bloků (skládajících se z AND, OR, multiplexeru a D klopného obvodu) na jeden integrovaný čip. Programovatelné spojení různých PLA bloků a výstupů umožnilo vytvořit požadovanou funkci. [1]

3.2 Aktuální složení FPGA

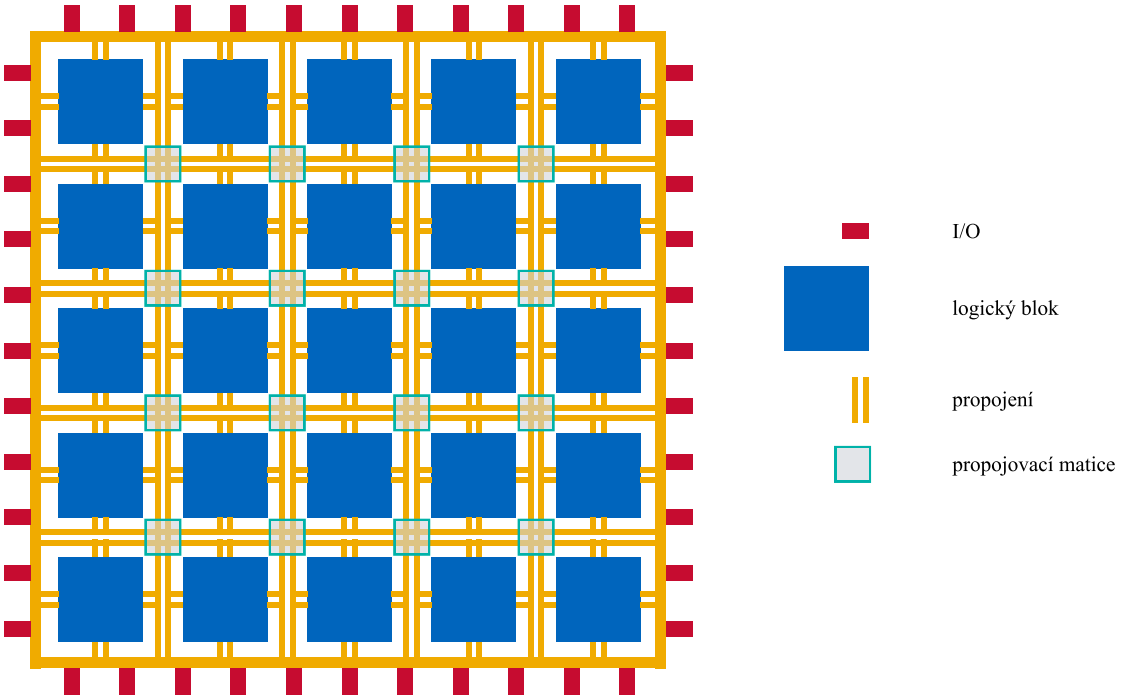
Moderní FPGA se skládají z 2D matice propojených programovatelných logických bloků, bloků speciálních funkcí a propojů vytvořených pomocí CMOS technologie. Po obvodě FPGA jsou rozmístěny vstupní a výstupní piny (I/O), připojené na zvláštní logické bloky. Použité logické bloky se skládají z mnoha logických buněk, které se skládají z generátorů funkcí a paměťových elementů. [1]

Na obr. 3 - 2 je možné pozorovat názorné schéma základního konceptu uspořádání FPGA. Na schématu jsou vyznačeny logické bloky, jejich propojení, propojovací matice pro aktivování jednotlivých propojů a vstupů a výstupů (I/O) FPGA.

I přesto, že se tato práce věnuje využití FPGA pro řízení elektrických pohonů je vhodné představit základní části FPGA a nastínit jejich funkci.



Obr. 3 - 1 Blokové schéma složení moderních FPGA.



Obr. 3 - 2 Základní koncept uspořádání FPGA.

