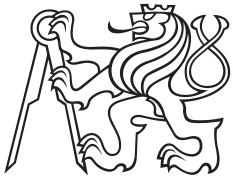
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra mikroelektroniky

Komunikace s MEMS mikrofony pomocí FPGA

Bc. Jan Šedivý

Vedoucí práce: Ing. Petr Honzík, Ph.D.
Studijní program: Elektronika a komunikace
Zaměření: Elektronika
Leden 2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Šedivý** Jméno: **Jan** Osobní číslo: **483884**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Elektronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Komunikace s MEMS mikrofony pomocí FPGA

Název diplomové práce anglicky:

Communication with MEMS Microphones Using FPGA

Pokyny pro vypracování:

Seznamte se s principy funkce MEMS mikrofonů a s formáty jejich digitálních výstupů. Vyhodnoťte vhodnost daných formátů pro použití v mikrofonních polích.
Na základě získaných poznatků implementujte na FPGA příjem digitálních signálů z těchto mikrofonů pomocí formátů I2S a TDM. Navrhněte a implementujte přenos digitálních signálů z FPGA do hostitelského počítače přes USB k dalšímu zpracování.
Vámi implementované řešení otestujte na reálných signálech a vyhodnoťte jeho funkčnost. Navrhněte doporučení pro další postup řešení problému.

Seznam doporučené literatury:

[1] I2S bus specification [online]. Philips Semiconductors, 1986 [cit. 2023-01-19]. Accessible from: https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf
[2] Time Division Multiplexed Audio Interface: A Tutorial [online], Cirrus Logic, 2006 [cit. 2023-02-07], Accessible from: https://gab.wallawalla.edu/~larry.aamodt/engr432/cirrus_logic_TDM_AN301.pdf

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Petr Honzík, Ph.D. katedra radioelektroniky FEL

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

Datum zadání diplomové práce: **14.02.2023** Termín odevzdání diplomové práce: **09.01.2024**

Platnost zadání diplomové práce: **22.09.2024**

Ing. Petr Honzík, Ph.D.	prof. Ing. Pavel Hazdra, CSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Chtěl bych tímto poděkovat svému vedoucímu práce, Ing. Petru Honzíkovi, Ph.D., za vedení a pravidelné konzultace při tvorbě této práce. Dále bych poděkoval svému kolegovi, Ing. Davidu Vagnerovi, za spolupráci a za realizaci mikrofonního pole, k jehož využití tato práce směřovala. V neposlední řadě děkuji své rodině za podporu.

Tato práce byla podpořena grantem číslo SGS23/185/OHK3/3T/13 Českého vysokého učení technického v Praze.

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, 09. ledna 2024

.....

Abstrakt

Tato diplomová práce se zabývá komunikací s MEMS mikrofony pomocí programovatelného hradlového pole (FPGA). V jazyce VHDL bylo na FPGA implementováno čtení zvukových dat z mikrofonů s digitálními rozhraními TDM a I²S. Přečtené vzorky z mikrofonů jsou předávány mikrořadiči FX2LP, který je posílá skrze rozhraní USB do počítače.

V rámci práce byla navržena vlastní deska plošných spojů v programu KiCAD, jež využívá FPGA Xilinx Spartan-3E. Bitovou konfiguraci FPGA na desce lze zapsat do paměti SPI flash pomocí mikrořadiče FX2LP. K tomuto účelu byl vytvořen vlastní skript v jazyce Python.

Navržená deska dovoluje připojit až 16 I²S mikrofonů, ze kterých lze číst data vzorkovacím kmitočtem 48 kHz. V praxi byla deska využita k nahrávání zvuku z kruhového mikrofonního pole s osmi I²S mikrofony, které realizoval Ing. David Vagner, během měření v akustické bezodrazové komoře.

Podařilo se také číst najednou až 64 simulovaných TDM mikrofonů při vzorkovací frekvenci 24 kHz.

Klíčová slova: MEMS mikrofony, FPGA, VHDL, I²S, TDM, mikrofonní pole, FX2LP, USB Audio Class, isochronní přenos, deska plošných spojů

Vedoucí práce:
Ing. Petr Honzík, Ph.D.

