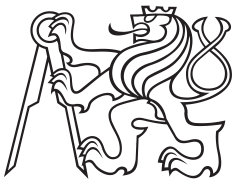
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Implementace komunikační knihovny v LabView FPGA

Dominik Siegel

Vedoucí: Ing. Jan Sobotka, Ph.D.
Obor: Otevřená informatika
Studijní program: Počítačové systémy
Květen 2018

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Siegel** Jméno: **Dominik** Osobní číslo: **457091**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačové systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Implementace komunikační knihovny v LabView FPGA

Název bakalářské práce anglicky:

Communication Library Implementation in LabView FPGA

Pokyny pro vypracování:

1. Seznamte se programovacím prostředím LabView FPGA.
2. Prozkoumejte dostupné možnosti digitální komunikace (UART, SPI, I2C, PWM, ...) pomocí univerzálních digitálních modulů NI 9425, NI 9426, NI 9476 a NI 9477.
3. Dle dohody s vedoucím implementujte knihovnu komunikačních bloků (VI) pro obecné použití.
4. Účelem knihovny je umožnit snadné připojení specializovaného hardware k testovacímu systému složeného z produktů firmy National Instruments.
5. Funkčnost implementovaných komunikačních rozhraní otestujte.

Seznam doporučené literatury:

[1] Conway, Jon and Watts, Steve: A software engineering approach to LabVIEW. Upper Saddle River, NJ: Prentice Hall, Professional Technical Reference, c2003, xiii, 221 p. ISBN 01-300-9365-3.
[2] NI LabVIEW High-Performance FPGA Developer?s Guide National Instruments, 2014. http://download.ni.com/pub/gdc/tut/labview_high-perf_fpga_v1.1.pdf
[3] Rumsey, F. and Watkinson, J.: Digital Interface Handbook, Elsevier/Focal Press, 2004.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Sobotka, katedra měření FEL

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

Datum zadání bakalářské práce: **09.01.2018** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce:
do konce letního semestru 2018/2019

Ing. Jan Sobotka podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. 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

Poděkování

Děkuji své rodině za podporu. Děkuji také svému vedoucímu, Ing. Janu Sobotkovi, Ph.D., za odborné vedení, za rady a pomoc při zpracování této bakalářské práce.

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, 25. května 2018

.....

Abstrakt

Tato práce se zabývá implementací digitální komunikační knihovny pro jednodušší propojení hardwaru s testovacím systémem složeného z produktů firmy National Instruments. V práci je představeno grafické programování. Dále je popsáno vývojové prostředí LabVIEW, ve kterém se tato práce implementovala. Následně je popsána implementace jednotlivých částí knihovny. Na závěr jsou prezentovány výsledky z testování implementované knihovny.

Klíčová slova: LabVIEW, FPGA, schéma, komunikace

Vedoucí: Ing. Jan Sobotka, Ph.D.

Abstract

This thesis deals with an implementation of a digital communication library for easier hardware interconnection with a test system comprised of National Instruments products. Graphical programming is presented in the thesis. Next, the LabVIEW development environment, in which this work was implemented, is described. Subsequently, the implementation of the individual parts of the library is described. Finally, the results from the testing of the implemented library are presented.

Keywords: LabVIEW, FPGA, diagram, communication

Title translation: Communication Library Implementation in LabView FPGA

Obsah

Kapitola 1

Úvod

V této bakalářské práci se budu zabývat digitální komunikací. Přenos informace v elektronických zařízeních je možná pomocí analogového nebo digitálního signálu. Cílem bakalářské práce je implementovat komunikační knihovnu digitálních komunikací UART, SPI, I^2C a PWM. Implementace je provedena v grafickém programovacím jazyce LabVIEW FPGA. Knihovna umožní jednodušší zapojení specifického hardwaru k testovacímu systému využívající produkty firmy National Instruments. Součástí práce je otestování implementované knihovny s hardwarem dostupném v laboratoři.

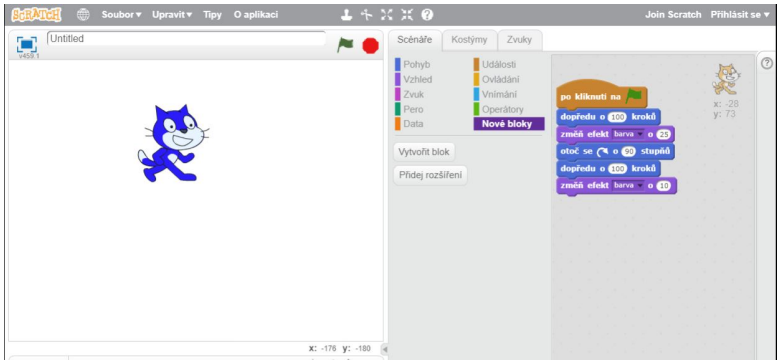
Kapitola 2

Grafické programování

Prvním vizuálním kódováním byly čárové kódy reprezentující číselné označení výrobku. V dnešní době chytrých telefonů se rozšířilo mezi jejich uživateli používání QR (Quick Response) kódů, který je schopen uvádět větší množství informací oproti čárovému kódu. Součástí QR kódu může být i například webový odkaz, nebo také krátký funkční kód programu, který může čtecí zařízení interpretovat.[1]

Principem grafického programování je implementace programu za využití obrázků nebo funkčních grafických bloků. Díky vizualizaci programovacích prvků není potřebná znalost programovacího jazyka, do kterého se výsledné schéma bude reprezentovat. Není tedy potřeba znát ani syntaxi daného jazyka.[2]