3.2.1 Generátory funkcí

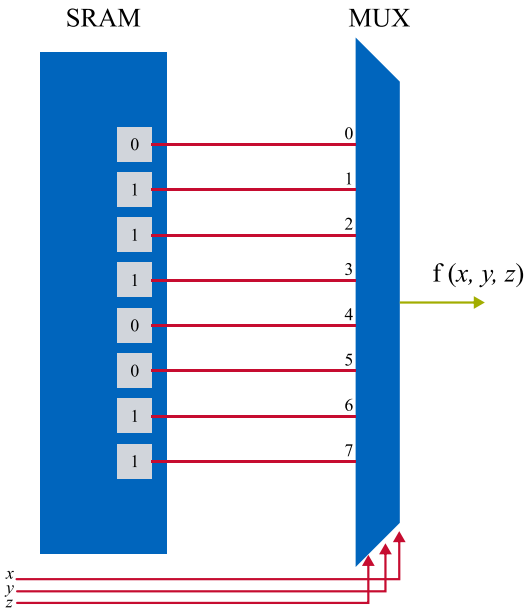
Oproti předchůdcům PLD, které pro generování funkcí používaly logická hradla tvořená CMOS tranzistory, využívají FPGA generátory funkcí.

Logickou funkci je možné popsat pravdivostní tabulkou, která má určitý počet vstupů a odpovídající počet výstupů. Dle [1] je možné si představit, že se generátor dané funkce skládá ze samostatné statické paměti (SRAM), jejíž výstupy jsou přímo přivedeny na vstup multiplexeru (MUX). Signály výběru výstupů by odpovídaly vstupním proměnným a jednotlivé vstupy do MUX výstupům funkce.

Pro bližší pochopení funkce generátoru funkcí z předchozího odstavce je možné představit realizaci smyšlené logické funkce $\mathrm{f}(x, y, z) = \bar{x}z + y$. Pravdivostní tabulka této smyšlené logické funkce je zobrazena v tab. 3 - 1. Odpovídající realizace pomocí MUX a SRAM je zobrazena na obr. 3 - 3. Tato reprezentace se nazývá look-up table (LUT). Grafické znázornění inspirováno [1].

Tab. 3 - 1 Pravdivostní tabulka ukázkové funkce, realizované v generátoru funkcí, umístěném v logickém bloku FPGA.

i	x	y	z	f(x, y, z)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1



Obr. 3 - 3 Ukázka, jakým způsobem realizuje funkční generátor požadovanou funkci pomocí SRAM a MUX.

Výhoda této reprezentace funkcí oproti logickým hradlům je, že doba zpoždění signálu (propagation delay) pro funkci je konstantní. Respektivě je konstantní, pokud funkci je možné realizovat jednou LUT. Pro relizace obecné funkce je zapotřebí multiplexer 2^n -> 1 a SRAM s počtem buněk 2^n, kde n je počet vstupních proměnných dané funkce. [1]

3.2.2 Paměťové elementy

Paměťové elementy jsou v FPGA realizovány pomocí D-klopných obvodů. Tyto obvody mohou při konfiguraci FPGA být nastaveny, že budou reagovat na nástupnou nebo sestupnou hranu časovacího signálu (clock, CLK) řídícího procesoru nebo na úroveň řídícího signálu (latch).[1]

Protože typ latch je citlivý na úroveň signálu, může být problematické dovést požadovaný signál na vstup klopného obvodu v požadovaném čase. Velmi často jsou proto paměťové členy konfigurovány jako D-klopné obvody reagující na hranu. Pokud je používán signál CLK vyšších frekvencí, je D-klopný obvod reagující na hranu snadněji schopný reagovat v požadovaném čase. [1]

Často jsou na vstup paměťových elementů připojeny výstupy multiplexerů *generátorů funkcí*. [1]

3.2.3 Logické buňky

Logické buňky jsou elementy, skládající se z *generátorů funkcí* a *generátorů funkcí*. Velmi často se počet logických buněk údává jako jeden ze základních parametrů FPGA, podle kterého je uživatel možný rozhodnout, zda je vhodný pro jeho aplikaci. Pomocí logické buňky nebo skupiny logických buněk je již možné vytvářet plnohodnotnou kombinační a sekvenčí logiku.[1]

3.2.4 Logické bloky

Logické bloky se skládají ze spojení několika logických buněk do jedné skupiny. Tato skupina buněk je obvykle na čipu geograficky blízko a proto dochází k minimalizaci zpoždění signálu. Velmi časté

je, že jednotlivé bloky mohou mít již předkonfigurovanou funkci, jako je např. sčítačka, dělička nebo násobička. [1]

3.2.5 Propojení bloků

Propojení bloků slouží ke spojení jednotlivých logických bloků a I/O. Pro spínání určených propojů jsou na čipu mezi jednotlivými propoji umístěny propojovací matice, resp. „přepínače“. Ty slouží ke spojení jinak oddělených propojů, logicých bloků a I/O. [1]