Abstract

This diploma thesis focuses on communication with MEMS microphones using a Field Programmable Gate Array (FPGA). Reading of audio data from microphones featuring digital interfaces TDM and I²S was implemented on an FPGA in the VHDL language. The acquired samples from microphones are conveyed to the FX2LP microcontroller, which sends them through the USB interface to a computer.

Within this work, a custom printed circuit board was designed in the KiCAD software, which utilizes the Xilinx Spartan-3E FPGA. The bitstream configuration of the FPGA on the board can be programmed into the SPI flash memory using the FX2LP microcontroller. For this purpose, a custom script in the Python language was created.

The designed board enables connecting up to 16 I²S microphones, from which data can be read with a sampling frequency of 48 kHz. In practice, the board has been utilized for recording sound from a circular microphone array consisting of eight I²S microphones, which was realized by Ing. David Vagner, during measurements in the acoustic anechoic chamber.

Additionally, up to 64 simulated TDM microphones were successfully read simultaneously at a sampling frequency of 24 kHz.

Keywords: MEMS microphones, FPGA, VHDL, I²S, TDM, microphone array, FX2LP, USB Audio Class, isochronous transfer, printed circuit board

Title translation: Communication with MEMS Microphones Using FPGA

Obsah

Obrázky

Tabulky

Úvod

Tato diplomová práce se zabývá návrhem a implementací komunikace s digitálními MEMS mikrofony pomocí programovatelného hradlového pole (FPGA). Svým účelem navazuje na bakalářskou práci Ing. Davida Vagnera [1], který navrhl sférické (kulové) mikrofonní pole z TDM mikrofonů InvenSense ICS-52000. Tyto mikrofony byly čteny pomocí I²S periferie mikropočítače Raspberry Pi. Řešení systému konektorů však bylo náchylné k poruchám, což znesnadňovalo práci s mikrofonním polem.

V jeho navazující diplomové práci spolupracoval se mnou na vytvoření modulárního mikrofonního pole, pro změnu s I²S mikrofony SPH0645LM4H-B výrobce Knowles [2]. Tyto mikrofony mají udávány lepší akustické vlastnosti než původní TDM mikrofony (především frekvenční charakteristiku).

Hlavní myšlenkou bylo číst data z pole mikrofonů několika paralelními I²S rozhraními. Tuto myšlenku jsem se pomocí programovatelného hradlového pole (FPGA) snažil realizovat, neboť díky flexibilitě těchto čipů je možné zpracovat data z velkého počtu mikrofonů souběžně.

Realizovaný logický obvod na hradlovém poli Xilinx pro tento účel byl navržen v jazyce VHDL. S jazykem VHDL a s hradlovým polem Xilinx jsem již pracoval v rámci své bakalářské práce [3]. Oproti bakalářské práci logický návrh neskončil na vývojové desce, ale pro účely této práce byla navržena i vlastní deska plošných spojů. Navržená deska, nazvaná MicArrayBoard, byla osazena hradlovým polem řady Spartan-3E. Pro komunikaci desky s počítačem jsem zvolil rozhraní USB, s čímž mi napomáhá integrovaný obvod Cypress FX2LP.

Nejprve je v teoretické části rozebrána problematika, kterou jsem se zabýval. Zaměřuji se na 3 stěžejní témata, kdy každému je věnována samostatná kapitola: MEMS mikrofony, USB komunikace a konfigurace hradlového pole.

V praktické části následně rozebírám a popisuji řešení, jež jsem realizoval. Není popisován pouze VHDL návrh na hradlovém poli, ale i firmware mikrořadiče FX2LP, navržená deska, programové vybavení pro používání desky a způsob přizpůsobení návrhu pro různá mikrofonní pole. Výsledek práce byl použit při měření kruhového mikrofonního pole v akustické komoře, což si také vyžádalo příspěvek z mé strany. V závěru shrnuji dosažené výsledky práce.

Poznámka ke grafické úpravě

V této práci jsou `strojovým písmem` sázeny:

- součásti zdrojových kódů jazyka VHDL (jako jsou názvy signálů, entit, portů a klíčových slov)
- součásti zdrojových kódů jazyka C,
- názvy souborů a výpis jejich obsahu,
- příkazy příkazové řádky a nebo názvy programů s textovým rozhraním,
- webové odkazy.

Některá klíčová slova a termíny jsou zvýrazněny *kurzívou*. Bezpatkovým písmem jsou sázeny názvy vývodů integrovaných obvodů, signálů a dalších prvků ve schématu, případně i textový obsah prvků v grafickém rozhraní programů.