Toto prostředí poskytuje možnost implementování vlastních programů i dětem nebo lidem mimo informační obory. Pro výuku dětí v programování byla vyvinuta vývojová prostředí s grafickými programovacími bloky. Jedním z nich je například Scratch, který se používá na všech vzdělávacích stupních a v různých předmětech.[3]



Obrázek 2.1: Příklad implementace v online editoru Scratch

2.1 Výhody grafického programování

Grafické programování přináší jednodušší pohled na věc, díky čemuž je lehčí programování hardwaru nebo vestavěných systémů oproti textové implementaci ve VHDL, assembleru či v jiném jazyku. Není nutné znát syntaxi jazyka do kterého se výsledný návrh přeloží.[4][5]

Je to podobné jako práce v operačním systému, který se dříve ovládal za pomoci příkazového řádku a uživatel musel používat dané textové příkazy. S příchodem ovládání pomocí ikon se stala práce v operačním systému více intuitivní a uživatel nemusí znát textové příkazy. [4][5]

2.2 Nevýhody grafického programování

Nevýhodou grafického programování je potřeba editoru a kompilátoru na konkrétní operační systém. Oproti textovému programovacímu jazyku, například jazyku C, jehož kód je možné psát v kterémkoliv textovém editoru bez ohledu na operační systém. Dále je potřeba interpretovat navržené schéma do daného programovacího jazyka.

Kapitola 3

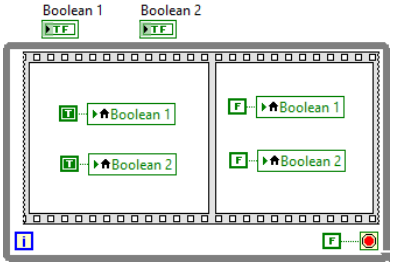
LabVIEW FPGA

3.1 LabVIEW

LabVIEW je vývojové prostředí vyvinuté firmou National Instruments. V LabVIEW se schéma navrhuje grafickým programováním. Tedy místo klasického psaní kódu se kreslí blokové diagramy.[6]

Navržené schéma v LabVIEW se spustí bez nutnosti kompilace. Všechna vytvořená schémata vytvořená v LabVIEW mají formát VI neboli Virtual Instrument. Kromě blokového diagramu obsahuje VI i front panel, na který je možné umisťovat ovladače nebo výstupní informační indikátory dat, které může na tomto panelu uživatel zadávat nebo číst.

LabVIEW využívá jiného programovacího paradigmatu a to tok dat. Data jsou zpracovávána zleva doprava. Daná operace se provede až v době, kdy má na vstupu všechna data. Pokud chceme dosáhnout serializované funkce, využijeme struktury Flat Sequence nebo Stacked Sequence, která zajistí sekvenční vykonání po sobě jdoucích rámců sériově od nejvíce vlevo po nejvíce vpravo. Obsah rámce znovu závisí na toku dat a případné obsažené paralelizace.[7]



Obrázek 3.1: Serializace a paralelizmus

Na obrázku 3.1 je vidět sériový zápis hodnot true a false do lokálních proměnných Boolean 1 a Boolean 2. V daném rámci však proběhne zápis paralelně a není možné přesně určit, do které z proměnných se hodnota

zapíše dříve.

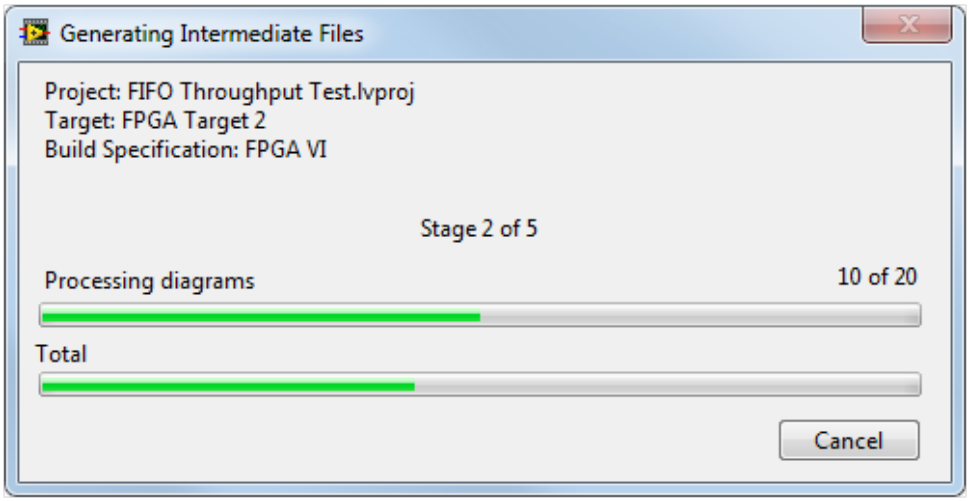
3.1.1 LabVIEW FPGA

LabVIEW FPGA je variantou LabVIEW pro návrh hardwaru pro FPGA. Návrhy vytvořené v LabVIEW FPGA se spouštějí na požadovaném hardwaru. Aby to bylo možné, musí návrh projít kompilačním procesem a vygenerováním bitfile.