3.2.6 I/O bloky

I/O bloky jsou obvykle umístěny na okraji designu FPGA. Slouží k přivedení resp. vyvedení signálů FPGA. [1]

3.2.7 Bloky speciálních funkcí

Aby došlo např. ke snížení zpoždění signálu, který by bylo nutné vyvést z FPGA do externího CPU a naopak, jsou některé speciální funkce implementovány jako funkční bloky přímo do struktury FPGA. To umožňuje efektivní využití FPGA pro různorodé aplikace. [1]

Block RAM (BRAM) je blok, který slouží k uchování dat. Sice by bylo možné vytvořit paměťový blok z *Logických bloků*, ale docházelo by k omezení využití FPGA pro jeho původní aplikace a bylo by potřeba využít mnoho bloků. BRAM mají oddělený vstup a výstup, současně s odděleným CLK. Proto je možné do BRAM zároveň data zapisovat a zároveň z něj číst. [1]

DSP, resp. digital signal processing bloky slouží ke zpracování digitálního signálu. V těchto blocích jsou implementované funkce AND, OR, NAND, NOT, násobičkya sčítačky. Mají nízkou spotřebu. DSP bloky jsou často umístěny geograficky blézko BRAM, které slouží jako „mezipaměti“ (buffer). [1]

Procesor implementovaný do struktury FPGA snižuje časové zpoždění při obsluhování FPGA. [1]

Digital Clock Manager slouží k vytvoření jiného, resp. nižšího taktovacího signálu CLK,oproti původnímu zdrojovému CLK, pro různé bloky v FPGA. [1]

Multi-Gigabit Transcievers slouží k přenosu dat takovým způsobem, aby došlo k minimalizaci vlivu ručení na přenášená data. Obecně obstarávají optimální serializaci a paralelizaci dat. [1]

Struktuře FPGA, která je obsauje všechny zdroje a funkcionalitu, potřebné pro kompletní realizaci aplikace, se nazývá *platform FPGA*.

3.3 Programování

3.3.1 Forma tvorby algoritmu pro FPGA

K programování, resp. konfiguraci FPGA je možné přistupovat z několika úrovní. Jednou z využívaných metod popisu požadovaného HW na FPGA je popis struktury/toku signálu obvody (structural/data flow circuits). K tomuto popisu je využíváno jazyků HDL, VHDL a Verilog (Hardware Description Language, VSIC HDL). V těchto jazycích je využíváno logických členů AND, OR, NOT nebo bloků sčítaček a násobiček. Forma popisu, jež naopak využívá vyššího programovacího jazyka než HDL je nazývána metoda popisu chování obvodů (behavioral circuits). Zatímco HDL slouží k popisu hardware s využitím nízké míry abstrakce, popis ve vyšších programovacích jazycích, které popis pomocí behavioral circuits umožňuje, je pro programátory značně příjemnější, protože využívá běžných procedurálních programovacích jazyků jako je C, C++ nebo Python. Tyto jazyky jsou následně přeloženy do HDL. Po překladu do HDL pomocí *high level synthesis* (HLS) jsou provedeny kroky *synthesis* (syntéza), *place-and-rout* (umístění-a-pospojování) a *bitgen* (generace bitstreamu). [1]

Při použití HLS může vzniknout situace, že bude vytvořen algoritmus, který bude takovým způsobem komplexní, že ho nebude možné syntetizovat na FPGA. Oproti tomu při použití popisu pomocí structural/data flow circuits, je prakticky vždy algoritmus syntetizovatelný. [1]

V praxi je k tvorbě algoritmů často využíváno vyšších programovacích jazyků a HLS, protože je tento přístup pro značný počet vývojářů SW srozumitelnější. Dalším častým přístupem v praxi je použití specializovaných SW jako je MATLAB™ a Simulink, které při použití odpovídajících balíčků jsou schopny přeložit vytvořený algoritmus do HDL, který je poté možné dále zpracovat a použít pro konfiguraci FPGA.

3.3.2 Konverze HDL na konfigurační Bitstream

V části *Forma tvorby algoritmu pro FPGA* byly představeny dvě hlavní formy tvorby algoritmu pro FPGA. Tyto formy je však pro realizaci na FPGA nutné zpracovat.

