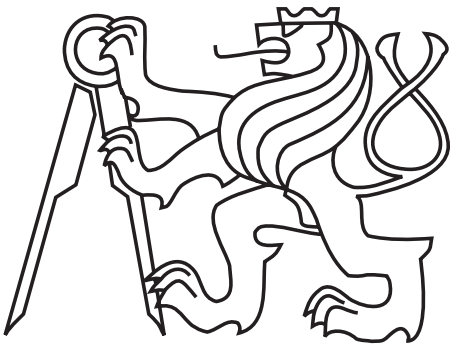
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídící techniky



BAKALÁŘSKÁ PRÁCE

Propojení FPGA s USB portem počítače

Studijní rogram: Kybernetika a robotika
Obor: Systémy a řízení
Vedoucí práce: Ing. Richard Šusta, Ph.D.

Praha, 2015 **Autor: Jiří Kerner**

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jiří Kerner**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Propojení FPGA s USB portem počítače**

Pokyny pro vypracování:

1. Navrhněte propojení vývojové desky Altera DE2 s FPGA obvodem s USB portem počítače.
2. Naprogramujte vhodný modul ve VHDL určený do vývojové desky včetně programu pro Windows k jeho obsluze.
3. Implementujte přenos dat z vývojové desky do počítače s jejich záznamem.
4. Implementujte přenos pokynů z klávesnice a polohy myši z počítače, pro testovací účely, a to i s možností vytvoření opakovatelné dávky povelů, kterou půjde přehrát.

Seznam odborné literatury:

[1] Pong P. Chu: RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability. Wiley-IEEE Press 2006
[2] Pedroni, Volnei A.: Circuit Design and Simulation with VHDL, The MIT Press 2010

Vedoucí: Ing. Richard Šusta, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

V Praze dne 23. 1. 2015

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 _____ _____

podpis

Poděkování

Rád bych poděkoval panu Ing. Richardu Šustovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce. Dále bych rád poděkoval své rodině a přítelkyni za podporu během celého mého studia.

Abstrakt

Bakalářská práce se zabývá USB komunikací mezi vývojovou deskou DE2-115 s FPGA a osobním počítačem s operačním systémem Windows. Práce seznamuje s USB standardem a USB čipem ISP 1362, kterým je osazena deska DE2-115. Dále popisuje USB modul napsaný ve VHDL a uživatelské rozhraní pro Windows určené ke komunikaci s tímto modulem. Součástí práce jsou dále ukázky zapojení modulu do projektu v programu Quartus II a popis ovládání uživatelského rozhraní.

Klíčová slova: USB, USB standard, USB komunikace, DE2-115, ISP 1362

Abstract

This bachelor thesis deals with USB communication between development board DE2-115 with FPGA and personal computer running Windows. This thesis introduces USB standard and USB chip ISP 1362, which is placed on DE2-115. It also describes USB module written in VHDL and user interface running under Windows that communicates with this module. Examples of connecting this module to project in Quartus II and description of user interface are next part of this thesis.

Key words: USB, USB standard, USB communication, DE2-115, ISP 1362

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Bakalářská práce se zabývá komunikací mezi počítačem s operačním systémem Windows a vývojovou deskou DE2-115 [1]. Pro tento účel využívá sběrnici USB.

V teoretické části je uveden popis USB standardu se zaměřením na popis USB zařízení. Je vysvětleno, jak je USB zařízení popsáno svými deskriptory, a také jakým způsobem probíhá připojení zařízení k hostiteli a následná komunikace. Dále je krátce představena vývojová deska DE2-115 a USB čip ISP1362, kterým je deska osazena.

Následující kapitoly jsou zaměřeny na vývoj USB modulu a popisují jeho fungování a způsob propojení s dalšími částmi projektu v programu Quartus II, který se používá pro vývoj programů pro DE2-115. Následuje popis uživatelského rozhraní pro operační systém Windows. V závěru práce je celý systém podroben testování na rychlost a spolehlivost přenosu.

Cílem bakalářské práce je vytvořit funkční USB modul v jazyce VHDL pro DE2-115, který bude snadno použitelný, spolehlivý a jednoduše začlenitelný do projektu v programu Quartus II. Dále pak vytvořit program pro ovládání VHDL modulu s grafickým uživatelským prostředím, který umožní komunikaci mezi vývojovou deskou a počítačem.

1.1 Motivace

Jednou z motivací pro vytvoření USB modulu ve VHDL je jeho použití pro výuku v předmětu Struktury počítačových systému (A0B35SPS), který se vyučuje na Katedře řídící techniky na FEL ČVUT. Pro tento účel je zcela nedostačující řešení od výrobce DE2-115, firmy Altera. Ta řeší USB komunikaci pomocí ovladače naprogramovaného pro procesor Nios II [2]. Toto řešení je náročné na použitou logiku a pro jeho zprovoznění je nutné být podrobně seznámen s fungováním procesoru Nios II a tím, jak se programuje. Navíc toto řešení nelze použít pro projekty psané ve VHDL, které studenti v rámci předmětu zpracovávají.

Během hledání stávajících řešení byl nalezen pouze jeden projekt, který se tvorbou USB modulu ve VHDL pro DE2 zabýval, jedná se o ISP1362 VHDL interface for DE2 v1.0 od Mikhaila Zakhaova [3], přístupné pod GNU GPL licencí [10]. Tento projekt byl značně nedokončený, ale poskytoval dobrý výchozí bod pro další vývoj a tato práce z něj vychází. Bližší popis stavu projektu při převzetí poskytuje kapitola 3.1.

Kapitola 2

Teoretická část

2.1 USB standard

2.1.1 Vznik a historie

USB je technický standard, který vznikl přibližně v polovině devadesátých let minulého století. Důvodem jeho vývoje byla snaha standardizovat připojení pro počítačové periferie, které dokáže nejen zprostředkovat komunikaci, ale zároveň i napájet připojené zařízení. Standard definuje konektory, kabeláž, komunikační protokol a také pravidla pro zmíněné napájení zařízení. V dnešní době se USB standard používá prakticky na všech zařízeních, které se připojují k osobním počítačům. Zároveň však nachází své uplatnění v mnoha dalších odvětvích, především při programování mikrokontrolérů nebo FPGA čipů.

USB standard se neustále vyvíjí a existuje v několika verzích, které jsou spolu zpětně kompatibilní.

- USB 1.1 - Existují dva typy zařízení, pomalá (Low-Speed) s rychlostí 1,5 Mbit/s a rychlá (Full-Speed) s přenosovou rychlostí 12 Mbit/s.
- USB 2.0 - Druhá generace USB přišla s novým režimem Hi-Speed, který nabízí přenosovou rychlost 480 Mbit/s.
- USB 3.0 - Přináší do světa USB změnu v podobě devíti vodičů místo původních čtyř a díky tomu nabízí rychlost až 5 Gbit/s. Konektor je ovšem stále navržen tak, aby zachoval kompatibilitu se staršími verzemi zařízení.

Vzhledem k rozsáhlosti materiálů, které popisují USB standard (640 stran + přílohy) by bylo nad rámec této práce popsat všechny detaily fungování. Pozornost je proto zaměřena na témata, která jsou klíčová pro pochopení dalších částí práce. Ostatní informace lze nalézt buď v kompletní technické dokumentaci [7], nebo ve zkráceném výtahu popisujícím klíčové vlastnosti [6].

2.1.2 Základní technická specifikace

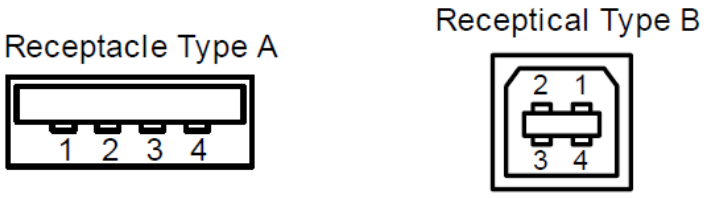
Přístroje zapojené přes USB se dělí na hostitele a zařízení. Hostitel je vždy pouze jeden[1] a řídí pohyb dat na sběrnici, nachází se tedy v roli master. USB sběrnice má hvězdicovou topologii a k jednomu hostiteli lze připojit až 127 zařízení. Tato zařízení jsou v roli slave a nemohou sama vyvolat komunikaci, pouze čekají na požadavky od hostitele.

Do verze USB 2.0 jsou v USB kabelu použity čtyři vodiče. První dva jsou napájecí (+5V a GND) a druhé dva jsou kroucené dvojlinky sloužící jako datové vodiče. Konektory pro připojení k hostiteli a zařízení jsou mechanicky nezaměnitelné. Nejvíce používané typy konektorů se značí jako typ A (používá se na straně hostitele) a typ B (používá se pro připojení zařízení), jak ukazuje obrázek 2.1. Od verze USB 2.0 dále existuje mini-USB B konektor používaný například pro telefony a tablety. Pro zakódování sériového přenosu se používá NRZI (Not Return to Zero Invert) kódování. USB zařízení je typu plug'n'play, uživatel tedy pouze připojí zařízení k hostiteli, ten provede proces enumerace, načte následně příslušný ovladač a zařízení je připraveno k použití. Ovladače k danému zařízení jsou rozlišovány pomocí PID/VID (Product ID/Vendor ID). VID jsou obchodovatelná čísla a je nutno takové číslo zakoupit, jestliže chceme na trh uvést vlastní USB zařízení. Pro nekomerční využití většinou výrobci čipů dávají k dispozici VID/PID o kterém se ví, že není použito v komerčních zařízeních. Jiní výrobci čipů pak prodávají práva na použití PID s jejich VID pro výrobu komerčních zařízení.

Dalším aspektem, ve kterém se USB odlišuje od jiných přenosových standardů, jsou přenosové módy. Řídící mód slouží především pro enumeraci zařízení po připojení, zatímco izochronní mód pro přenos streamů (audio, video). Blokový přenos je určen pro spolehlivý přenos větších bloků dat bez podmínky přenosu v reálném čase, ale na rozdíl od

[1] Neplatí pro USB On the Go, kde se role mohou měnit za běhu

izochronního garantuje doručení všech paketů ve správnem pořadí. Přerušovací přenos je využit v případě, kdy zařízení potřebuje odeslat data zřídka, ale požadují rychlé doručení. Tento typ přenosu garantuje přenos dat v daném čase a v případě chyby opakované odeslání dat.



Obrázek 2.1: Typy USB konektorů [6]

2.1.3 Koncové body a roury

USB zařízení odesílá a přijímá data na definovaných koncových bodech, hostitel přenáší data pomocí rour. Roura je logické spojení mezi hostitelem a koncovým bodem zařízení. Roury lze podle typu přenosů rozdělit na dva druhy.

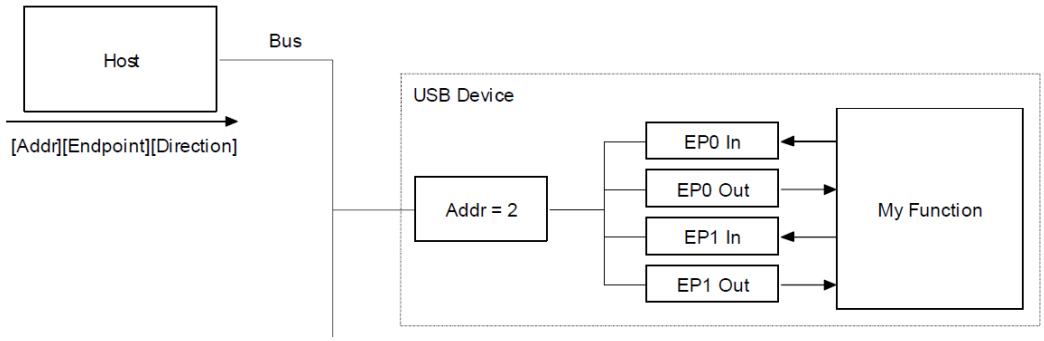
- Stream pipes - Využívají se pro přenosy typu blokové, izochronní a přerušovací. Jsou pouze jednosměrné a přenáší nestrukturovaná data.
- Message pipes - Slouží pro řídící přenosy, jsou obousměrné a typicky posílají strukturovaná data. Jednotlivé přenosy probíhají v sekvenci žádost - data - potvrzení.

Koncové body si lze představit jako ústí roury, ať už vstupní, nebo výstupní. Koncové body jsou označovány IN a OUT podle směru přenosu dat. Pokud hostitel odesílá data na koncový bod s označením EP01, pak se koncový bod označuje EP01 OUT. Zařízení nemůže samo iniciovat přenos, protože USB sběrnice je typu Master-Slave, a proto zapisuje data do vyrovnávací paměti na koncovém bodu EP01 IN a následně čeká, dokud hostitel nezašle IN paket s požadavkem na data. Každý koncový bod je přesně popsán svým deskriptorem koncového budu, o kterém bude řeč dále.