Překladač lze mít na stejném počítači, pokud splňuje požadavky daného překladače jako je například podporovaný operační systém. Nebo lze přistupovat k překladači vzdáleně, a to přes vzdálený počítač s nainstalovaným překladačem, vzdálený kompilační server nebo kompilační cloudový servis.[8]

V mém případě se jednalo o překladač Xilinx s podporovaným operačním systémem Windows 7, který pracoval na kompilačním serveru. Dále pak Compile Worker, který byl spuštěný pod operačním systémem Linux. Díky této kombinaci lze snížit dobu překladu průměrně od 20 do 50 procent oproti Compile Worker spuštěný pod Windows. Kompilaci vytvořeného schématu v LabVIEW FPGA se spustí kliknutím na šipku v levém horním rohu návrhu. Pokud bitfile neexistuje nebo se schéma upravilo, pak se spustí kompilace.[9][10]

LabVIEW nejprve přeloží grafické schéma do textového kódu VHDL obrázek 3.2. Během této fáze mohou být detekovány chyby, a to například nepřípustné VI obsahující nepřípustné bloky nebo vložené VI v single-cycle časovaném loopu. Poté přichází na řadu překladač Xilinx, který vygenerovaný VHDL kód optimalizuje, redukuje a syntetizuje do FPGA obvodu. Výstupem je bitfile, který se nahraje do příslušného hardwaru.[8][11]



Obrázek 3.2: Generování pokročilých souborů [8]

3.2 NI 9144

Jedná se o šasi s EtherCAT slave a rozšířením až o 8 modulů. Toto šasi jsem rozšířil o vstupní modul NI 9426 a výstupní modul 9477, které jsem použil pro testování. NI 9144 je kompatibilní s kterýmkoliv CompactRIO či PXI. Disponuje také FPGA Spartan-3 2M do kterého se nahrává výsledný bitfile.[12]

Obrázek 3.3: NI 9144 [12]

Toto zařízení se doporučuje napájet zdrojem 24 VDC. Přípustné napájecí napětí šasi se pohybuje od 9 V až do 30 V. Disponuje 100BaseTX Ethernet, pomocí kterého komunikuje s EtherCAT masterem.[13]

Kapitola 4

Komunikační knihovna

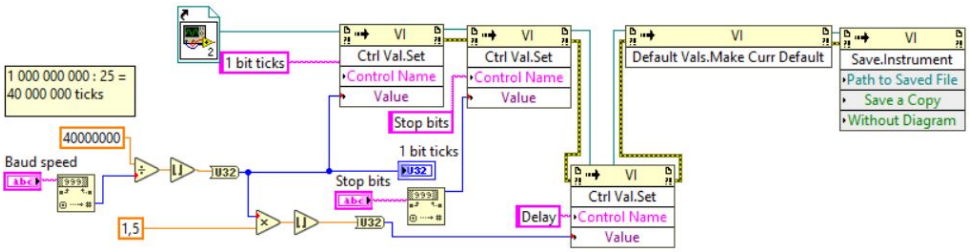
4.1 UART

V UART (Universal Asynchronous Receiver and Transmitter) je udávána rychlost komunikace v baudech. Vytvořil jsem schéma nastavení Setup.vi, ve kterém se vypočítají parametry požadované komunikace. Vstupní parametry jsou rychlost v baudech a počet stop bitů viz. Tabulka 4.1.

Parametr	Hodnoty
Baud speed	9600; 19200; 38400; 57600; 115200
Stop bits	1; 2

Tabulka 4.1: Nastavitelné parametry UART

Schéma obrázek 4.1 se spouští před kompilací. Toto VI předpočítá, jak dlouho trvá jeden bit a hodnotu zpoždění. Délku jednoho bitu, zpoždění a počet stop bitů zapíše do příslušných prvků cílového souboru UART.vi. Zapsané hodnoty jsou uloženy jako defaultní. Pokud by se výpočty prováděli v UART.vi, zabíralo by to tak zbytečné prostředky na příslušné FPGA.



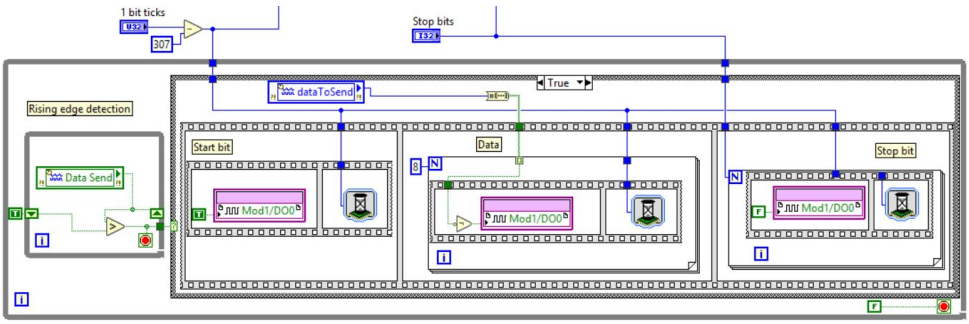
Obrázek 4.1: UART Setup.vi

Implementace UART se dělí na vysílač TX a přijímač RX. Vysílač obrázek 4.2 při detekci dat k odeslání na náběžnou hranu proměnné Data Send vygeneruje start bit následující osmi datovými bity od nejméně významného

po nejvíce významný. Nakonec vygeneruje daný počet stop bitů, kolik bylo nastaveno.

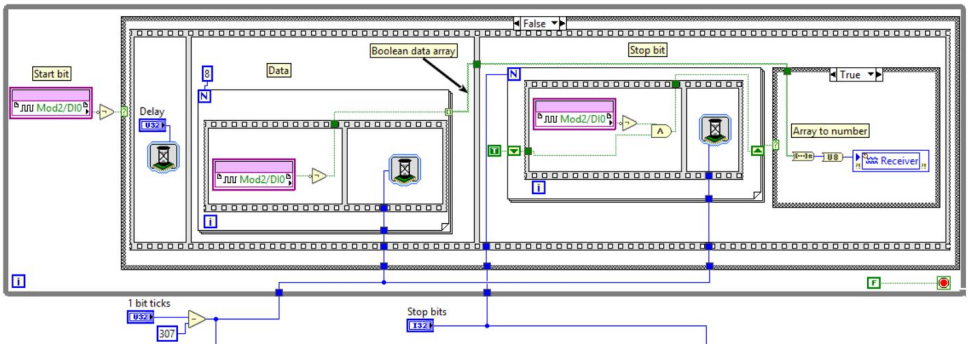
Vstupy	Výstupy
Data to Send Data Send	Receiver

Tabulka 4.2: Vstupní a výstupní proměnné UART



Obrázek 4.2: UART vysílač

Přijímač obrázek 4.3 detekuje start bit. Po jeho detekci čeká přijímač 1,5 násobku bitového času, aby nedošlo k opakovanému čtení start bitu jako datový bit. Osm datových bitů uloží do pole. Po přijetí požadovaného počtu stop bitů se zpracují přijatá data v poli na osmi bitové číslo bez znaménka a znovu se čeká na detekci start bitu.