Všechny vyšší úrovně reprezentace algoritmů jsou převedeny na HDL. Následným krokem je *syntéza* (synthesis), která slouží k převodu HDL na tzv. *netlist*. Při převodu je HDL převáděna na logické členy AND, OR apod. [1]

Po vytvoření netlistu je nutné rozhodnout, jakým způsobem je možné a výhodné realizovat jednotlivé bloky v logických buňkách a LUT. Konečné sloučení členů závisí na rozsahu vstupů realizovatelných LUT. Proces seskupování logických členů a určování funkce LUT se nazývá mapování (MAP). Výsledkem MAP je opět netlist. Tento netlist však reprezentuje FPGA členy (LUT, klopné obvody apod.). [1]

Po mapování následuje proces umisťování (placement), při kterém je rozhodováno, které logické bloky budou realizovat FPGA členy, získané v kroku MAP. [1]

Bloky, které jsou umístěné ve struktuře FPGA je nutné spojit pomocí dostupných propojů na FPGA. Proces spojování a optimalizace propojů takovým způsobem, aby bylo minimalizováno časové zpoždění signálu, se nazývá *routing*. Obvykle se proces slučuje s MAP do jedné fáze a nazývá se *place-and-route* (PAR). [1]

Posledním krokem je vytvoření binárního souboru, nazývaného *bitstream*, který je poté vkládán do FPGA. Tento proces převede netlist z kroku PAR na nastavení SRAM v jednotlivých logických buňkách FPGA tak, aby byl vytvořen požadovaný design v FPGA. Proces také převede konfiguraci propojů a propojovacích matic do SRAM, ovládající příslušné propoje a matice. [1]

Soubor *bitstream* je poté možný pomocí daného nástroje „nahrát nakonfigurovat jím FPGA“.

Obr. 3 - 4 Blokové schéma převodu aplikace, naprogramované v procedurálním jazyce, na bitstream, který je vhodný pro konfiguraci FPGA.

3.4 Spotřeba

FPGA je využíváno pro akceleraci aplikací také pro svou nízkou spotřebu energie. Oproti ASICs však FPGA má stále značnější spotřebu, proto je podnikán výzkum, který má za cíl jejich energetickou náročnost snížit ale zachovat jejich výkon a spolehlivost.

Nižší potřebný výkon pro realizaci nepohonářské aplikace podporuje výzkum a článek [4], ve kterém autoři představují svoji práci, v níž realizovali HW hru. Ve hře je hlavním úkolem aplikace výpočet stínů

a odrazů materiálů. Způsob vykrlesení, který je v aplikaci použit je nazýván *ray tracing*. Ray tracing je označován jako výpočetně náročný způsob, který není vhodný pro on-line aplikace ale pro vykreslování nepohyblivých obrazů, které není nutné zobrazovat v reálném čase. [5]

Autoři v textu popisují, že v případě využití FPGA pro výpočty v reálném čase byla jeho spotřeba 660 mW. Hru autoři vyzkoušeli spustit také na CPU platformě skládající se z Ryzen™ 4900H 8-core/16 threads 64-bit CPU @ up to 4,4 GHz clock. V případš testování na CPU byla indikována spotřeba 33 W. Tudíž při použití FPGA spotřeba klesla přibližně 50x. [4].

I přes nízkou spotřebu energie v FPGA jsou prováděny výzkumy, jak minimalizovat disipaci elektrické energie v podobě tepla a přiblížit se tak energetické náročnosti ASICs.

Disipace energie v FPGA je rozdělena na statickou a dynamickou.

Statická disipace je způsobena zbytkovým proudem tranzistorů ve vypnutém stavu mezi drain a source elektrodou, mezi gate a drain a jevem nazvaným gate direct-tunneling. [6]

Dynamická disipace je způsobena spínacími a vypínacími ztráty použitých tranzistorů (obvykle CMOS) a je závislá na použitém napětí, frekvenci a kapacitě přechodů, kterou je třeba nabít a vybít při spínání a vypínání tranzistorů. [6]

3.5 Výpočetní výkon a propustnost

3.6 Využití

Programovatelná logická hradlová pole se pro svoji nízkou spotřebu, vysoký výpočetní výkon a klesající cenu elektroniky začínají využívat mnohem častěji v mnoha odvětví, ve kterých bylo doposavaď využíváno CPU a GPU. Aplikace FPGA je možné v rámci této práce rozdělit na nepohonářské a pohonářské.