2.1.4 USB adresace

Na obrázku:2.2 vidíme způsob připojení USB zařízení k hostiteli (blok "Host"). V bloku "My Function"se pro každý koncový bod nachází vyrovnávací paměť, která ukládá přijatá

data, nebo data čekající na odeslání. Průběh USB přenosu může vypadat například následovně. Hostitel vyšle paket na sběrnici. USB zařízení podle adresy rozpoznají, pro které z nich je paket určen a následně dané zařízení nakopíruje přijatá data do paměti koncového bodu, kterému byl přenos adresován. Zařízení po úspěšném uložení dat odešle hostiteli ACK signál informující o úspěšném přijetí (pouze izochronní přenos neinformuje hostitele o přijetí, může tedy dojít ke ztrátě dat). Zároveň se v zařízení vytvoří přerušení pro daný koncový bod, které informuje zařízení o přítomnosti dat ve vyrovnávací paměti koncového bodu.



Obrázek 2.2: Zapojení hostitele a zařízení[6]

2.1.5 Řídící přenos

Řídící přenos je většinou použit pro enumeraci zařízení po připojení k hostiteli. Pro Low-speed zařízení je délka paketu dána na 8 bytů, pro High-speed zařízení může být délka paketu 8, 16, 32 nebo 64 bytů (definuje se v koncovém bodu). Při tomto přenosu se používá různě dlouhých bloků dat a protokol se snaží zajistit co největší pravděpodobnost správného doručení pomocí sekvence žádost - data - potvrzení.

2.1.6 Blokový přenos

Blokový přenos se typicky používá pro přenos velkého objemu dat, u kterého záleží na spolehlivém doručení všech paketů ve správném pořadí. Může se jednat například o dokument k vytisknutí odesílaný tiskárně. Blokový přenos je možné použít pouze u Full a High-speed zařízení. Přenos je jednosměrný a není garantovaná přenosová rychlost nebo minimální latence, důležitým parametrem je bezchybný přenos všech dat. K tomu se

využívá CRC kódování, které kontroluje všechny přijaté pakety a při chybě se přenos opakuje.

2.1.7 USB deskriptory

Každé USB zařízení využívá deskriptory k tomu, aby o sobě hostiteli podalo potřebné informace k enumeraci zařízení při připojení. Tyto deskriptory obsahují například informaci o výrobci, podporované verzi USB, počtu koncových bodů a typu konfigurace. USB zařízení popisují následující deskriptory.

- Deskriptor zařízení
- Deskriptor konfigurace
- Deskriptor rozhraní
- Deskriptor koncového bodu
- Textový deskriptor

2.1.7.1 Deskriptor zařízení

USB zařízení může mít pouze jeden deskriptor zařízení. Ten udává informace o výrobci, VID, PID atd. Celkovou podobu deskriptoru zařízení ukazuje tabulka 2.1.

Tabulka 2.1: Deskriptor zařízení

Offset	Pole	Velikost	Hodnota	Popis
0	bLength	1	Number	Velikost deskriptoru v bytech
1	bDescriptorType	1	Constant	Deskriptor zařízení (0x01)
2	vcdUSB	2	BCD	Verze USB specifikace se kterou zařízení pracuje
4	bDeviceClass	1	Class	Kód třídy. Pokud je nulový, každé rozhraní si specifikuje svou vlastní třídu. Pokud je roven 0xFF, pak je kód třídy specifikován výrobcem.
5	bDeviceSubClass	1	SubClass	Subclass code přidělen od USB Org
6	bDeviceProtocol	1	Protocol	Protocol code přidělen od USB Org
7	bMaxPacketSize	1	Number	Maximální velikost paketu pro Endpoint 0
8	idVendor	2	ID	ID výrobce (přiděluje USB Org)
10	idProduct	2	ID	ID produktu (přiděluje výrobce)
12	bcdDevice	2	BCD	Verze zařízení
14	iManufacturer	1	Index	Index textového deskriptoru popisující výrobce
15	iProduct	1	Index	Index textového deskriptoru popisující produkt
16	iSerialNumber	1	Index	Index textového deskriptoru obsahujícího seriové číslo
17	bNumConfiguratons	1	Integer	Počet možných konfigurací

Textové deskriptory slouží k poskytnutí bližších informací o výrobci, produktu a sériovém čísle zařízení. Tyto položky jsou nepovinné, a pokud nejsou uvedeny, měl by pro ně být použit v deskriptoru nulový index.

2.1.7.2 Deskriptor konfigurace

Zařízení může mít více deskriptorů konfigurace. Příkladem může být zařízení, které má vlastní zdroj napájení, ale může být napájeno i přes USB bez připojení externího zdroje. U takovéhoto zařízení se hostitel rozhodne mezi těmito dvěma konfiguracemi podle toho, jak je zrovna zařízení zapojeno. Dalším případem odlišných konfigurací může být případ, kdy se dva konfigurační deskriptory liší v deskriptorech rozhraní a koncových bodů. Většina používaných zařízení je však jednoduchá a mají pouze jeden deskriptor konfigurace. Jeho strukturu ukazuje tabulka 2.2. Jakmile hostitel přijme všechny informace o konfiguraci zařízení, odešle do zařízení příkaz SetConfiguration s nenulovou hodnotou, která se rovná bConfigurationValue jedné z konfigurací. Ve chvíli, kdy hostitel přečte a potvrdí deskriptor konfigurace, vrátí zařízení celou konfigurační hierarchii, tedy všechny deskriptory rozhraní a koncových bodů spadající pod danou konfiguraci. Celková délka odeslaných dat je pak vyjádřena v poli wTotalLength.

Tabulka 2.2: Deskriptor konfigurace

Offset	Pole	Velikost	Hodnota	Popis
0	bLength	1	Number	Velikost deskriptoru v bytech
1	bDescriptorType	1	Constant	Konfigurační deskriptor (0x02)
2	wTotalLength	2	Number	Celková délka odeslaných dat
4	bNumberInterfaces	1	Number	Počet rozhraní
5	bConfigurationValue	1	Number	Hodnota, která se použije jako argument pro vybrání dané konfigurace
6	iConfiguration	1	Index	Index textového deskriptoru popisující tuto konfiguraci
7	bmAttributes	1	Bitmap	D7 Napájení přes sběrnici, D6 Vlastní napájení, D5 Remote wakeup [2], D4-D0 Rezervované (0)
8	bMaxPower	1	mA	Maximální spotřeba energie

2.1.7.3 Deskriptor rozhraní

Deskriptor rozhraní si lze představit jako hlavičkový soubor pro skupinu koncových bodů. Například u tiskárny lze mít jedno rozhraní pro tisk, druhé pro skener atd. Struk-

[2]Zařízení může probudit hostitele, který se nachází v suspend state

turu deskriptoru rozhraní ukazuje tabulka 2.3.

Tabulka 2.3: Deskriptor rozhraní

Offset	Pole	Velikost	Hodnota	Popis
0	bLength	1	Number	Velikost deskriptoru v bytech
1	bDescriptorType	1	Constant	Deskriptor rozhraní (0x04)
2	bInterfaceNumber	1	Number	Index daného rozhraní
3	bAlternateSetting	1	Number	Hodnota pro výběr alternativního nastavení
4	bNumEndpoints	1	Number	Počet koncových bodů daného interface
5	bInterfaceClass	1	Class	Class code (přidělen USB Org)
6	bInterfaceSubClass	1	SubClass	Subclass code (Přidělen USB Org)
7	bInterfaceProtocol	1	Protocol	Protocol code
8	iInterface	1	Index	Index textového deskriptoru popisujícího toto rozhraní

BInterfaceClass, bInterfaceSubClass a bInterfaceProtocol lze použít pro specifikaci typu zařízení (například Mass storage device). Díky tomu lze využít ovladače pro danou třídu zařízení a není nutné vytvářet specifické ovladače.

2.1.7.4 Deskriptor koncového bodu

Deskriptor koncového bodu určuje typ přenosu, směr a maximální velikost paketu, kterou je schopen daný koncový bod přijmout. Kontrolní koncový bod EP0, přes který komunikuje hostitel se zařízením při enumeraci, odpovída na pakety s adresou nula a je ve standardu přesně definován, proto pro něj není potřeba definovat deskriptor koncového bodu. Strukturu deskriptoru popisuje tabulka 2.4.

Tabulka 2.4: Deskriptor koncového bodu

Offset	Pole	Velikost	Hodnota	Popis
0	bLength	1	Number	Velikost deskriptoru v bytech
1	bDescriptorType	1	Constant	Deskriptor koncového bodu (0x05)
2	bIEndpointAdress	1	Endpoint	Adresa koncového bodu
3	bmAttributes	1	Bitmap	Vlastnosti koncového bodu
4	wMaxPacketSize	2	Number	Maximální velikost paketu, který je koncový bod schopen odeslat, nebo přijmout
6	bInterval	1	Number	Interval pro kontrolu koncového bodu hostitelem. Udává se v počtu rámců. Pro blokové a kontrolní přenosy se nevyužívá, pro izochronní je 1 a pro přerušovací může nabývat hodnoty 1 až 255.

Adresa koncového bodu je kódovaná následujícím způsobem. Bity 0 až 3 udávají číslo koncového bodu, bity 4 až 6 jsou rezervované, hodnota se nastavuje na 0. Bit 7 udává směr. Pro OUT koncový bod je hodnota 0 a pro IN koncový bod 1.

Vlastnosti koncového bodu jsou zakódované v bytu takto. Bity 0 až 1 udávají typ přenosu. Kontrolní = 00, izochronní = 01, blokový = 10 a přerušovací = 11. Bity 2 až 7 jsou rezervované pro izochronní koncové body.

2.1.8 Enumerace zařízení

Enumerace je proces, při kterém hostitel zjišťuje, jaké zařízení bylo připojeno na sběrnici a jaké má parametry. Typická enumerace zařízení v prostředí Windows probíhá následovně [8].

1. Stabilizace portu - po dobu alespoň 100ms nesmí nastat na portu změna připojení.
2. Reset portu - hostitel resetuje port a umístí zařízení do základního stavu, to nyní odpovída na adrese 0.
3. Požadavek na deskriptor zařízení - hostitel zašle požadavek na deskriptor zařízení, tento požadavek je použit pouze k určení maximální velikosti kontrolního paketu a po přečtení prvních osmi bytů zaslaných zařízením dochází k druhému resetu USB portu.
4. Nastavení USB adresy - v tomto kroku je USB zařízení přidělena adresa a hostitel čeká 10ms, než se zařízení stabilizuje.
5. Druhý požadavek na deskriptor zařízení - hostitel přečte ze zařízení celý deskriptor zařízení a nastaví pole bLength a bDescriptorType podle získaných hodnot.
6. Požadavek na konfigurační deskriptor - z důvodů kompatibility očekává hostitel délku konfiguračního deskriptoru 255 bytů. Po přečtení celého deskriptoru se správnost přijatých dat ověří tak, že počet přijatých bytů musí být větší nebo roven poli wTotalLength. Při správném přečtení celého deskriptoru dochází k validaci zařízení a Windows začne hledat ovladač pro dané zařízení pomocí služby Windows update.

2.2 Altera DE2-115

Tato kapitola krátce seznamuje s vývojovou a výukovou deskou DE2-115 od firmy Altera [5], která je určena pro výuku struktur počítačových systémů, digitální logiky a FPGA technologie. Jedná se o zařízení s velkým množstvím periferií a širokým spektrem využití. Lze ho použít na drobné školní projekty, i na řešení komplexních problémů a pro složitější návrhy.

Deska je osazena FPGA čipem Cyclone IV EP4CE115F29C7 a disponuje několika druhy pamětí, jmenovitě 128 MB SDRAM, 2MB SRAM a 8MB Flash. Dále je možné paměť rozšířit pomocí SD karty. Hodinový signál je generován 50 MHz oscilátorem, ale lze využít i externí SMA vstup a výstup. Zařízení je vybaveno I/O porty pro zprostředkování komunikace s okolím.