Obrázek 4.3: UART přijímač

4.2 SPI

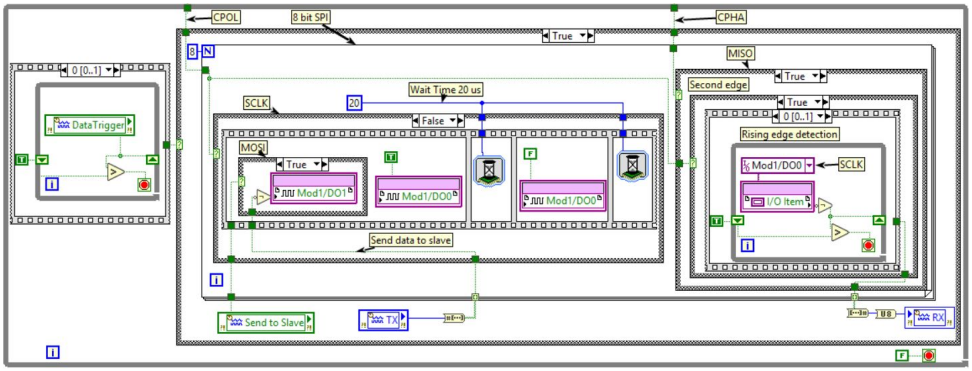
SPI (Serial Peripheral Interface) komunikace se skládá z vodičů *Slave Select* $\overline{SS}$, systémových hodin SCLK, *Master In Slave Out* MISO a *Master Out Slave In* MOSI. Každý slave je propojen samostatným SS, ostatní vodiče se

sdílejí. U SPI komunikace je nastavení polarity CPOL a fáze CPHA hodin. Tabulka 4.3, tím je dosaženo čtyř různých kombinací. Přičemž master se vždy musí přizpůsobit nastavení slave.

Parametr	Hodnoty
CPOL	0; 1
CPHA	0; 1

Tabulka 4.3: Nastavitelné parametry SPI

Master obrázek 4.4 detekuje náběžnou hranu proměnné DataTrigger a poté vyhodnotí stav $\overline{SS}$. Pokud je $\overline{SS}$ aktivní, začne generovat hodinový signál s příslušnou polaritou CPOL. Pokud má master příznak Send to Slave roven hodnotě true, vysílá data na výstup MOSI každým tikem hodin. V opačném případě generuje pouze hodinový signál. Současně čte master na vstupu MISO vstupní data od slave na příslušné hraně dle slave fáze CPHA. Všechna data se přenáší od nejméně významného bitu LSB po nejvíce významný bit MSB.

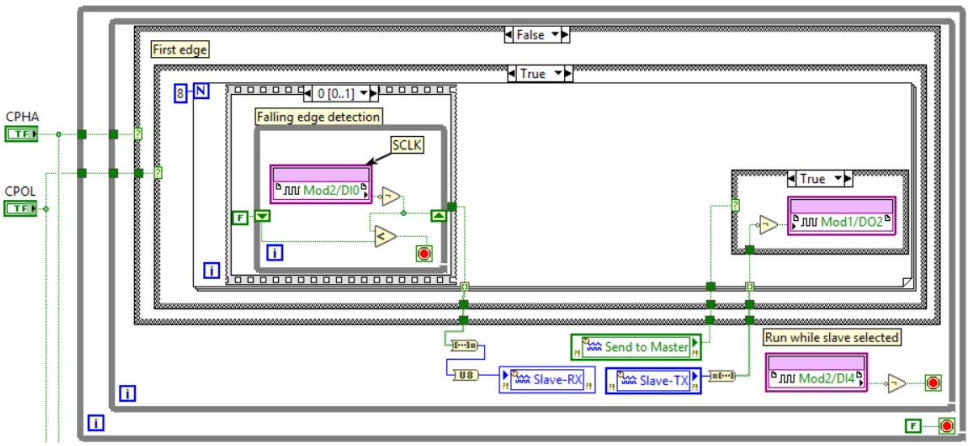


Obrázek 4.4: SPI Master

	Vstupy	Výstupy
Master:	Send to Slave, TX	RX
Slave:	Send to Master, Slave-TX	Slave-RX
	DataTrigger	

Tabulka 4.4: Vstupní a výstupní proměnné SPI

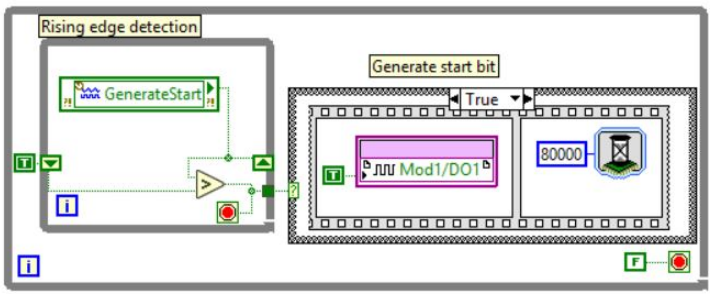
Jak jsem zde již zmínil, slave obrázek 4.5 zpracovává data podle nastavených hodnot CPHA a CPOL. Při CPHA=0 čte slave data z MOSI na první hraně hodinového signálu. Pokud je CPHA=1, pak se data čtou na druhé hraně hodinového signálu. Podle hodnoty CPOL se pak při příslušné hodnotě CPHA čtou data na náběžnou nebo sestupnou hranu hodinového signálu. Pokud má slave data k odeslání, zapisuje je při každém průběhu na MISO výstup.