3.6.1 Aplikace v nepohonářských odvětví

Díky univerzalitě FPGAs je možné je využít v mnoha aplikacích různých odvětví. Stále se zvyšující požadavky na výpočetní výkon urychlují nasazování FPGAs do provozů, kde jsou v současné době instalovány CPU nebo GPU.

Poptávka po dostupnosti FPGA způsobila vznik Cloud služeb, které nabízí FPGA výkon on-demand. Jedním z velkých poskytovatelů je Amazon Web Services (AWS), který nabízí FPGA akceleraci v Cloudu. Tuto službu ocení především aplikace, které nejsou vázány na reálný hardware ale potřebují pouze dostupný výpočetní výkon, který mohou v průběhu tvorby, debuggingu či realizace aplikace měnit bez nutnosti pořizování výkonných a někdy drahých FPGA desek. Více o *Amazon EC2 F1 Instances* služby virtuálních FPGA je dostupné na [7].

Existuje mnoho výpočetně náročných aplikací jako jsou např. výpočty finančních modelů pro ekonomiku, výpočty pro bioinformatiku, seismické modelování při hledání vzácných surovin apod. Více informací o těchto výpočetně náročných aplikacích je možné získat v [8].

Na akceleraci zpracování audiovizuálních děl je převážně určeno GPU. Ovšem pro aplikace, v nichž je vyžadováno zpracování obrazu v reálném čase s minimální spotřebou energie a nízkou hmotností aplikace, je často využíváno FPGA. Aplikace využití FPGA pro vozidla, která analyzují okolní prostor jsou popsány v [9]. Tyto aplikace nesou souhrnný název „inteligennt spaces applications“. Obvykle je pro analýzu okolního prostoru využíváno více kamer, z nichž každá obsahuje vlastní výpočetní jádro (FPGA). Díky tomu výpočetně náročné aplikace, jako např. analýza hloubky obrazu pro rozpoznání objektů, probíhá v FPGA a ostatní nenáročné výpočty a řízení v SW. [9]

Protože momentálním trendem je snižování energetické náročnosti a zvyšování výpočetního výkonu

dochází neustále k vývoji nových aplikací, které využívají FPGA pro akceleraci výpočetně náročných kroků, proto není možné všechny v tomto textu obsáhnout.

3.6.2 Aplikace v elektrických pohonech

V některých případech je elektrický pohon rozměrná a finančně náročná sestava, proto zkoumání určitých kritických stavů těchto soustav by mohlo být ekonomicky i technicky nevýhodné. V tomto případě je vhodné vytvořit přesný matematický model jednotlivých analyzovaných součástí a nezbytné náročné výpočty akcelerovat pomocí FPGA. Na základě odezvy modelu je poté možné analyzovat stavy, které by v případě analýze na reálném stroji mohly způsobit jeho destrukci či částečnou ztrátu funkčnosti. Proto se v průmyslu využívá Hardware-in-the-loop simulation (HILS), kdy je vytvořen požadovaný matematický model, který poskytuje elektrické signály do testovaného systému a na základě jeho reakce je možné vyhodnotit díky matematickému modelu jakým způsobem by se choval reálný modelovaný systém. [9], [10]

Kromě HIL simulace je možné FPGA využít také pro řízení elektrických pohonů. Možnosti realizace řízení AC elektrických strojů pomocí FPGA a analogově digitálních převodníků (ADC) jsou prezentovány v [11]. V dokumentu jsou popisovány tři realizace řízení, resp. regulace pohonu. Nejprve byla regulace realizována pomocí hystérézních on-off regulátorů, následně byly použity PI regulátory. Pomocí nich byl pohon regulován na základně měření a změny vektoru statorového proudu, resp. jeho složek $\alpha\beta$ po aplikování Clarkové transformace. Jako poslení prezentovaný způsob autoři realizovali model ovládání synchronního motoru na základě prediktivních regulátorů. [11]

Všechny prezentované způsoby regulace v [11] byly před syntézou realizovány v prostředí MATLAB™ a Simulink. Tento způsob tvorby modelů a algoritmů je v praxi upřednostňován, protože umožňuje i expertům na řízení a regulaci pracovat na dané problematice bez znalostí mikroelektroniky, programování v HDL a způsobu fungování FPGA. Oproti tomu je třeba zvážit, jaké jsou požadavky na rychlost, výkonnost a optimalizované řízení aplikace a zdali použití předpřipravených knihoven a zjednodušených nástrojů nebude mít příliš značný vliv na rychlost výpočtu a tudíž zpracování dat a řízení v reálném čase. [11]

4 Vývojová deska Digilent Zybo

Vývoj akcelerovaných aplikacích je možné realizovat na realativně velikém množství dostupného HW. V některých případech je design vývojových desek dokonce výrobcem uveřejňován a tudíž v případě dostatečných znalostí je dokonce možné si sestavit vlastní HW s dostupných komponent takovým způsobem, aby vyhovoval požadované embedded aplikaci. Výhodné ovšem je využít již připravená řešení vývojových desek, které zjednodušují prvotní tvorbu aplikace.