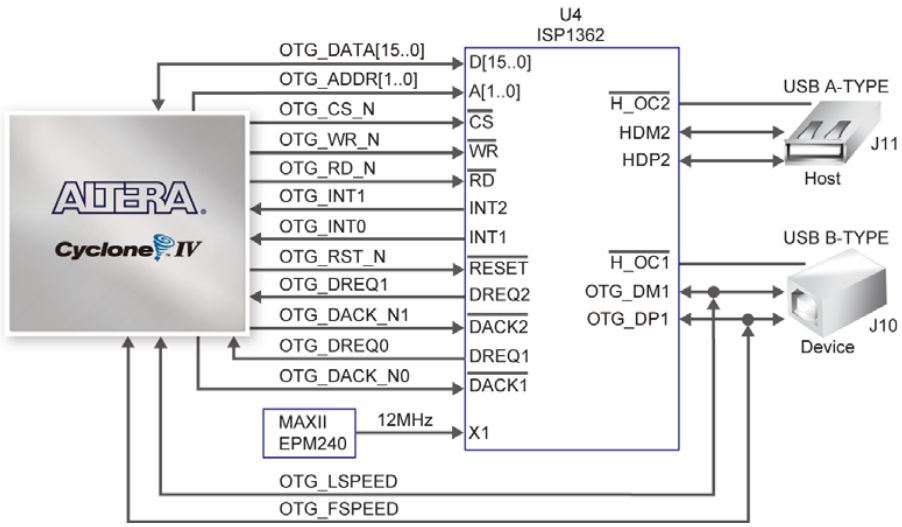
- Zabudovaný USB-Blaster pro konfiguraci FPGA
- Mikrofon (24-bitový Audio CODEC)
- Video Out (VGA 8-bit DAC)
- Video IN (NTSC/PAL/Multi-format)
- RS232
- Vstupní infračervený port
- PS/2 port pro myš nebo klávesnici
- 2x 10/100/1000 Ethernet
- USB 2.0 (Typ A a B)
- HSMC

Zároveň jsou k dispozici periferie pro zobrazování dat a ovládání vývojové desky. Pro zobrazení lze využít osm sedmisegmentových displejů a 16 x 2 LCD displej. Dále pak 18 červených LED a 9 zelených LED. Pro ovládání je připraveno 18 switchů a 4 tlačítka.

2.3 ISP1362

Pro zprostředkování USB komunikace u DE2-115 je na vývojové desce Philips ISP1362 single-chip Universal Serial Bus (USB) On-The-Go (OTG) controller (dále ISP1362) [4]. Tento čip je tvořen z Philips slave host controller (dále HC) a ISP1811B Device controller (dále DC). K ISP1362 jsou připojeny dva USB porty, první z nich je typu A a slouží k připojení USB zařízení, pokud čip slouží jako hostitel. Druhý port je typu B a USB zařízení se přes něj připojuje USB kabelem k hostiteli.

ISP1362 podporuje i OTG specifikaci, která umožňuje střídání rolí (Master-Slave) mezi dvěma zařízeními na základě vyjednávacího protokolu k tomu určeného. ISP1362 je plně kompatibilní s USB specifikací 2.0 a v režimu Full-speed podporuje přenos dat v rychlosti 12 Mbit/s. Na obrázku 2.3 lze vidět jak je FPGA Cyclone připojen k ISP1362.



Obrázek 2.3: Propojeni FPGA a ISP1362 [5]

Tabulka 2.5 popisuje jednotlivé signály použité při propojení a jejich roli při komunikaci mezi FPGA Cyclone a ISP1362.

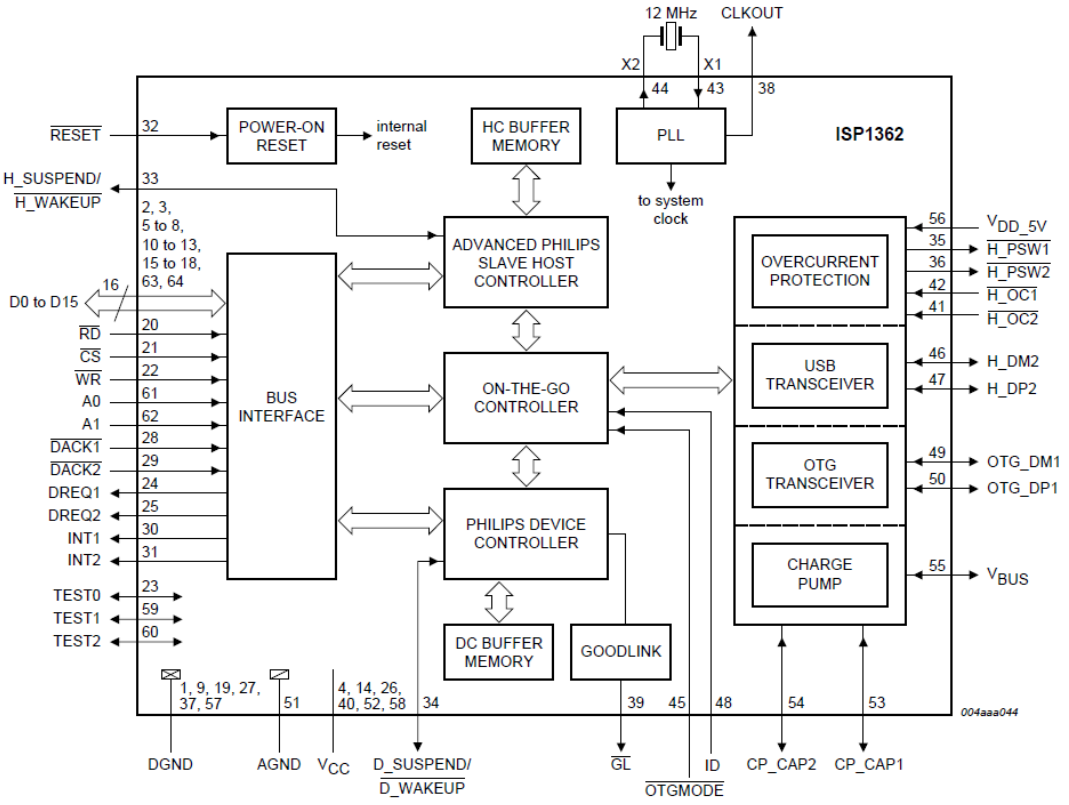
Tabulka 2.5: Propojení FPGA a ISP1362 [5]

Signál	Směr	Použití
OTG_ADDR[0]	output	Rozhoduje, zda se jedná o příkazovou či datovou fázi při zápisu, nebo čtení registru.
OTG_ADDR[1]	output	Vybírá mezi Host controller a Device controller (viz. obr.2.4).
OTG_DATA[15..0]	bidirectional	Obousměrná datová sběrnice, pomocí které se lze připojit k vnitřním registrům a k vyrovnávacím pamětím v ISP1362.
OTG_CS_N	output	Chip select rozhoduje, zda se přistupuje do registru, nebo do vyrovnávací paměti.
OTG_WR_N	output	Pokud je OTG_WR_N aktivní, lze zapisovat je do registru, nebo vyrovnávací paměti
OTG_RD_N	output	Pokud je OTG_RD_N aktivní, lze číst z registru nebo vyrovnávací paměti.
OTG_RST_N	output	Vstupní reset signál restartuje zařízení, to je znovu nakonfigurováno a připojeno k hostiteli.
OTG_INT[1..0]	input	Přerušovací signál od ISP1362, který informuje o změně ve vyrovnávací paměti koncového bodu
OTG_DACK[1..0]	output	Vstup pro DMA - není použit
OTG_DREQ[1..0]	input	Výstup pro DMA - není použit
OTG_FSPEED	bidirectional	USB Full Speed, 0 = Enable, Z = Disable
OTG_LSPEED	bidirectional	USB Low Speed, 0 = Enable, Z = Disable

2.3.1 Device controller

V bakalářské práci funguje DE2-115 v roli USB zařízení, komunikace je tedy řízena obvodem ISP1811B Device controller. DC obsahuje 14 programovatelných koncových bodů a 2 kontrolní IN/OUT koncové body. Přímo na čipu je integrováno 2462 bytů multikonfigurační vyrovnávací paměti a je podporován double buffering pro zvýšení propustnosti a ulehčení přenosů v reálném čase. ISP1362 se připojuje k mikrokontroléru, nebo FPGA

čipu (dále zmiňováno jen jako mikrokontrolér) pomocí vysokorychlostní šestnácti bitové sběrnice s rychlostí přenosu až 10 MB/s. Na obrázku 2.4 se je vidět blokové schéma USB čipu.



Obrázek 2.4: ISP1362 [4]

ISP1362 používá I/O mód pro přístup k vnitřním registrům a vyrovnávací pamětí pomocí externího mikrokontroléru. Využívá k tomu signály popsané v tabulce2.5. ISP1362 nabízí i Direct Memory Acces (DMA) přístup, neboli přímý přístup do paměti, ale tento způsob nebude v práci použit, proto není blíže popsán. Více informací lze nalézt v datasheetu výrobku.

Vyrovnávací paměť pro zařízení je rozdělená do šestnácti bloků. První dva jsou určeny pro kontrolní vstupní a výstupní koncové body a následuje prostor pro dalších čtrnáct programovatelných koncových bodů. Přestože kontrolní koncové body mají fixní velikost danou v datasheetu produktu, je potřeba určit prostor ve vyrovnávací pamětí pro všech šestnáct koncových bodů, než dojde k rozdělení vnitřní vyrovnávací paměti. Programova-

telné koncové body mohou mít velikost od 16 do 1023 bytů. Důležité je ale nepřesáhnout v součtu celkovou velikost vyrovnávací paměti 2462 bytů.

Tabulka 2.6 ukazuje jednotlivé kombinace na vstupu pro přístup do interních registrů zařízení, nebo do vyrovnávací paměti. Zatímco adresa pro přístup je dekódována ze signálu CS, A0 a A1, směr signálu je určen pomocí signálů RD a WR. Pro úplnost je nutno dodat, že označení L značí stav LOW(logická nula), zatímco H je zkratka pro stav HIGH(logická jedna).

Tabulka 2.6: Adresace na I/O portu [4]

CS	A1	A0	Druh přístupu	Šířka datové sběrnice	Popis
L	L	L	R/W	16 bitů	HC data port
L	L	H	W	16 bitů	HC command port
L	H	L	R/W	16 bitů	DC data port
L	H	H	W	16 bitů	DC command port

Přístup k registru nebo do vyrovnávací paměti zařízení probíhá ve dvou fázích. V první fázi se vyšle 16bitový příkaz, který slouží jako adresa registru, se kterým bude komunikováno v datové fázi. Pro určení registru se využívá pouze nižší byte, data ve vyšším bytu jsou ignorována. Při druhé fázi dochází přenosu dat mezi mikrokontrolérem a vnitřním registrem ISP1362. Většina registrů v ISP1362 je 16bitová, ale čip obsahuje i několik 32bitových registrů. V jejich případě pak za příkazovou fází následují dvě fáze datové. Kompletní seznam DC registrů včetně jejich jmen, adres a významů jednotlivých bitů v nich lze nalézt v datasheetu, kap. 16.

O změnách ve vyrovnávací paměti koncových bodů informuje ISP1362 příslušný mikrokontrolér pomocí hardwarového přerušení. V základním nastavení slouží signál OTG_INT[0] k přenosu přerušení od HC a OTG_INT[1] informuje o přerušení od DC. Pro každý koncový bod lze nastavit, zda má posílat přerušení, když dojde ke změně ve vyrovnávací paměti.

Kapitola 3

Řešení projektu

Praktická část bakalářské práce sestává ze dvou hlavních částí. První z nich je vytvořit USB modul napsaný ve VHDL do vývojové desky, který bude snadno použitelný, spolehlivý a nenáročný na použitou logiku. Druhým úkolem je vytvoření programu pro Windows, který bude zajišťovat komunikaci s vývojovou deskou a poskytne uživateli pohodlné rozhraní pro řízení komunikace. Jak již bylo zmíněno v úvodu práce, jako výchozí bod pro splnění těchto úkolů je použit projekt ISP1362 VHDL interface for DE2 v1.0 od Mikhaila Zakhaova. Tento projekt je volně dostupný k použití a úpravě pod GNU GPL licencí.

3.1 Výchozí stav projektu

Zdrojové kódy projektu jsou dostupné z [3]. Projekt byl naposled upraven v září roku 2012 a při převzetí se nacházel v následujícím stavu vývoje.

- VHDL modul lze nahrát do vývojové desky DE2 pomocí softwaru Quartus II. Po jejím připojení dojde k úspěšné enumeraci zařízení. Modul operuje pouze ve smyčce. To znamená, že data odeslána hostitelem do zařízení, jsou zařízením přijata a odeslána zpět hostiteli. Zcela chybí možnost připojit k modulu vstupní signál, nebo z modulu přečíst data. Na jeden USB přenos lze odeslat nebo přijmout dva byty dat pomocí blokového přenosu.

- Je navrhnuto použití knihovny libusb-win32 v programovacím jazyce C pro obsluhu USB portu na straně hostitele, tedy v osobním počítači [9].

- Jako program pro připojení k USB zařízení a následnou komunikaci je použit upravený ukázkový program dodávaný spolu s libusb-win32. Tento program nabízí pouze možnost jednorázového připojení zařízení, odesílání předem daných dat do zařízení a jejich následné přečtení a vypsání do konzole. Poté je zařízení odpojeno a program končí.

3.2 Souhrn řešených problémů

USB modul v původní verzi měl několik nedostatků, které bránili jeho propojení s jinými částmi projektu. Proto na něm byly učiněny následující změny.

- Dávka dat přenášená v jednom USB přenosu byla zvýšena z dvou bytů na čtyři byty. Díky tomu lze přenášet větší množství informací a zároveň lze zakódovat více údajů do jednoho přenosu. Lze tak například přenést v jednom přenosu informaci o poloze kurzoru a stavu tlačítek na myši.