Obrázek 4.5: SPI Slave CPHA=0 CPOL=1

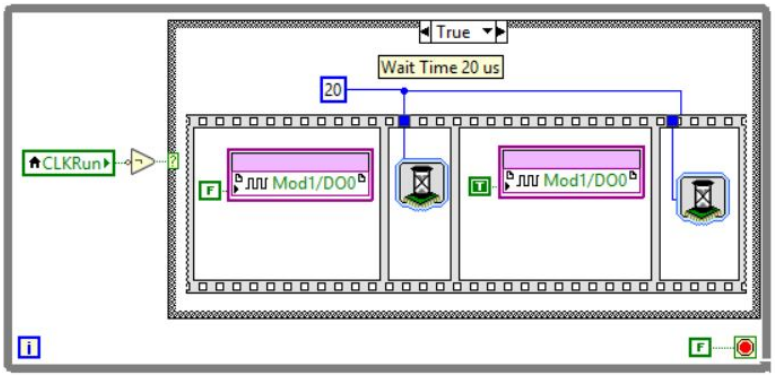
4.3 I²C

Komunikace I²C (Inter-Integrated Circuit) je podobná SPI s jistými rozdíly. Používají se zde pouze dva vodiče, a to na systémové hodiny SCL a data SDA, ke kterým je připojen jak master, tak každý slave. Každý slave má přidělenou svoji sedmi bitovou adresu.



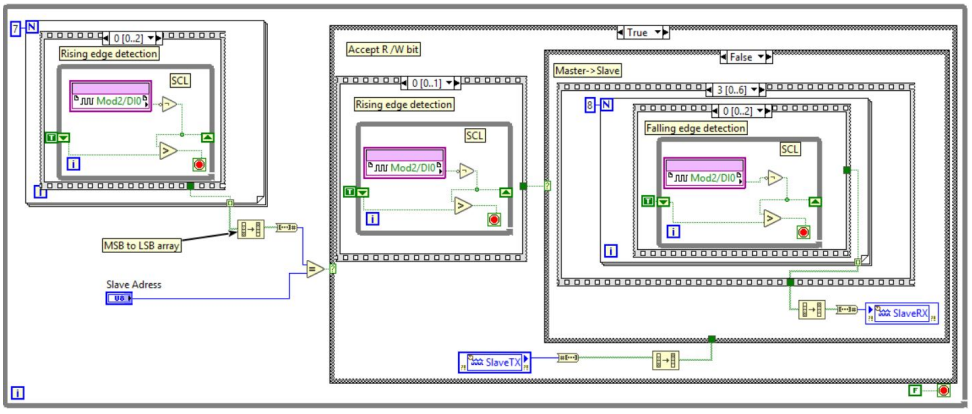
Obrázek 4.6: I²C generování příznaku start bitu

Při detekci náběžné hrany proměnné GenerateStart se vygeneruje příznak start bitu obrázek 4.6. Při příznaku start bitu začne master generovat hodinový signál obrázek 4.7 na SCL. Zápis dat na SDA se provádí, když jsou hodiny v logické 0, což lze detekovat pomocí hradlového detektoru[14] SCL na spádovou hranu, po jejíž detekci následuje logická 0. Čtení dat se provádí, když jsou hodiny v logické 1. Tedy podobným způsobem jsem využil detektoru vzestupné hrany hodinového signálu a po její detekci čtu příslušnou hodnotu z SDA.



Obrázek 4.7: I²C SCL generátor

Jako první po vygenerování start bitu posílá master na SDA sedmi bitovou adresu slave s příznakem čtení R nebo zápisu $\overline{W}$. Protože proměnná Adress je osmi bitová, musí se provést logický posun vlevo o jeden bit. Slave obrázek 4.8 čte data z SDA na vzestupnou hranu, a pokud se zaslaná adresa shoduje s adresou slave, pak po přijetí příznakového bitu čtení či zápisu odešle potvrzení stažením SDA do logické 0.

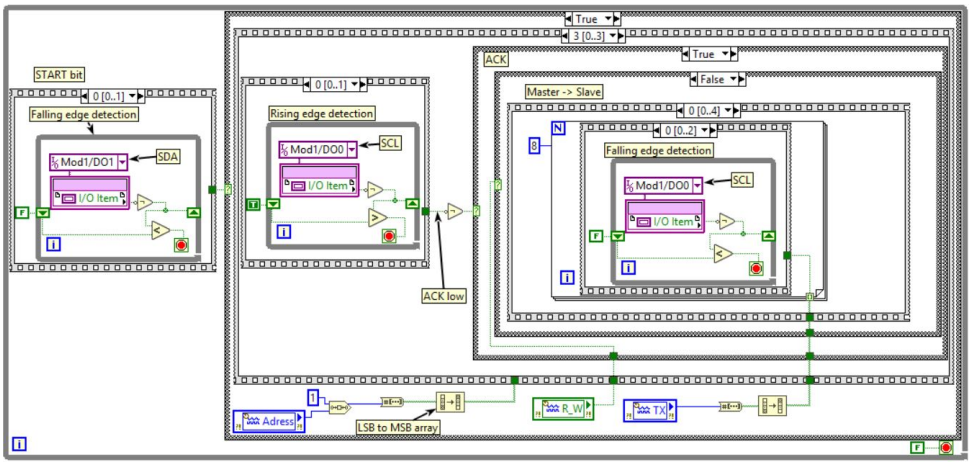


Obrázek 4.8: I²C Slave

Při příznaku čtení zapisuje slave na SDA při sestupné hraně hodin osm bitů dat od nejvíce významného bitu po nejméně významný bit a master obrázek 4.9 tyto data čte po náběžné hraně. V tomto případě nemusí master přijatá data potvrzovat. Při příznaku zápisu zapisuje na SDA master a slave čte. Slave po přijetí dat odešle potvrzení. Pro ukončení komunikace se vygeneruje příznak pro stop bit a ukončí se generování hodinového signálu.