V této práci je realizován vývoj aplikace na vývojové desce *Digilent ZYBO Zynq-7000 ARM/FPGA SoC Trainer Board* od firmy Digilent. [12] Jedná se o model vývojové desky, který byl nahrazen novějšími variantami s označením *ZYBO Z7-10* a *ZYBO Z7-20*, které jsou stále v aktivním prodeji. Hlavním rozdílem představených desek je verze Zynq čipu, který v moderních deskách disponuje ARM procesorem s vyšší taktovací frekvencí a s modernějším FPGA s vyšším počtem LUT, klopných obvodů a s rozsáhlejší pamětí RAM apod. Bližší porovnání specifikací těchto desek je dostupné na [13].

V další části textu jsou představeny významné komponenty vývojové desky *Digilent ZYBO Zynq-7000 ARM/FPGA SoC Trainer Board*.

4.1 Základní přehled

4.1.1 CPU a FPGA čip

Hlavní částí vývojové desky je čip, obsahující FPGA a CPU jednotky zakomponované v jedné polovodičové struktuře. Jak již bylo zmíněno v části *Hardware Accelerated Applications*, tato struktura se nazývá heterogenní.

Deska obsahuje čip Xilinx Zynq-7000 (typ XC72010), který umožňuje pro vývoj aplikací použít SDK od firmy Xilinx. V tomto čipu je integrován dvou jádrový procesor ARM Cortex-9, který slouží jako host akcelerovaných aplikací na Xilinx FPGA sedmé série. Detailní schéma blokové architektury SoC s označním sběrnic a komunikace jednotlivých částí čipu je zobrazené na obr. 4 - 1.

Z naznačené architektury je možné vyvodit, že se SoC skládá ze dvou hlavních částí, které je možné dále rozdělit na jednotlivé bloky:

- Processing System (PS),
  - Application processor unit (APU),
  - Memory interfaces,
  - I/O peripherals (IOP),
  - Interconnect,
- Programmable Logic (PL).

Blok PS

Blok PS se skládá z dílčích bloků, které neslouží k akceleraci aplikací, ale k podpoře běhu hostitelského programu. Blok PS reprezentuje prakticky celou architekturu čipu vyjma části věnované PL.

Blok APU

Blok APU obsahuje CPU Cortex-A9 a další podpůrné bloky jako např. přímý přístup do paměti (DMA controller), Genral interrupt controller (GIC) pro maskování a ovládání přerušení, watchdog a další podpůrné bloky.

Blok Memory interfaces

Memory interfaces slouží k přístupu APU a PL k pamětím typu DDR3, DDR3L, DDR2 a LPDDR-2. Je

možné také vybrat, zda šířka sběrnice bude 16 nebo 32 bitů. K dispozici jsou zakoponované kotroléry přenosu dat pro optimalizaci rychlosti, Static Memory Controller nebo Quad-SPI Controller.

Blok IOP

IOP se skládá ze standardizovaných rozhraní vhodných pro průmyslovou komunikaci. Obsahuje nař. GPIO, Gigabit Ethernet, dva bloky USB Controller, dva bloky SD/SDIO Controller pro bootování SD karty, dva bloky SPI Controller, dva bloky CAN Controller, dva bloky UART Controller a dva bloky I2C Controller.