- Ovladač USB modulu byl přepsán tak, aby generoval ACK signály při přijetí přerušení z koncových bodů.

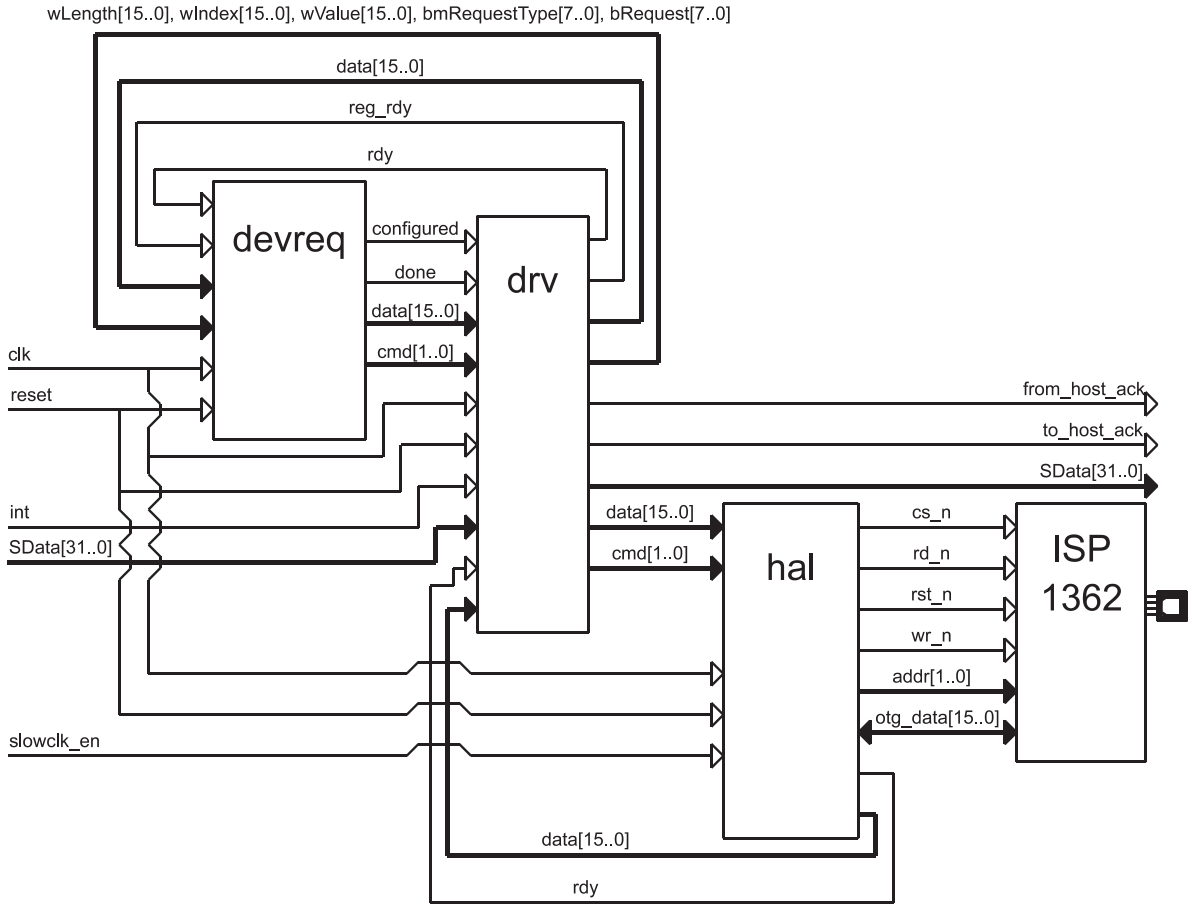
- Pro propojení USB modulu s projektem v Quartus II, bylo vytvořeno rozhraní umožňující rychle vložení modulu do projektu. Byl dopsán proces, který řídí přenos dat ze vstupu USB modulu do vyrovnávací paměti koncového bodu EP02 a zápis dat z koncového bodu EP01 na výstup modulu. O změnách na koncových bodech je nyní uživatel informován pomocí ACK signálů a zápis dat do paměti lze řídit pomocí vstupu DATA_SEND_REQ.

- Pro řízení odesílání dat hostiteli byly navrhnuty a implementovány tři odesílací módy.

- Pro Windows bylo vytvořeno uživatelské rozhraní, které umožňuje připojení k USB zařízení a zároveň funguje jako TCP server. Uživatel pak může využít jednu z aplikací, která byla v rámci této práce vytvořena, nebo si může napsat svou vlastní a pomocí socketů se k rozhraní připojit a komunikovat s vývojovou deskou.

3.3 USB modul

USB modul je popsaný šesti VHDL zdrojovými soubory. Jednotlivé soubory jsou usb_inc.vhd (balíček obsahující USB deskriptory, konstanty a datové struktury), *isp_inc.vhd* (balíček obsahující názvy a adresy registrů v ISP1362). Dále pak drv.vhd, devreq.vhd, hal.vhd a usb.vhd. Na obr. fig:modul je blokové schéma USB modulu. Propojení jednotlivých bloků a propojení s top-level entity je definováno v usb.vhd.

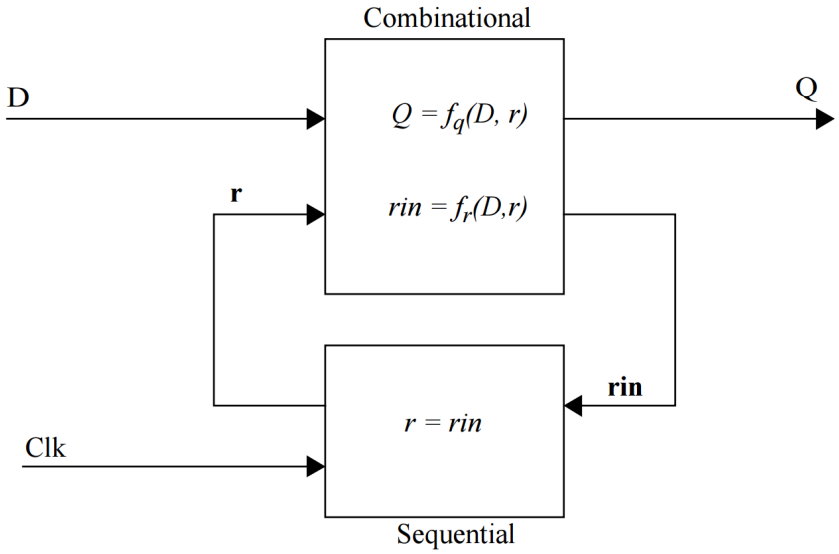


Obrázek 3.1: Blokové schéma USB modulu

3.3.1 Metoda pro strukturovaný VHDL design

Při psaní souboru bylo využito strukturované metody pro psaní VHDL od Jiřího Gaislera [11]. Podstatou této metody je použití dvou procesů pro každou entitu. Graficky je tato myšlenka zobrazena na obrázku 3.2. Vstupy entity jsou označeny D a jsou připojeny k procesu "Combinational". Vstupy pro proces "Sequential"jsou označeny *rin*.

V procesu "Sequential"je vstup zkopírován na výstup při náběžné hraně hodin. Cílem této metody je především zlepšit čitelnost kódu a sjednotit design jednotlivých VHDL entit. První proces, v obrázku 3.2 označený jako "Combinational"obsahuje všechnu asynchronní logiku, zatímco proces "Sequential"obsahuje všechnu sekvenční logiku (registry).



Obrázek 3.2: Generický obvod s dvěma procesy

V případě této práce je v entitách drv.vhd, devreq.vhd a hal.vhd definován výčtový typ reg_t, který obsahuje množinu všech stavů a proměnných. V architektuře dané entity jsou definovány signály r a rin typu reg_t. V procesu "Combinational"je vytvořená promněná v typu reg_t. Proces "Combinational"má typově následující strukturu.

```
comb : process(r, D)
variable v : reg_t;
begin
v := r;
   finite state machine;
   Q <= r;
   rin <= v;
end process;
```

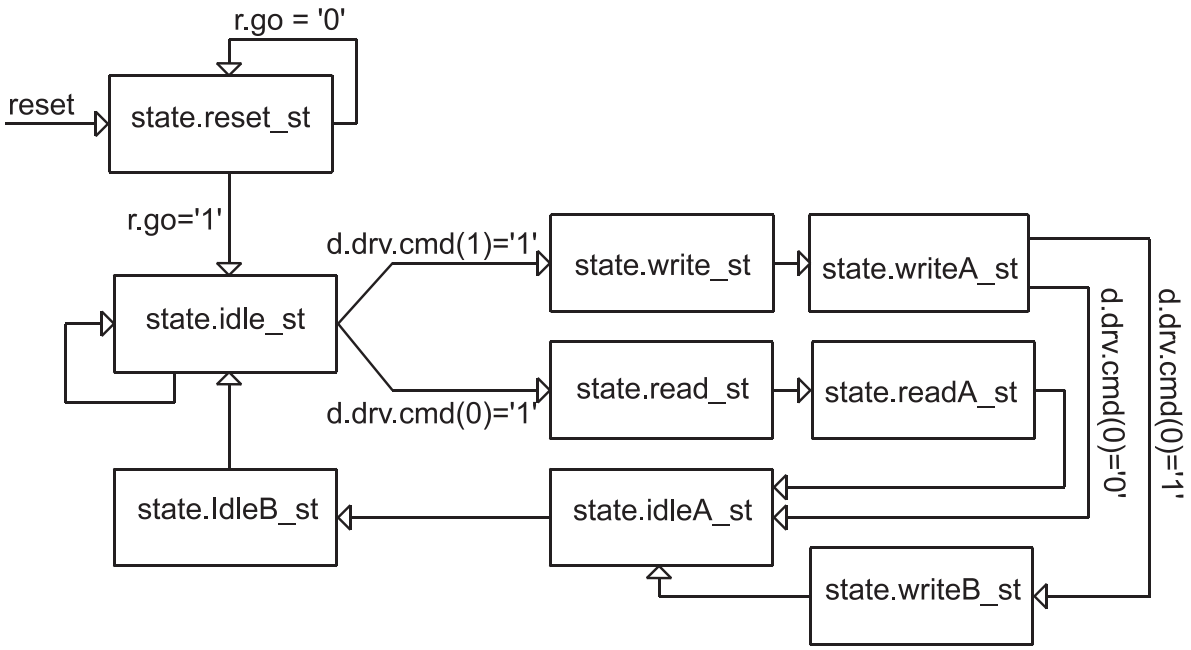
Proces "Sequential"kopíruje vstupy (*rin*) na výstup (*r*) při náběžné hraně hodin. Zároveň při každém průchodu procesem kontroluje, zda uživatel nepožádal o reset obvodu. Pokud ano, USB modul je resetován.

```
seq : process(reset, clk)
begin
if (reset= '1') then
r <= reset;
elsif rising_edge(clk) then
r <= rin
end if;
end process;
```

3.3.2 hal.vhd

HAL, neboli Hardware Abstract Layer se stará o správné časování I/O operací. Konečný stavový automat typu Moore implementuje I/O časování dané datasheetem ISP1362 (kapitola 20.1.2). Ten říká, že jeden cyklus čtení musí trvat minimálně 180 ns a jeden cyklus zápisu musí trvat minimálně 180 ns. V příkazové části zápisu do registru je minimální délka cyklu 205 ns. Hal získává příkazy a data od ovladače USB modulu (drv.vhd) a slouží jako prostředník při komunikaci mezi ovladačem a ISP1362. Konečný stavový automat jak bylo řečeno je typu Moore 3.3, to znamená, že výstup je generován na základě současného stavu automatu a změna na vstupu se na výstupu projeví až v následujícím stavu. Sekvenční proces je řízen hodinami o frekvenci 25MHz, proto každý přechod mezi stavy trvá 40 ns.