	Vstupy	**Výstupy**
Master:	GenerateStart, Adress, TX	RX
Slave:	SlaveTX	SlaveRX

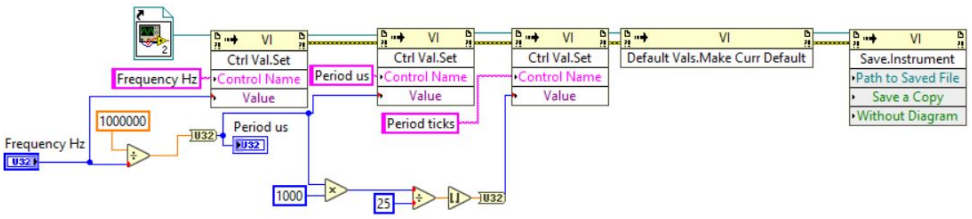
Tabulka 4.5: Vstupy a výstupy I²C



Obrázek 4.9: I²C Master

4.4 PWM

Využil jsem zde podobné řešení jako v případě UART pro výpočet potřebných hodnot pro správné časování PWM (Pulse Width Modulation) signálu. Zde se zadává frekvence v Hz. Po spuštění Setup.vi se vypočítá perioda, díky které se podle dané střídy spočítá, jak dlouho má generátor vysílat logickou 1 a logickou 0, neboli zapnuto či vypnuto. Frekvence a spočítaná perioda se uloží do PWM.vi.



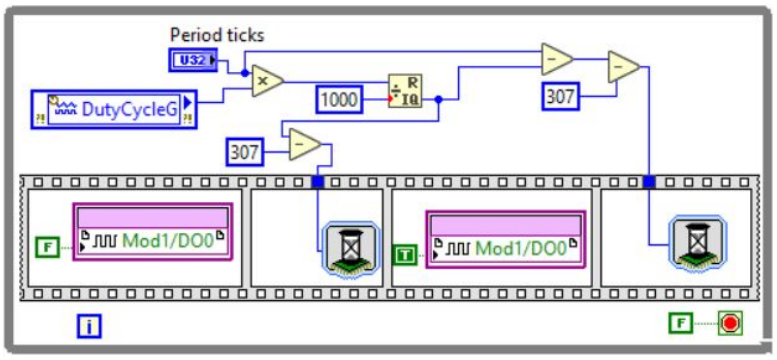
Obrázek 4.10: PWM Setup.vi

Vstupy	Výstupy
DutyCycleG	DutyCycleA, Period

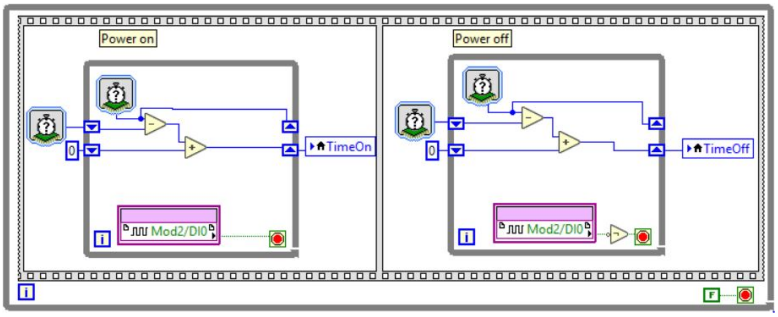
Tabulka 4.6: Vstupní a výstupní proměnné PWM

Úlohu PWM jsem rozdělil na tři části, generátor PWM signálu, přijímač PWM signálu a výpočet periody a střídy přijatého signálu. Generátor signálu obrázek 4.11 vypočítá z periody a ze zadané střídy dobu vysílání logické 1 a logické 0.

Přijímač obrázek 4.12 počítá, jak dlouho přijímá logickou 1 a logickou 0. Tyto hodnoty se ukládají do lokálních proměnných TimeOn a TimeOff.

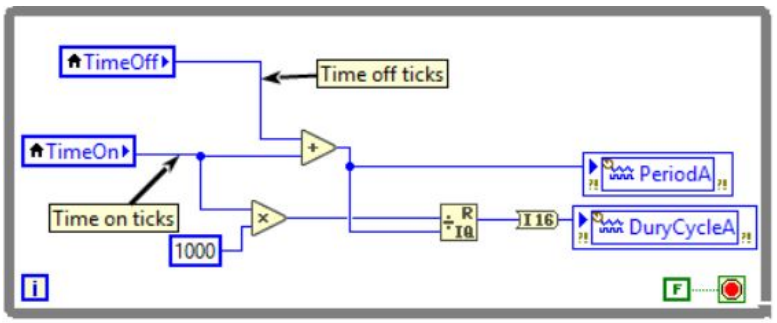


Obrázek 4.11: PWM generátor



Obrázek 4.12: PWM přijímač

Výpočetní část obrázek 4.13 obsahuje operaci dělení, a proto trvá více hodinových tiků. Pro správný chod zařízení ovládané touto PWM je zapotřebí provádět výpočet střídy externě. Výstupem výpočtu je perioda a střída v procentech.