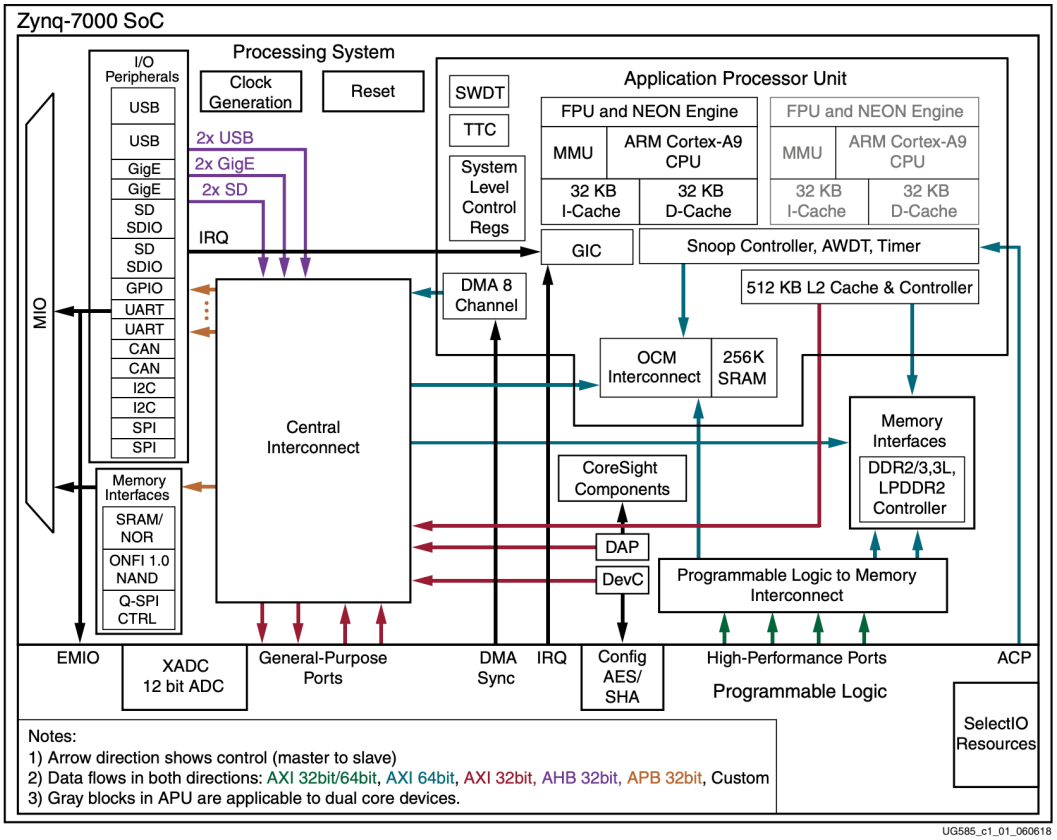
Blok Interconnect

Blok Interconnect, resp. na obr. **??** označený Central Interconnect slouží k propojení jednotlivých bloků SoC dle požadované technologie a rychlosti.

Blok PL

Blok PL reprezentuje logické programovtelné pole (FPGA), v němž jsou zakomponovány další podpůrné bloky jako např. blok zpracování digitálních signálů, řízení taktovacích hodin, analogově digitální převodník (ADC) apod.

Veškeré technické specifikace, složení a parametry jmenovaných bloků a jsou uvedeny v [14].



Obr. 4 - 1 Detailní schéma čipu Zynq-7000, umístěného na vývojové desce Digilent ZYBO Zynq-7000 ARM/FPGA SoC Trainer Board. (převzato z [14])

4.1.2 Uspořádání vývojové desky Zybo Zynq-7000

Tab. 4 - 1 Popis označených komponent na vývojové desce Digilent Zybo Zynq-7000.

označení	popis	poznámka
0	Power Switch	galvanické sepnutí napájecího obvodu
7	1	1

4.2 Možné alternativy
5 Matematický model stroje
5.1 Představení stroje
5.2 Odvození modelu
5.3 Optimalizace modelu
6 Program pro FPGA a CPU
6.1 Použité nástroje
6.1.1 Xilinx Vivado
6.1.2 Xilinx Vitis
6.1.3 Petalinux
6.1.4 Programovací prostředí – operační systém Linux
6.2 Tvorba HW architektury Xilinx Vivado
6.3 Tvorba Petalinux
6.4 Tvorba SW pro CPU a FPGA
7 Představení pracoviště
8 Dosažené výsledky

Závěr

Aliquam dapibus leo velit, ultrices eleifend mi feugiat eget. Aliquam euismod facilisis turpis, nec lobortis libero aliquet sit amet. Aenean suscipit ante eget ipsum viverra hendrerit. Ut sed massa sed nisi tempus dapibus in eu enim. Nullam vitae odio laoreet, malesuada purus non, faucibus orci. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam eget odio quis enim laoreet imperdiet nec eu nunc. Maecenas ut consequat purus. Duis faucibus risus nec metus cursus placerat. Phasellus sapien justo, laoreet in pulvinar ut, maximus nec velit.