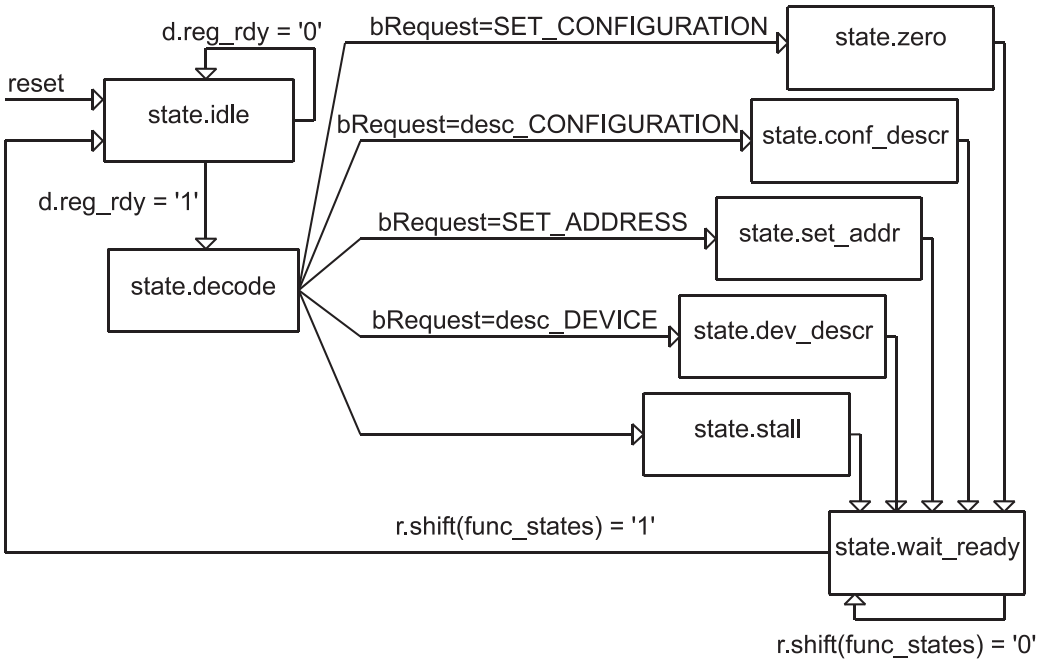
Po nahraní modulu do vývojové desky vyšle HAL signál POR (power on reset), čímž resetuje ISP1362 a přechází do stavu idle_st, ve kterém čeká na příkazy od ovladače. Pokud ovladač odešle příkaz pro čtení nebo zápis, přejde stavový automat do následujícího stavu (write_st,nebo read_st), nastaví příslušné hodnoty řídích signálů a provede požadovanou akci. Následující stavy slouží jako časová výplň pro dodržení požadavků ISP1362 na délku cyklů a přechod mezi stavy probíhá při náběžné hraně hodin.



Obrázek 3.3: Konečný stavový automat entity HAL

3.3.3 devreq.vhd

Entita DEVREQ obsluhuje požadavky na deskriptory odeslané hostitelem na kontrolní koncový bod při enumeraci zařízení a odesílá deskriptory uložené v usb_inc. Komunikace s hostitelem během enumerace je řízena konečným stavovým automatem typu Moore (obrázek 3.4). Když ovladač obdrží požadavek na kontrolním koncovém bodu, signalizuje tuto skutečnost změnou úrovně na portu reg_rdy na logickou jedna. Stavový automat přejde do stavu decode a přeloží aktuální požadavek hostitele. Následně přejde do stavu wait_rdy, ve kterém setrvá, dokud není aktuální požadavek vyřízen. Poté přechází do stavu idle a z něj do stavu decode. Tento cyklus se opakuje, dokud nejsou všechny deskriptory odeslány a hostitel nepotvrdí konfiguraci. Následně entita DEVREQ ohlásí ovladači, že je zařízení nakonfigurováno a předá mu zpět řízení.



Obrázek 3.4: Konečný stavový automat entity DEVREQ

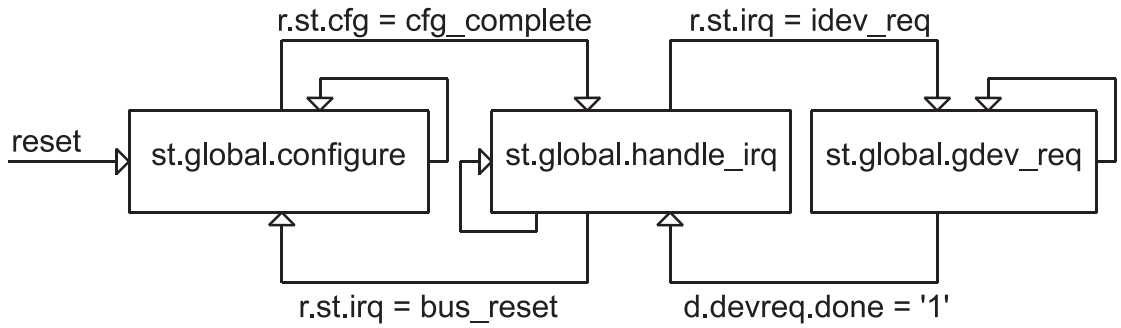
3.3.4 drv.vhd

Entita DRV (v textu dále jako ovladač) řídí komunikaci mezi USB modulem a ISP1362 pomocí pěti stavových automatů typu Moore.

- CFG - nakonfiguruje ISP1362
- GLOBAL - uchovává aktuální stav modulu
- DEV - uchovává aktuální stav zařízení
- IRQ - řídí zápis na koncové body a obsluhuje přerušení od ISP1362
- TXFSM - přijímá data od uživatele modulu a předává je k zápisu do vyrovnávací paměti EP02

USB modul se může nacházet ve třech stavech, konfigurování ISP1362, enumerace zařízení hostitelem a zařízení čekající na požadavky od hostitele. Informace o aktuálním stavu je uchována ve stavovém automatu GLOBAL. Stavový diagram tohoto automatu je na obrázku 3.5. Dále je potřeba znát i aktuální stav USB zařízení. Tato informace je získávána z aktuálního stavu automatu DEV. Ten může nabývat buď stavu in_reset (konfigurace ISP a enumerace), nebo stavu configured (zařízení je nakonfigurováno a připojeno

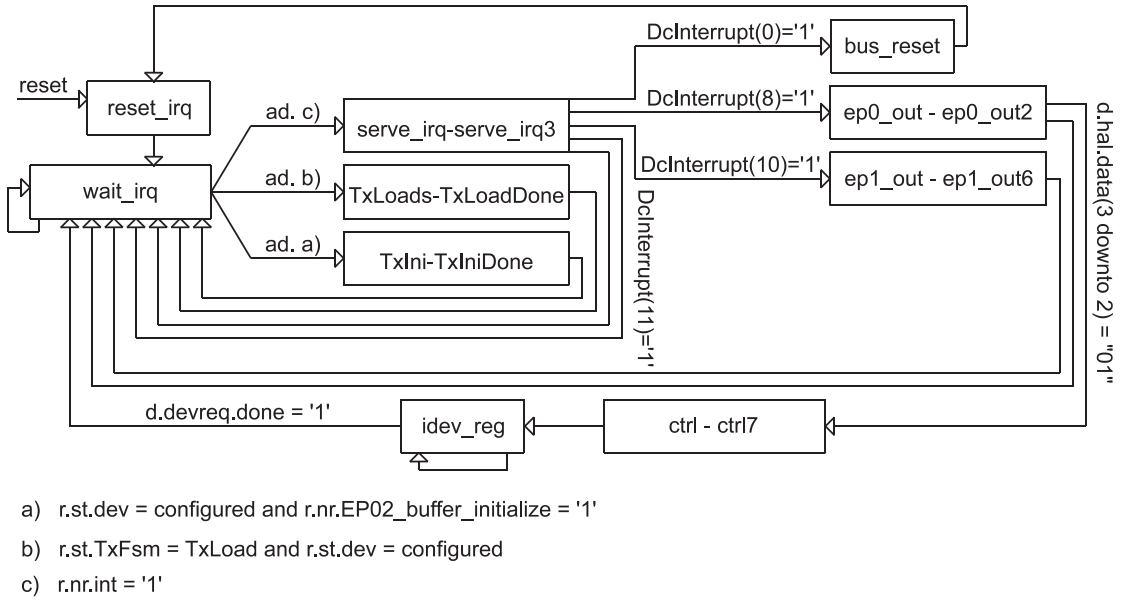
k hostiteli).



Obrázek 3.5: Konečný stavový automat entity DRV, automat GLOBAL

Po nahrání modulu do vývojové desky se zařízení nachází ve stavu configure. V tomto stavu probíhá konfigurace ISP1362, o kterou se stará konečný stavový automat CFG. Ten nejdříve nakonfiguruje USB čip (povolí generování přerušení a softconnect v registru DcMode a zakáže přechod zařízení do suspend state při nečinnosti v registru DcHardwareConfiguration). Následně se konfigurují jednotlivé koncové body pomocí zápisu do jejich DcEndpointConfiguration registru. Pro kontrolní koncové body a koncové body EP01 a EP02 je vyčleněna část vyrovnávací paměti a nastaven jejich směr. Ostatní koncové body je nutné také nakonfigurovat a označit je jako neaktivní. Posledním krokem konfigurace ISP1362 je zápis do DcInterruptEnable registru a povolení generování přerušení na aktivních koncových bodech. Tím je konfigurace ISP1362 kompletní a zařízení přejde do globálního stavu handle_irq, ve kterém čeká na připojení k hostiteli a jeho následné požadavky na deskriptory zařízení při enumeraci.

O zachycení přerušení od ISP1362 a jeho obsluhu se stará konečný stavový automat IRQ. Jeho stavový diagram je na obrázku 3.6. Vzhledem k množství stavů daného automatu jsou některé stavy v diagramu sdruženy do skupin. Mezi stavy v jednotlivých skupinách se přechází ve chvíli, kdy entita HAL signalizuje na vstupním portu rdy dokončení I/O operace s ISP1362. Pokud daný stav neobsahuje I/O operaci, mezi stavy se přechází při náběžné hraně hodin. Po nakonfigurování ISP1362 je stavový automat IRQ ve stavu wait_irq a čeká na přerušení od ISP1362. Po zachycení přerušení načte ovladač hodnotu z DcInterrupt registru (stavy serve_irq - serve_irq2). Ve stavu serve_irq3 zjistí ovladač, který koncový bod přerušení vyvolal (příznakový bit daného koncového bodu je v logické jedna), a následně obslouží tento koncový bod. Před úspěšnou enumerací



Obrázek 3.6: Konečný stavový automat entity DRV, automat IRQ

zařízení lze komunikovat s hostitelem pouze na kontrolních koncových bodech.

Po vyhodnocení, který koncový bod vyslal přerušení, přečte ovladač DcEndpointStatus registr daného koncového bodu, tím vynuluje přerušovací bit v DcInterrupt registru. Následně vyhodnotí byte získaný z DcEndpointStatus registru. Pokud se jednalo o kontrolní kocový bod a ve vyrovnávací paměti koncového bodu je přítomen set-up paket, pak přejde do stavu ctrl, kde čte obsah kontrolního koncového bodu (požadavky na deskriptor zařízení) a následně předá řízení entitě DEVREQ. Konečný stavový automat GLOBAL přechází do stavu gdev_reg a v něm setrvá, dokud není enumerace zařízení dokončena. Úspěšná enumerace je signalizována entitou DEVREQ na portu done logickou jedna. Konečný stavový automat GLOBAL přechází zpět do stavu handle_irq a řízení je předáno automatu IRQ. Ten přejde za stavu wait_irq do stavu TxIni a inicializuje vyrovnávací pamět koncového bodu EP02 zápisem čtyř nulových bytů. V tuto chvíli je USB modul plně nakonfigurován a připraven přijímat a odesílat data.

Nakonfigurovaný a připojený USB modul je řízen automatem IRQ. Ten setrvává ve stavu wait_irq a čeká na přerušení od ISP1362, nebo na pokyn pro zápis dat do vyrovnávací paměti koncového bodu EP02.

3.3.4.1 Čtení dat z EP01

Pokud hostitel zapíše nová data na koncový bod EP01, USB modul je o této skutečnosti informován pomocí přerušení. Automat IRQ přečte obsah DcInterrupt registru a zjistí, že přerušení přišlo z koncovéhé bodu EP01. Ovladač přečte DcEndpointStatus registr daného koncového bodu, tím vynuluje interrupt bit z DcInterrupt registru, poté vyšle požadavek pro přístup k vyrovnávací paměti koncového bodu EP01. Poté z paměti přečte přijatá data a zapíše je na výstup SData. Nakonec vymaže obsah vyrovnávací paměti a odešlě ACK signál Dfrom_host_ack entitě USB.

3.3.4.2 Zápis dat na EP02

Požadavek na zápis dat do EP02 vyjadřuje entita USB pomocí signálu RDy v logické jedna. Konečný stavový automat TXFSM typu Moore ve stavu TxIdle čeká na tento RDy signál. Ve chvíli, kdy ho obdrží, zapíše data ze vstupu do proměnné Din a automat přejde do stavu TxLoad. Tím signalizuje automatu IRQ, že data jsou připravená k zápisu do vyrovnávací paměti EP02. Zároveň se tak zabrání entitě USB požadovat zápis dalších dat, dokud nejsou aktuální data úspěšně zapsána do vyrovnávací paměti.

Automat IRQ přejde do stavu TxLoads, přečte DcEndpointStatus registr, tím vynuluje interrupt bit koncového bodu. Následně zapíše data a validuje obsah vyrovnávací paměti. Validovaná data jsou připravena k odeslání, jakmile přijde požadavek od hostitele. Automat TXFSM přejde do stavu TxIdle a čeká na další požadavek pro zápis od entity USB a automat IRQ přejde do stavu wait_irq.