Obrázek 4.13: PWM přijímač - výpočet

Kapitola 5

Testování

Implementovanou knihovnu jsem otestoval na zařízení NI-9144 se vstupním modulem NI-9426 a výstupním modulem NI-9477. Dostupný modul NI-9476 jsem nepoužil z důvodu vysokého obnovovacího času, který u NI-9476 je 500 μs na rozdíl od NI-9477 s obnovovacím časem 8 μs.[15]

Pomocí programu VeriStand a propojením počítače ethernetem s PXI jsem nahrával bitfile do NI 9144. V programu VeriStand jsem přistupoval k jednotlivým vstupním a výstupním proměnným distribuovaných do šasi přes EtherCAT. Pomocí osciloskopu jsem zaznamenal průběhy signálů na vodičích.

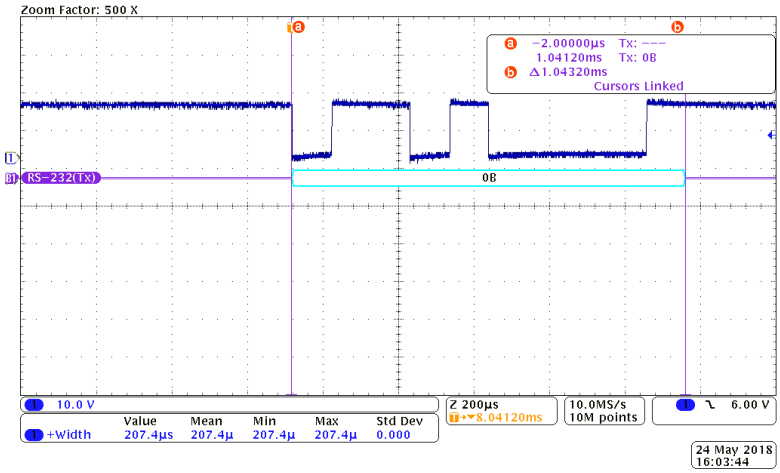
Během testování navržených komunikací jsem detekoval několik nedostatků v implementaci. Ty jsem posléze napravil. Kompilace opravovaných návrhů však proces testování velmi zpomalila.

5.1 UART

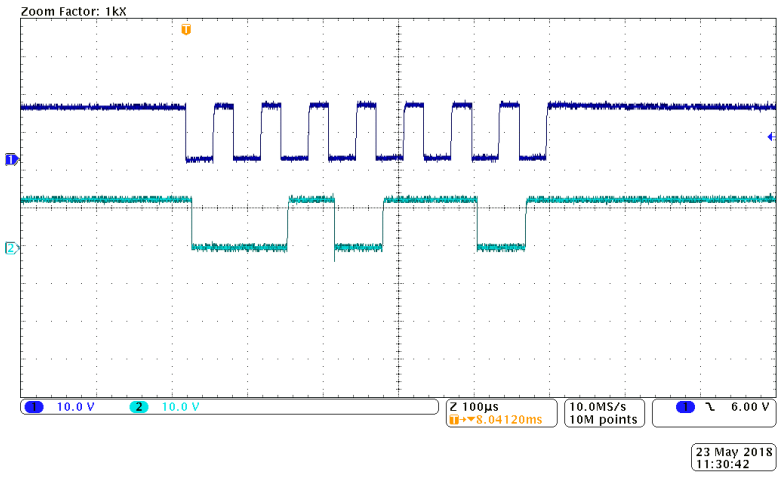
Nastavení Setup.vi byla rychlost 9600 baud a jeden stop bit. Při detekci náběžné hrany proměnné Data Send se vygeneroval start bit následující bity hodnoty 11 od nejméně významného bitu po nejvíce významný bit. Odeslaná data jsou ukončena jedním stop bitem. Na obrázku 5.1 je vidět zaznamenaný průběh signálu společně s dekódovanou hodnotou 11.

5.2 SPI

Nastavená polarita a fáze hodin byla CPOL = 0 a CPHA = 0. Při detekci náběžné hrany na proměnné DataTrigger, aktivním $\overline{SS}$ a hodnotě proměnné Send to Slave rovné true se na MOSI odesílaly bity hodnoty 180, v pořadí od nejméně významného bitu po nejvíce významný bit, při každém hodinovém cyklu. Na obrázku 5.2 je vidět modře SCLK a tyrkysově MOSI.