Literatura

[1] SASS, Ronald; SCHMIDT, Andrew G. *Embedded systems design with platform FPGAs: principles and practices*. Boston: Morgan Kaufmann, 2010. ISBN 0123743338.

[2] ANDINA, Juan J. R.; TORRE ARNANZ, Eduardo de la; VALDÉS PEÑA, María D. *FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics*. 1. vyd. Bosa Roca: CRC Press, 2017;2015; ISBN 9781439896990;1439896992;

[3] What Is Accelerated Computing, and Why Is It Important? In: *Xilinx, Inc.* [Online]. [B.r.] [cit. 2022-11-03]. Dostupné z: https://www.xilinx.com/applications/adaptive-computing/what-is-accelerated-computing-and-why-is-it-important.html.

[4] Sphery vs. shapes, the first raytraced game that is not software. In: *GitHub* [online]. 14. 07. 2022 [cit. 2022-11-06]. Dostupné z: https://github.com/JulianKemmerer/PipelineC-Graphics/blob/main/doc/Sphery-vs-Shapes.pdf.

[5] Ray tracing (graphics). In: *Wikipedia* [online]. 06. 11. 2022 [cit. 2022-11-06]. Dostupné z: https://en.wikipedia.org/wiki/Ray_tracing_(graphics).

[6] GROVER, Naresh; K.SONI, M. Reduction of Power Consumption in FPGAs - An Overview. *International journal of information engineering and electronic business*. 2012, roč. 4, č. 5, s. 50–69. ISBN 2074-9023.

[7] Amazon EC2 F1 Instances. In: *Amazon AWS* [online]. [B.r.] [cit. 2022-11-06]. Dostupné z: https://aws.amazon.com/ec2/instance-types/f1/.

[8] VANDERBAUWHEDE, Wim; BENKRID, Khaled. *High-Performance Computing Using FPGAs*. Springer Publishing Company, Incorporated, 2013. ISBN 1461417902.

[9] RODRÍGUEZ-ANDINA, Juan J.; VALDÉS-PEÑA, María D.; MOURE, María J. Advanced Features and Industrial Applications of FPGAs—A Review. *IEEE Transactions on Industrial Informatics*. 2015, roč. 11, č. 4, s. 853–864. Dostupné z DOI: 10.1109/TII.2015.2431223.

[10] Hardware-in-the-Loop (HIL) Simulation. In: *The MathWorks, Inc.* [Online]. [B.r.] [cit. 2022-11-06]. Dostupné z: https://www.mathworks.com/discovery/hardware-in-the-loop-hil.html.

[11] NAOUAR, Mohamed-Wissem; MONMASSON, Eric; NAASSANI, Ahmad Ammar; SLAMA-BELKHODJA, Ilhem; PATIN, Nicolas. FPGA-Based Current Controllers for AC Machine Drives—A Review. *IEEE Transactions on Industrial Electronics*. 2007, roč. 54, č. 4, s. 1907–1925. Dostupné z DOI: 10.1109/TIE.2007.898302.

[12] DIGILENT, Inc. Zybo. In: *Digilent Documentation* [online]. [B.r.] [cit. 2022-11-11]. Dostupné z: https://digilent.com/reference/programmable-logic/zybo/start.

[13] DIGILENT, Inc. Zybo Z7 Migration Guide. In: *Digilent Documentation* [online]. [B.r.] [cit. 2022-11-11]. Dostupné z: https://digilent.com/reference/programmable-logic/zybo-z7/migration-guide.

[14] XILINX, Inc. Zynq-7000 SoC Technical Reference Manual. In: *Xilinx Documentation* [online]. 02. 04. 2021 [cit. 2022-11-11]. Dostupné z: https://docs.xilinx.com/v/u/en-US/ug585-Zynq-7000-TRM.

[15] XILINX, Inc. SoCs with Hardware and Software Programmability. In: *Xilinx Website* [online]. [B.r.] [cit. 2022-11-11]. Dostupné z: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html.

Příloha A: Seznam symbolů a zkratek

A.1 Seznam symbolů

$\vec{F}$ (N) vektor síly

A.2 Seznam zkratek

DCM DC Master