3.3.5 usb_inc.vhd

Obsahuje deskriptory zařízení, konfigurace, rozhraní a koncového bodu. Vytváří z nich balíček, ze kterého čerpá entita DEVREQ data při enumeraci zařízení.

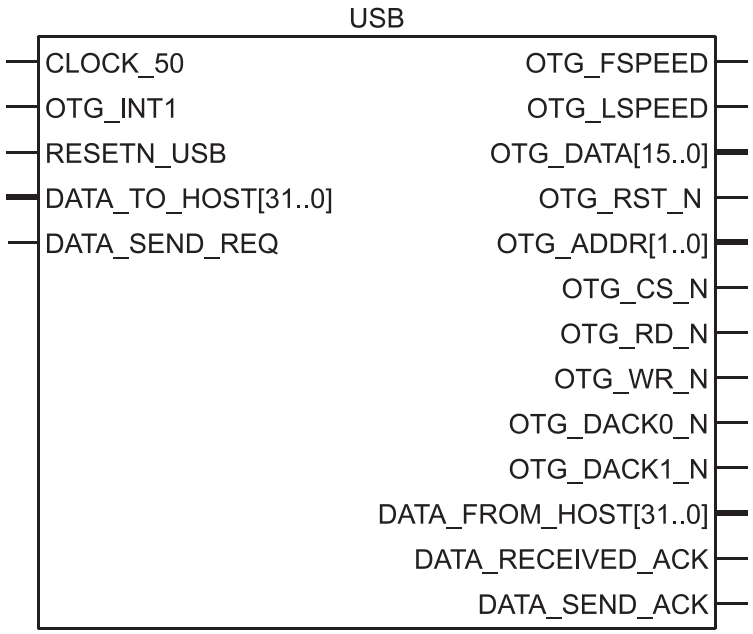
3.3.6 isp_inc.vhd

Obsahuje názvy a adresy registrů a konstant DC interpretovaných pro VHDL. Jednotlivé registry a jejich podrobný popis lze nalézt v datasheetu ISP1362, kapitola 16.

3.3.7 usb.vhd

Slouží k propojení entit modulu, a zároveň poskytuje rozhraní pro komunikaci mezi modulem a projektem, ve kterém je vložený. Pro přidání USB modulu do projektu je připraven symbol usb.bsf (obr. 3.7), který lze snadno vložit do blokového schématu v libovolném Quartus II projektu. Porty OTG se připojí k příslušným pinům podle tabulky pin assignments dané DE2 (pro různé verze desek se může lišit). Tyto piny se musí nacházet v top-level entity daného projektu, aby jim mohli být přiřazeny příslušné assignmenty. Ostatní porty se připojují následovně.

- CLOCK50 - Vstup pro přijení hodinového signálu o frekvenci 50MHz.
- RESET_USB - pokud je na vstup přivedena logická nula, celý USB modul je resetován, následně se automaticky znovu nakonfiguruje a čeká na enumeraci od hostitele. Na tento port je doporučeno připojit jedno z tlačítek, lze ale použít i spínač.
- DATA_TO_HOST[31..0] - vstupní port pro odesílaní dat hostiteli. Čtyři byty dat z portu jsou nahrány do vyrovnávací paměti koncového bodu EP02 a následně ISP1362 čeká na požadavek (IN token) od hostitele pro jejich odeslání.
- DATA_FROM_HOST[31..0] - na výstupním portu lze přečíst data odeslaná hostitelem. Přítomnost nových dat je ohlášena na výstupním portu DATA_RECEIVED_ACK.
- DATA_RECEIVED_ACK - Pokud jsou z vyrovnávací paměti koncového bodu EP01 přečtena nová data, pak je tato skutečnost oznámena změnou úrovně na tomto výstupním portu do logické jedna po dobu jedné periody hodinového signálu.
- DATA_SEND_REQ - Tento port slouží k ovládání zápisu do vyrovnávací paměti koncového bodu EP02.
- DATA_SEND_ACK - Ve chvíli kdy hostitel přečte data z vyrovnávací paměti koncového bodu EP02, je tato událost signalizována změnou úrovně na logickou jedna a v té setrvá, dokud logická úroveň na portu DATA_SEND_REQ není nula.



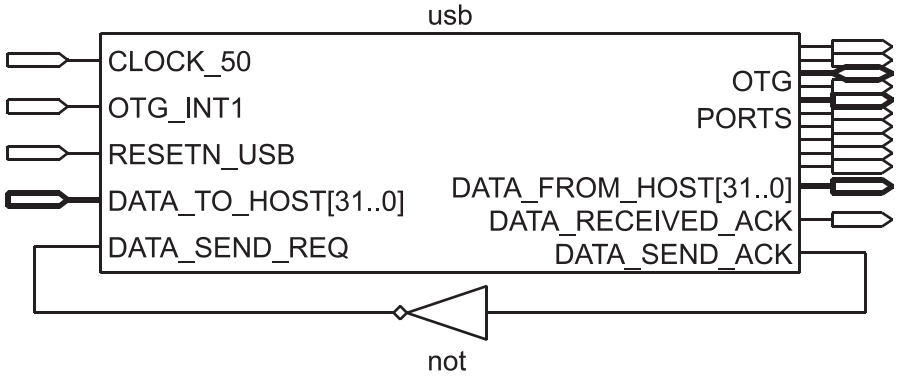
Obrázek 3.7: Symbol pro USB modul

Proces p_send řídí komunikaci mezi vnitřní logikou USB modulu a vstupními signály a přenáší data z USB modulu na výstupní porty. Pokud hostitel odeslal data na koncový bod EP01, proces p_send přečte data z entity DRV a zapíše je na výstupní port DATA_FROM_HOST. Zároveň je vygenerován potvrzovací ACK signál a ten je odeslán na port DATA_RECEIVED_ACK.

Odeslání požadavku na zápis dat z portu DATA_TO_HOST do USB modulu je řízeno pomocí vstupu DATA_SEND_REQ. Pokud je DATA_SEND_REQ v úrovni logické jedna a příznakové bity bufferfull a acknowledge jsou v nule, data ze vstupního portu jsou odeslána do vyrovnávací paměti koncového bodu EP02 a je nastaven příznakový bit bufferfull do logické jedna. Ve chvíli, kdy hostitel přečte data z koncového bodu, USB modul vyšle acknowledge signál a DATA_SEND_ACK je nastaven do úrovně jedna, příznakový bit acknowledge je nastaven do úrovně jedna a bufferfull je vynulován. Při přechodu DATA_SEND_REQ do logické nuly jsou příznakové bity vynulovány a modul je připraven pro další zápis. USB modul lze zapojit třemi různými způsoby a získat tak rozdílné chování.

3.3.7.1 Zapojení se zpětnou vazbou

Výstup DATA_SEND_ACK se přes hradlo NOT připojí ke vstupu DATA_SEND_REQ, jako na obrázku 3.8. Pokaždé, když hostitel přečte data ze zařízení, nahrají se do vyrovnávací paměti koncového bodu nová data ze vstupu DATA_TO_HOST. Příklad takového zapojení v Quartus II se nachází na přiloženém CD v projektu USB_Module_Method_1. V tomto i následujících dvou ukázkových projektech nejsou použity zdrojové soubory jak byly popsány vyše, ale pro větší uživatelské pohodlí jsou sdruženy do jednoho souboru, který lze do Quartus II projektu vložit jako knihovnu. Modul se pak s projektem propojuje v blokovém schématu pomocí bloku usb.bsf.



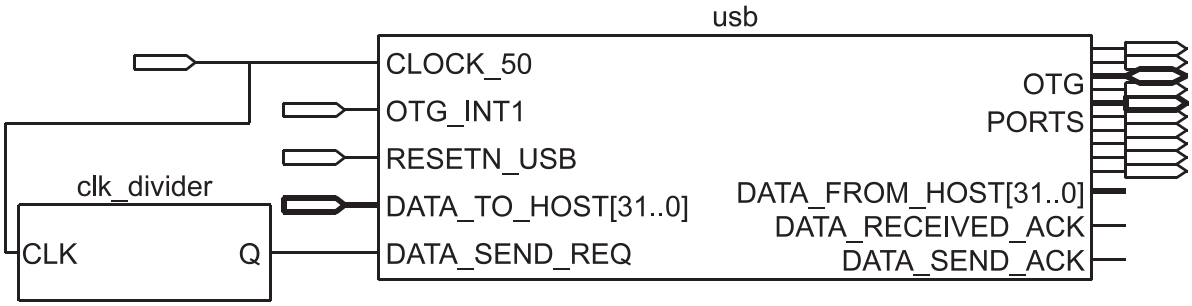
Obrázek 3.8: Zapojení se zpětnou vazbou

Výhodou tohoto zapojení je jeho jednoduchost. Nevýhodou je pak fakt, že nelze reagovat na příchod nových dat na port DATA_TO_HOST a nelze ani posílání nových dat řídit, pokud řídící blok používá jiné hodiny než 50 MHz.

3.3.7.2 Zapojení s periodickým zápisem

Na vstup DATA_SEND_REQ se přivede hodinový signál s maximální frekvencí 0,625 MHz, viz. obrázek 3.9. Tato frekvence nesmí být překročena, protože při vyšší frekvenci uvázne modul v nekonečné smyčce, kdy stále dokola přepisuje data ve vyrovnávací paměti EP02. Při každé změně hodin z logické nula do logické jedna dojde k přepsaní aktuální hodnoty ve vyrovnávací paměti EP02 novou hodnotou ze vstupu DATA_TO_HOST. Příklad takového zapojení se nachází na přiloženém CD v projektu USB_Module_Method_2.

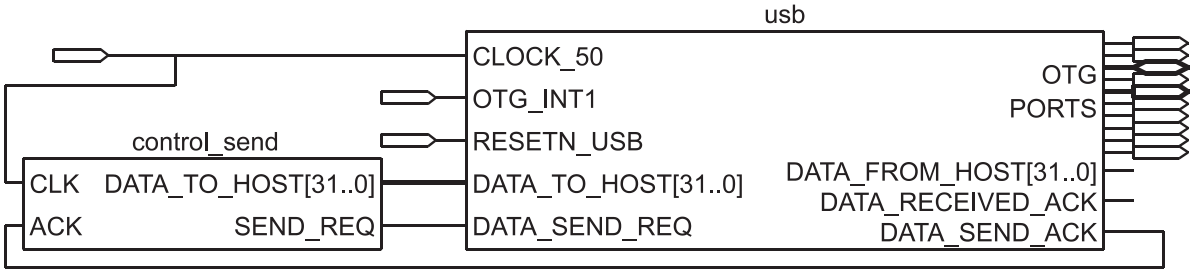
Toto zapojení je ideální pro projekty, kde chceme při každém čtení znát aktuální hodnotu na portu DATA_TO_HOST v čase čtení.



Obrázek 3.9: Zapojení s periodickým zápisem

3.3.7.3 Zapojení s řídícím obvodem

Užival modulu vytvoří zapisovací entitu, která řídí zápis dat na port DATA_TO_HOST. K modulu se připojí jako na obr. 3.10



Obrázek 3.10: Zapojení s řídícím obvodem

Entita control_send obsahuje proces, který podle stavu na vstupu ACK upravuje výstupy. V logické úrovni nula na vstupu ACK se zapíše jedna na DATA_SEND_REQ. Poté se čeká, dokud na vstupním portu ACK není logická jedna od DATA_SEND_ACK (hostitel přečetl data z EP02). Následně se zapíše nula na port DATA_SEND_REQ a entita USB změní hodnotu DATA_SEND_ACK na nula, čímž signalizuje připravenost na další zápis dat. Odesílací rychlost je pak omezena pouze tím, jak rychle dokáže hostitel data číst. Při použití tohoto postupu nedojde k závislosti na dvou hodinách a budou přečtena všechna data ve správném pořadí. Příklad takového zapojení v Quartus II se nachází na přiloženém CD v projektu USB_Module_Method_3.

Pro správné fungování modulu v tomto zapojení musí uživatel zajistit, aby frekvence vstupních hodin entity control_send byla alespoň o řád větší, než předpokládaná frekvence příchozích požadavků od hostitele. Pokud například předpokládáme frekvenci požadavků

od hostitele 1 kHz, frekvence řídících hodin pro entitu control_send by měla být alespoň v řádu desítek kHz. Při nedodržení této podmínky může dojít k poklesu přenosové rychlosti a nesprávnému fungování modulu.

3.4 Uživatelské rozhraní

Uživatelské rozhraní a aplikace jsou naprogramovány v jazyce C/C++. Pro tvorbu grafického uživatelského prostředí byla použita knihovna Qt. Jedná se o multiplatformní knihovnu určenou pro tvorbu grafického uživatelského rozhraní. Pro programování bylo využito vývojové prostředí Qt Creator. Ukázka grafiky vytvořených programů se nachází v příloze C.