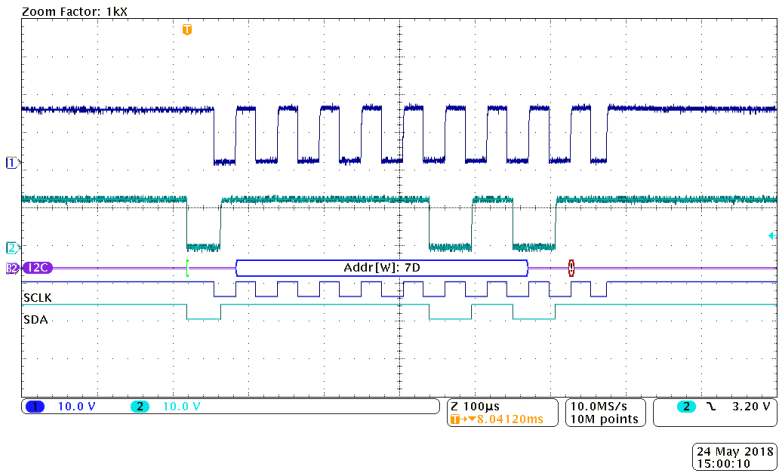
Obrázek 5.1: UART - posílání hodnoty 11



Obrázek 5.2: SPI - posílání hodnoty 180

5.3 I²C

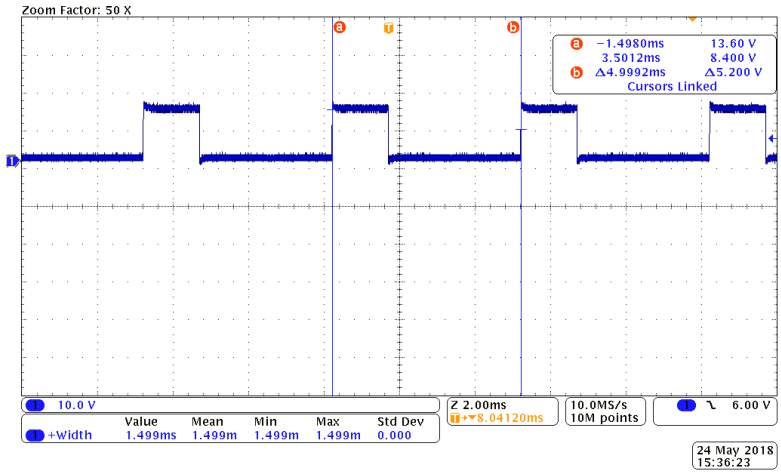
Při náběžné hraně proměnné GenerateStart se vygeneroval příznak start bitu. Poté se při hodnotě logické 0 na SCL mění data na SDA. Zadaná odesílaná adresa byla 125 což je 0x7D. Dále byl zadán příznak zápisu. Adresa slave byla nastavena na hodnotu 3. Na obrázku 5.3 je vidět správně odeslaná adresa s příznakem zápisu. Protože se adresy neshodovali, nedošlo k potvrzení a komunikace se ukončila.



Obrázek 5.3: I²C - adresa 125, příznak zápisu

5.4 PWM

Zadaná frekvence byla 200 HZ. S průběžně měněnou střídou se měnil poměr logické 1 a logické 0. Na obrázku 5.4 jsem zaznamenal signál PWM se střídou 30% a periodou 5 ms.



Obrázek 5.4: PWM - střída 30%

Kapitola 6

Závěr

Cílem této bakalářské práce byla implementace komunikační knihovny obsahující digitální komunikace UART, SPI, I^2C a PWM, které se mi podařili navrhnout ve vývojovém prostředí LabVIEW FPGA. Nejdříve jsem se seznámil se samotným prostředím LabVIEW.

Při vytváření projektu pro FPGA jsem narazil na absenci některých modulů jako je například real-time a ovladače pro specifický hardware National Intruments. Po doinstalování chybějících modulů a ovladačů jsem se pustil do navrhování.

Pomocí grafického programování jsem navrhl jednotlivé komunikace. Implementace v LabVIEW FPGA mi přišla intuitivní a všechny potřebné prvky jsem našel ve funkcích, což jistě bylo efektivnější než se seznamovat s jazykem VHDL, do kterého se výsledné schéma přeloží.

Na závěr jsem jednotlivé komunikace testoval v laboratoři. Při testování jsem objevil několik nedostatků v prvotních návrzích, které jsem následně odstranil. Nutnost kompilace návrhu při každé jeho změně tento proces protáhla. Průměrná doba kompilace činila 11 minut. Poté jsem zaznamenal průběhy signálů na jednotlivých vodičích.

Literatura

[1] A. Zmrzlý, "Vizuální programování." https://is.muni.cz/el/1433/podzim2013/PV226/um/Vizualni_programovani.pdf, 11 2013. (Zobrazeno 19.5.2018).

[2] E. Glinert, *Visual programming environments: applications and issues*. IEEE Computer Society Press tutorial, IEEE Computer Society Press, 1990.

[3] "O scratchi." https://scratch.mit.edu/about. (Zobrazeno 19.5.2018).

[4] J. Kohoutek, "Vizuální programování." https://is.muni.cz/th/xe6oc/scan.pdf, 4 1999. Masarykova univerzita, Fakulta informatiky, Diplomová práce (Zobrazeno 19.5.2018).

[5] E. Glinert, *Visual Programming Environments: Paradigms and Systems*. No. sv. 2 in IEEE Computer Society Press tutorial, IEEE Computer Society Press, 1990.

[6] "What is labview?." http://www.ni.com/cs-cz/shop/labview.html. (Zobrazeno 7.5.2018).

[7] "Graphical programming." http://www.ni.com/getting-started/labview-basics/dataflow. (Zobrazeno 23.5.2018).

[8] "Labview fpga compilation process: From run button to bitfile." http://www.ni.com/white-paper/9381/en/, 12 2016. (Zobrazeno 8.5.2018).

[9] "Labview fpga module compatibility with windows 10." http://www.ni.com/product-documentation/53899/en/, 6 2017. (Zobrazeno 8.5.2018).

[10] "Ni labview fpga compilation options." http://www.ni.com/white-paper/11573/en/, 8 2014.

[11] "Why does it take so long to compile an fpga vi for a compactrio in hybrid mode." https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019N3PSAU, 3 2018. (Zobrazeno 8.5.2018).

[12] "Ni-9144." http://www.ni.com/cs-cz/support/model.ni-9144.html. (Zobrazeno 13.5.2018).

[13] "Specifications ni 9144." http://www.ni.com/pdf/manuals/372498a_02.pdf, 4 2016. (Zobrazeno 10.5.2018).

[14] "12. implementing simple event triggers in labview fpga." http://www.ni.com/tutorial/14532/en/#toc12, 1 2018. (Zobrazeno 10.4.2018).

[15] "Ni 9476 datasheet." http://www.ni.com/pdf/manuals/373964c_02.pdf, 5 2017. (Zobrazeno 2.5.2018).