Část I

Teoretická část

Kapitola 1

Uvedení do problematiky MEMS mikrofonů

V této kapitole popisuji problematiku týkající se MEMS mikrofonů, která je stěžejním tématem této práce. Nejdříve je stručně popsán samotný MEMS mikrofon, následně jsou rozebrána rozhraní I²S a TDM.

1.1 MEMS mikrofon

Zkratka MEMS (Mikro elektro-mechanický systém) je označením pro elektro-mechanické struktury, jež jsou vyrobené na polovodičovém substrátu litografickými technologiemi, s obvyklými rozměry v řádu mikrometrů.

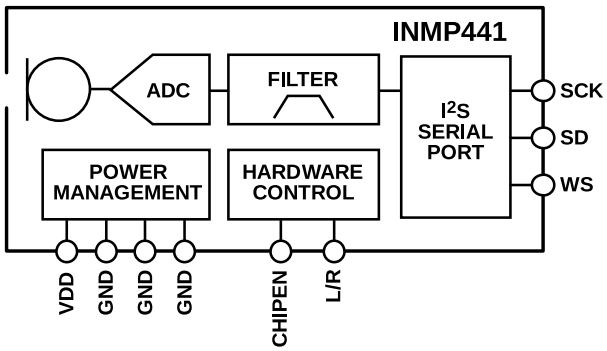
Tato práce je zaměřená na použití *MEMS mikrofonů*, kde mikrofon je integrován v jednom společném pouzdře spolu s podpůrnou elektronikou. Pouzdro mikrofonu je výhradně určeno pro povrchovou montáž, jelikož nabývá malých rozměrů, přibližně 4×3×1 mm [4]. Na spodní či vrchní straně pouzdra se nachází otvor pro vstup akustického signálu. Obrázek 1.1 znázorňuje MEMS mikrofon se spodním otvorem. Lze na něm také vidět, že vstupního otvor ohraničuje pájecí ploška ve tvaru prstence.

Blokové schéma MEMS mikrofonu je znázorněno na obrázku 1.2. Uvnitř takového mikrofonu je integrovaný analogově-digitální převodník, podpůrné obvody a číslicové komunikační rozhraní, přes které jsou z mikrofonu čteny vzorky.

Nejrozšířenějšími rozhraními u těchto mikrofonů jsou PDM (Pulse Density Modulation), I²S (Inter-Integrated Circuit Sound) a TDM (Time Division Multiplex) [5]. V této práci se zabývám pouze rozhraními I²S a TDM. Mikrofony s I²S rozhraním mají vstup L/R [6], na kterém se přivedením dané napěťové úrovně vybírá, zda mikrofon odesílá vzorek pro levý či pravý kanál. Vstup L/R je na obrázku 1.2 taktéž znázorněn.

Pro své malé rozměry nacházejí MEMS mikrofony uplatnění tam, kde je žádoucí miniaturizace, například v nositelné elektronice a mobilních telefonech. Konstrukce mikrofonního pole s těmito mikrofony dovoluje zahrnout více mikrofonů na omezené ploše (například povrch kulového mikrofonního pole).

Obrázek 1.1: Znázornění pouzdra MEMS mikrofonu se spodním otvorem [4]



Obrázek 1.2: Blokové schéma MEMS mikrofonu [6]

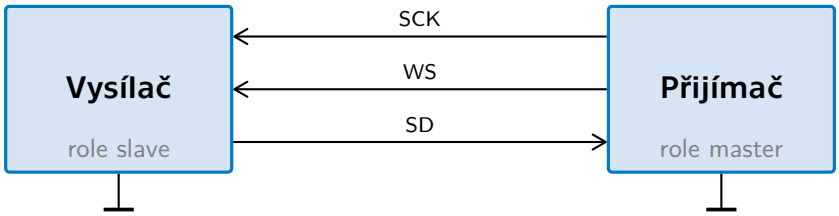
1.2 Rozhraní I²S

Rozhraní I²S (Inter-Integrated Circuit Sound) je digitální rozhraní určené pro přenos zvuku. Toto rozhraní vytvořili zaměstnanci firmy Philips, v roce 1986 byla vydána jeho specifikace [7]. Rozhraní může přenášet jeden (mono) či dva (stereo) zvukové kanály.