Pro komunikaci s USB zařízením je užita knihovna libusb-win32, která poskytuje jednoduché programátorské rozhraní pro přístup k USB zařízení. Jedná se o verzi knihovny libusb-1.0 přepsané pro operační systém Windows. Výhodou této knihovny je přiložený program inf_wizard, který vytvoří ovladač pro vybrané USB zařízení a následně tento ovladač nainstaluje. Díky tomu může uživatel rychle začít používat USB komunikaci s DE2-115 aniž by musel podrobně rozumět USB standardu.

3.4.1 DE2_USB_Interface

DE2_USB_Interface slouží jako komunikační rozhraní mezi DE2-115 a uživatelskou aplikací. Při spuštění programu je spuštěn TCP server, který poslouchá v místní síti (IP 127.0.0.1) na portu 4444 (pokud je port zabrán, zkouší další porty, dokud nenajde volný port). Uživatelská aplikace se může k tomuto serveru připojit a odesílat požadavky na zápis či čtení z DE2-115. V daný čas může být k serveru připojen maximálně jeden klient. Toto řešení bylo zvoleno především pro zvýšení variability využití spojení s DE2-115. K DE2_USB_Interface se může připojit program s TCP klientem napsaným v jakémkoliv programovacím jazyce a studenti umí s TCP sockety pracovat už z kurzů programování. To jim dává možnost vytvořit si vlastní program pro jejich konkrétní projekt, pokud by jim nestačili možnosti programů vytvořených v rámci této práce. Pro případné přímo spojení jsou funkce používané pro připojení a komunikaci s USB zařízením ve zdrojovém

souboru usbhandler.cpp.

Při startu programu je odeslán požadavek na připojení k DE2-115. Pokud USB zařízení není připojeno, program odpovídá na požadavky od TCP klienta zprávou "USBdevice not connected". Program při svém běhu odchytává zprávy systému Windows o změnách v připojených zařízeních. Pokud je zachycena zpráva týkající se připojení, či odpojení USB zařízení (program rozezná DE2-115 podle VID a PID), je provedena příslušná akce. Připojování a odpojování zařízení je tedy automatizované a uživatel programu nemusí nic složitě nastavovat.

Pro jednoduché aplikace, nebo pro možnost rychlého ověření funkčnosti USB modulu jsou v programu k dispozici tlačítka Read a Write. Tlačítko Write odešle data z textového pole (lze zadat pouze maximálně 32 bitů v hexadecimálním tvaru). Tlačítko read přečte data z USB modulu a zapíše je do konzole programu.

TCP server reaguje na následující požadavky od klientů. Pokud klient zašle požadavek začínající na 'w' (0x77) následovaný až čtyřmi byty dat, tato data jsou následně zapsána do DE2-115 a klientovi je odeslána zpráva potvrzovací zpráva "write". Pokud je v požadavku odesláno méně dat než čtyři byty, server doplní odesílaná data nulovými byty. Pokud je odesláno více bytů než čtyři, jsou nadbytečná data serverem ignorována. Pro čtení dat z DE2-115 odesílá klient požadavek 'r' (0x72). Server přečte čtyři byty dat z DE2-115 a odešle je klientovi. Při špatném formátu požadavku vrací server návod na zadání požadavku ve správném tvaru.

3.4.2 DE2_USB_Interface_Table

Uživatelská aplikace s tabulkovým editorem určená k použití s DE2_USB_Interface (dále server). Umožňuje vytvářet, otevírat a ukládat textové soubory obsahující 32 bitová čisla v hexadecimálním tvaru. Do tabulky lze také zadávat z klávesnice 32 bitová čísla v hexadecimálním tvaru. Pomocí TCP klienta se lze připojit k serveru a následně odesílat hodnoty z tabulky do DE2-115 pomocí tlačítka Write a to i opakovaně. Stisknutím tlačítka Read je odeslán požadavek na server pro přečtení hodnoty z DE2-115, hodnota vrácená serverem je zapsána do vybraného řádku tabulky.

Čtení a zápis lze provádět i periodicky a periodu lze nastavit v rozmezí 1-1000 ms.

Pro periodické čtení či zápis je nutno zaškrtnout CheckBox v pravém dolním rohu programu. Kliknutím na požadovanou akci je spuštěn časovač, který vždy po uplynutí dané periody provede požadovanou akci. Použitý časovač garantuje minimální dobu periody, tedy při nastavení periody 25 ms je tento údaj minimální doba mezi provedením dvou akcí, skutečná doba mezi akcemi však může být delší.

3.4.3 DE2_USB_Interface_Mouse

Uživatelská aplikace pro sledování polohy kurzoru myši. Zmáčknutím tlačítka Start se začne s periodou 40 Hz zjišťovat aktuální poloha kurzoru myši. Poloha je měřena v rámci textového pole ve středu programu. Nezávisle na velikosti okna programu je použit přepočet na rozlišení daného monitoru a data jsou kódována podle klíče, který je uveden v programu. Pokud se kurzor nachází mimo textové pole, je odeslána poslední známá poloha kurzoru v rámci textového pole.

3.5 Návod na zprovoznění USB modulu a uživatelského rozhraní

Návod na zprovoznění USB modulu a ovládacích programů na počítači s operačním systémem Windows. Návod se odkazuje na soubory na přiloženém CD.

1. Nejprve je potřeba vytvořit projekt v programu Quartus II a do jeho top-level entity vložit USB modul a připojit piny na příslušné porty. Návod na připojení jednotlivých portů je v 3.3.7. Lze též použít jeden z ukázkových projektů připravený na přiloženém CD. Hotový projekt zkompilujte a nahrajte do desky.

2. Ve chvíli, kdy je program s USB modulem nahrán v DE2-115, propojte USB kabelem port osobního počítače s USB portem s označením DEVICE na DE2-115. Lze použít kabel pomocí kterého jste desku programovali, nebo i druhý nezávislý kabel. Operační systém Windows by měl zaznamenat připojení nového zařízení a začít hledat ovladač pro dané zařízení.

3. Ve složce libusb-win32-bin-1.2.6.0 ve složce bin zpusťte program inf-wizard, ze seznamu vyberte neznámé zařízení s Vendor ID 0x09FB a Product ID 0x600A, vytvořte ovladač a nainstalujte ho na svůj počítač. Po úspěšné instalaci lze daný ovladač nalézt ve Správci zařízení pod libusb-win32 devices.

4. Nyní spusťte program DE2Interface.exe. Po spuštění programu se USB zařízení automaticky připojí a lze začít komunikovat s DE2-115. Pokud při spuštění programu Windows nahlásí chybějící knihovnu MSVCx110.dll, je nutno doinstalovat Visual C++ Redistributable Packages for Visual Studio 2013.

Kapitola 4

Testování

V této kapitole experimentálně ověřím správnou funkčnost modulu a zjistím maximální rychlosti pro zápis a čtení jak experimentálně, tak výpočtem. Pro experimentální měření jsem USB modul zapojil podle 3.3.7.3 a jako data se odesílají hodnoty šestnácti bitového čítače, který se inkrementuje po každém zápisu dat do USB modulu. Data jsou posílána v rámci dvou nižších bytů, dva vyšší byty jsou nulové. Testovací schéma dále obsahuje šestnácti bitový čítač, který se inkrementuje při každém příjetí potvrzovacího pulsu DATA_RECEIVED_ACK a zobrazuje svou hodnotu na sedmisegmentové displeje (HEX7 - HEX4). Přijatá data (dva dolní byty) z portu DATA_FROM_HOST jsou zobrazována na zbylých čtyřech sedmisegmentových displejích (HEX3 - HEX0).

Pro testování jsem použil notebook ASUS N55S s operačním systémem Windows 7 Home Premium, procesorem Intel(R) Core(TM) i7 - 2670QM CPU @ 2.20 GHz a pamětí RAM 6GB. Dále bylo využito uživatelské rozhraní DE2_USB_Interface vytvořené v rámci této práce. Testovací program se přes TCP socket připojí k rozhraní a spustí for cyklus, který provede 50 000 akcí (zápis, nebo čtení). Čas se měří od okamžiku vstupu do for cyklu až po jeho dokončení.

1. Pro měření rychlosti čtení z DE2-115 je v každém průchodu for cyklem odeslán požadavek na přečtení hodnoty a ta je následně uložena do pole integerů.

2. Pro měření rychlosti zápisu je v každém průchodu for cyklem zapsán do DE2-115 integer z předem inicializovaného pole.

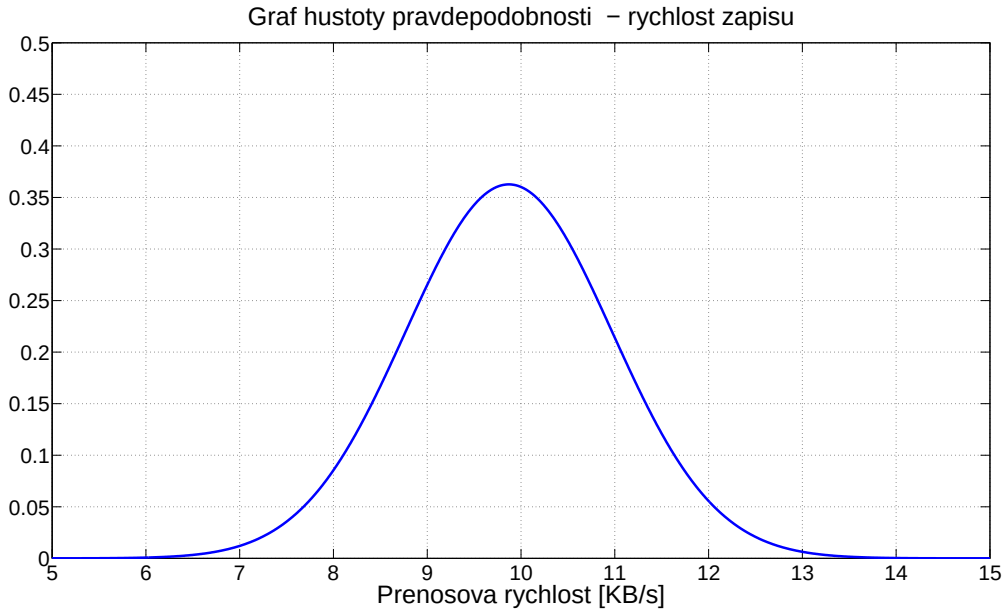
3. Pro ověření správnosti přenosu je v každém průchodu for cyklem přečtena hodnota z DE2-115 a následně je tato hodnota zapsána zpět. Při správném fungování

je očekáváno, že počet průchodů for cyklem bude roven počtu zápisů načítaných čítačem a poslední přečtená hodnota se bude rovnat poslední zapsané hodnotě.

Pro každý z testů bylo provedeno deset nezávislých měření. Pro každé měření bylo odesláno nebo přijato 200 KB. Výsledné časy byly přepočítány na přenosové rychlosti. Přenosová rychlost je spojitá náhodná veličina u které předpokládám normální rozdělení. Z naměřených dat byla vypočtena střední hodnota a výběrová směrodatná odchylka a z nich byl vytvořen graf hustoty pravděpodobnosti normálního rozdělení. Přenosová rychlost je při každém spuštění testovacího programu různá. Její hodnotu ovlivňuje mnoho faktorů, jako například zatížení CPU, obsazení paměti a výpočetní čas přidělený operačním systémem. Toto měření slouží pouze pro vytvoření představy, jaké maximální rychlosti můžeme při používání USB modulu očekávat.

4.1 Rychlost zápisu do USB modulu

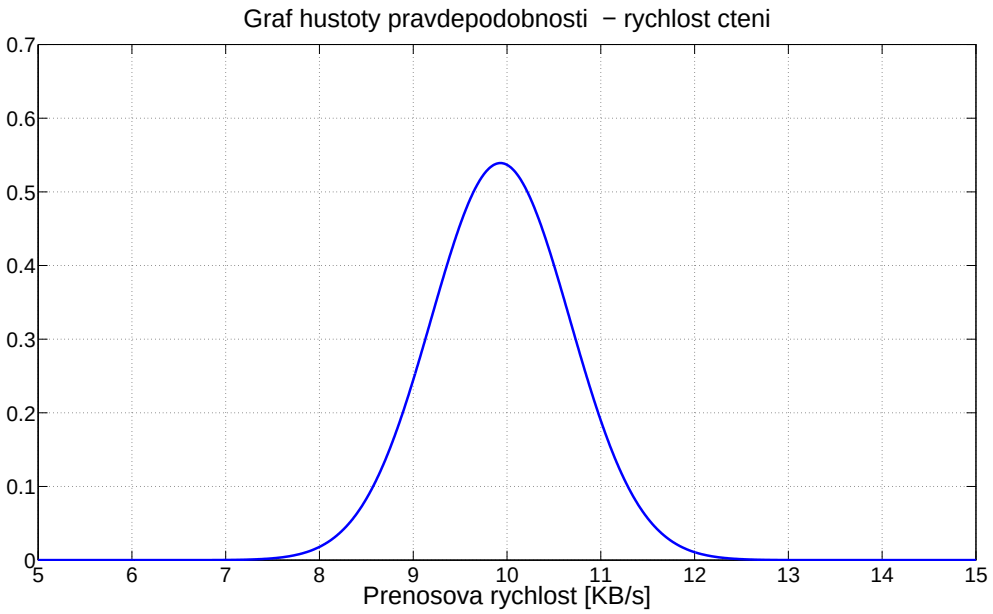
Střední hodnota rychlosti zápisu je 9,87 KB/s. Výběrová směrodatná odchylka je rovna 1,10 KB/s. Výsledný graf hustoty pravděpodobnosti přenosové rychlosti je na obrázku 4.1.



Obrázek 4.1: Graf hustoty pravděpodobnosti - rychlost zápisu

4.2 Rychlost čtení z USB modulu

Střední hodnota rychlosti zápisu je 9,93 KB/s. Výběrová směrodatná odchylka je rovna 0,74 KB/s. Výsledný graf hustoty pravděpodobnosti přenosové rychlosti je na obrázku 4.2.



Obrázek 4.2: Graf hustoty pravděpodobnosti - rychlost čtení

4.3 Výpočet maximální rychlosti

Maximální rychlost pro čtení a zápis lze teoreticky vypočítat, protože známe dobu, po kterou ovladač setrvává v daném stavu příslušného stavového automatu IRQ. Sečtou-li se časy pro jednotlivé stavy stavového automatu, výsledkem je čas, který ovladač potřebuje, aby provedl kompletní zápis nebo čtení na koncovém bodu.

Pro čtení z paměti koncového bodu EP01 je celkový čas I/O operací roven 2200 ns. Pro tento čas by maximální frekvence přenosu byla přibližně 0,45 MHz, to odpovídá přenosové rychlosti 13,7 Mbit/s. Maximální rychlost pro USB 2.0 ve full-speed režimu je ovšem pouze 12 Mbit/s.

Pro zápis do paměti koncového bodu EP02 je celkový čas I/O operací roven 1560 ns. Teoreticka maximální frekvence je tedy ještě vyšší než v předchozím případě. Stejně jako v případě čtení teoretická rychlost zápisu přesahuje možnosti použitého USB čipu. USB modul je tedy teoreticky schopen čtení a zápisu maximální možnou rychlostí 12 Mbit/s.

4.4 Spolehlivost přenosu dat

Testování na spolehlivost přenosu bylo provedeno v desítce nezávislých testů a při žádném z testů nebyla zaznamenána ztráta dat, neprovedení akce, nebo přijetí chybných dat. Zároveň byla sledována správnost přenesených dat i při testech rychlosti zápisu a čtení a ani při nich nebyl pozorován chybný přenos.

Kapitola 5

Závěr

V rámci bakalářské práce byl vytvořen USB modul napsaný v jazyce VHDL, který umožňuje komunikovat s osobním počítačem běžícím pod operačním systémem Windows. Na jeden přenos lze odeslat čtyři byty dat v blokovém přenosu. Je implementováno rozhraní pro řízení komunikace mezi modulem a zbytkem projektu. Modul lze do projektu zapojit ve třech řídících módech podle plánovaného použití. Při zapojení s řídícím obvodem lze kontrolovat přenos dat a zajistit jejich spolehlivé doručení při průměrné rychlosti přenosu okolo 10 kB/s. Tato rychlost je přímo závislá na frekvenci, s jakou hostitel vysílá požadavky.

Pro jednoduché ovládání přenosu v osobním počítači byl napsán program s grafickým uživatelským prostředím, který je připojen k DE2-115 a zároveň funguje jako TCP server v lokální síti. Díky tomu si každý uživatel může napsat vlastní program a s pomocí jednoduché instrukční sady komunikovat se serverem, respektive s DE2-115. Pro základní úlohy a práci byly vytvořeny dva ukázkové uživatelské programy. Jedním z nich je program umožňující odesílat periodicky údaj o poloze kurzoru myši. Druhý je jednoduchý tabulkový editor, který dovoluje práci s daty v tabulce, práci se soubory, odesílání či čtení dat.

Možné další rozšíření této práce vidím v implementování dalších koncových bodů a jiných typů USB přenosů. Zajímavý by byl i projekt USB modulu v roli hostitele a propojení dvou vývojových desek přes USB.

Literatura

[1] Altera: DE2-115 Development and Educational Board. [online], 2015. [cit. 2015-5-13]. Dostupné z: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=502

[2] Altera: Procesor NIOS. [online], 2015. [cit. 2015-5-13]. Dostupné z: http://www.altera.com/devices/processor/nios2/ni2-index.html

[3] ISP1362 VHDL interface for DE2 v1.0. [online], 2012. Dostupné z: https://github.com/mzakharo/usb-de2-fpga

[4] Philips: *ISP1362 datasheet.* [online], 2004. [cit. 2015-5-13]. Dostupné z: http://www.cs.columbia.edu/ sedwards/classes/2008/4840/Philips-ISP1362-USB-controller.pdf

[5] Altera: *DE2-115 User manual.* [online], 2010. [cit. 2015-5-13]. Dostupné z: ftp://ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE2-115/DE2_115_User_Manual.pdf

[6] Craig Peacock: *USB in nutshell.* [online], 2002. [cit. 2015-5-13]. Dostupné z:http://www.ru.lv/ peter/macibas/datoru_arhitektura/usb.pdf

[7] USB - oficiální webové stránky. [online], 2015. Dostupné z:http://www.usb.org/developers/docs/

[8] USB Blog: *USB enumerace ve Windows.* [online], 2009. [cit. 2015-5-13]. Dostupné z:http://blogs.msdn.com/b/usbcoreblog/archive/2009/10/31/how-does-usb-stack-enumerate-a-device.aspx

[9] Libusb-win32. [online], 2014. Dostupné z:http://sourceforge.net/p/libusb-win32/wiki/Home/

[10] GNU GPL. [online], 2015. [cit. 2015-5-13]. Dostupné z:https://www.gnu.org/copyleft/gpl.html

[11] Jiri Gaisler: *A structured VHDL design method.* [online], 2015. [cit. 2015-5-13]. Dostupné z:http://www.gaisler.com/doc/vhdl2proc.pdf

Příloha A

Seznam použitých zkratek

USB - Universal Serial Bus
ISP1362 - Philips ISP1362 single-chip Universal Serial Bus (USB) On-The-Go (OTG) controller
DE2-115 - Altera DE2-115 Development and Education Board
EP01 - koncový bod 01
EP02 - koncový bod 02
OUT koncový bod - směr toku dat je od hostitele k zařízení
IN koncový bod - směr toku dat je od zařízení k hostiteli
VHDL - Very High Speed Integrated Circuit Hardware Description Language
GNU GPL - GNU General Public License
FPGA - Field Programmable Gate Array
VID - Vendor ID
PID - Product ID
OTG - On The Go
HC - Host Controller
DC - Device Controller
DMA - Direct Memory Access
ACK - Acknowledge
TCP - Transmission Control Protocol
I/O - Input/Output
POR - Power On Reset

Příloha B

Obsah přiloženého CD

- /
  - GUI Executables ..... Uživatelské programy
  - GUI Source Codes ..... Zdrojové kódy uživatelských programů
    - DE2_USB_Interface
    - DE2_USB_Interface_Mouse
    - DE2_USB_Interface_Table
  - libusb-win32-bin-1.2.6.0 ..... USB knihovna
  - Text
    - Bakalářská_práce_Jiří_Kerner_2015.pdf
  - USB_Module_Method_1 ..... Quartus II projekt - zapojení se zpětnou vazbou
  - USB_Module_Method_2 ..... Quartus II projekt - zapojení s periodickým zápisem
  - USB_Module_Method_3 ..... Quartus II projekt - zapojení s řídícím obvodem
  - VHDL Module Library ..... Knihovna modulu pro vložení do Quartus II projektu
  - VHDL Module Source Codes ..... Zdrojové kódy USB modulu

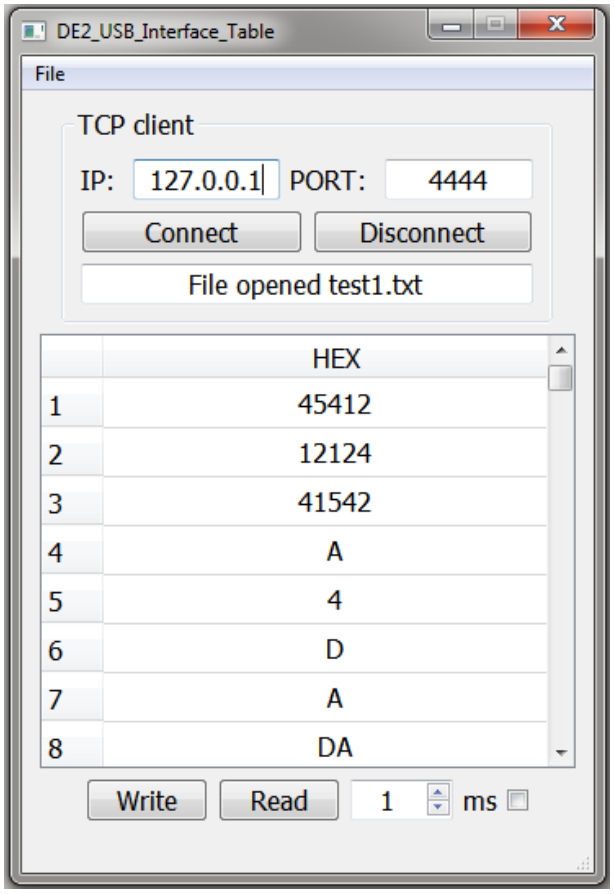
Příloha C

Grafické uživatelské prostředí

C.1 DE2_USB_Interface

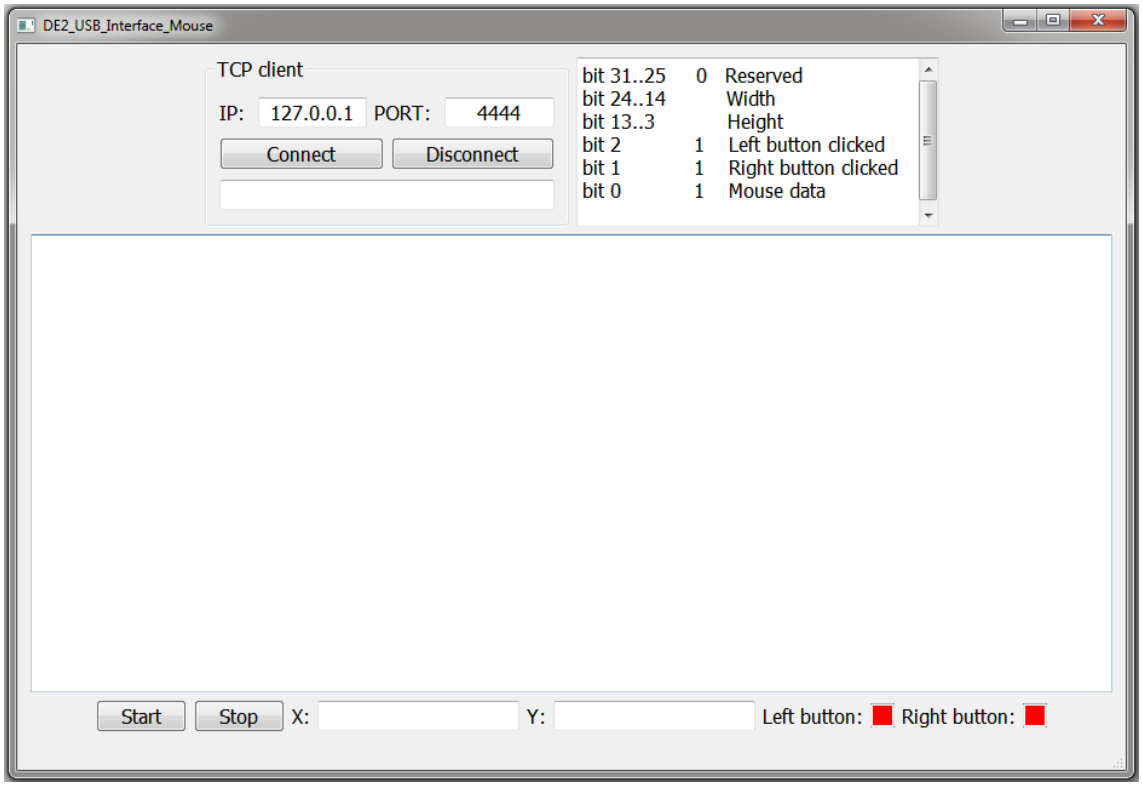
Obrázek C.1: DE2_USB_Interface

C.2 DE2_USB_Interface_Table



Obrázek C.2: DE2_USB_Interface_Table

C.3 DE2_USB_Interface_Mouse



Obrázek C.3: DE2_USB_Interface_Mouse