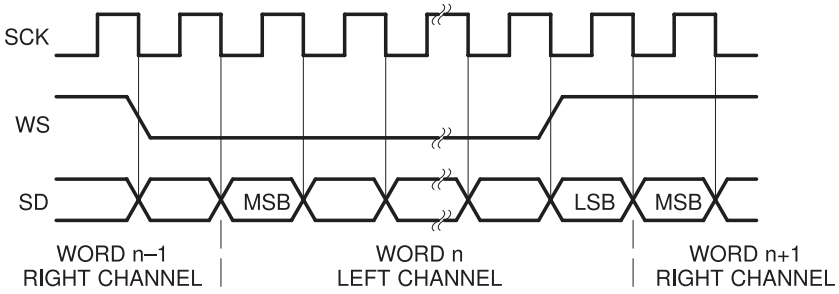
I²S používá 3 signálové vodiče:

- hodinový signál – má označení SCLK nebo SCK (Serial Clock),
- signál výběru kanálu – značený LRCLK (Left-Right Clock) nebo WS (Word Select),
- datový signál – SD (Serial Data).

Diagram 1.3 obsahuje znázornění komunikace prostřednictvím rozhraní I²S. Přijímací zařízení (receiver) v roli master je zdrojem signálů SCK a WS, protější vysílač (transmitter) je v roli slave a je zdrojem datového signálu SD. V případě, kdy FPGA čte z levého a pravého mikrofonu vzorky, je na levé straně dvojice mikrofonů v režimu slave a na pravé straně FPGA v režimu master. V roli master (tj. zdrojem signálů SCK a WS) může dle specifikace být namísto přijímače vysílač nebo i třetí zařízení. Zde takovéto možnosti však nejsou využity.



Obrázek 1.3: Diagram komunikace přes rozhraní I²S



Obrázek 1.4: Časové průběhy signálů I²S [7]

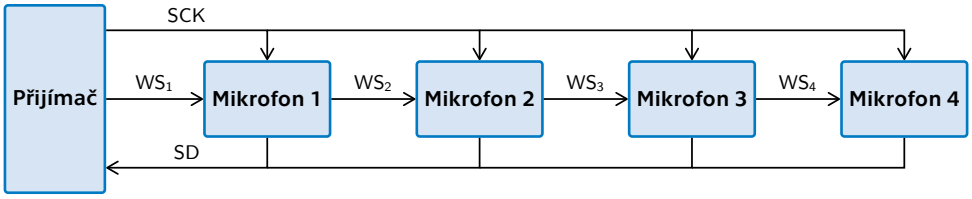
Časové průběhy signálů jsou na obrázku 1.4. Signál SCK přivádí do mikrofonů hodinový kmitočet. V případě čtení dvou 32bitových kanálů vzorkovací frekvencí f_{vz} je požadovaný kmitočet hodin $f_{SCK} = 2 \cdot 32 \cdot f_{vz}$. Například pro vzorkovací kmitočet $f_{vz} = 48$ kHz je $f_{SCK} = 3,072$ MHz.

Každou vzestupnou hranu hodin je čten jeden bit vzorku na vodiči SD. Vzorky jsou čteny od nejvýznamnějšího bitu (MSB) po nejméně významný (LSB). V případě, kdy je signál WS v úrovni logické 0, je čten levý kanál. Zvedne-li se úroveň WS na logickou 1, bude ještě jeden takt hodin čten levý vzorek, až poté se započne čtení pravého vzorku. Signál WS má tedy o jeden cyklus hodin předstih před datovým signálem.

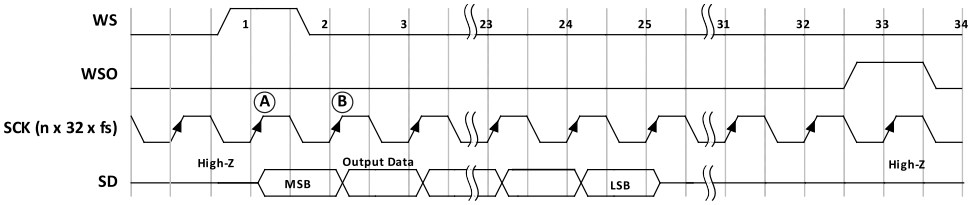
1.3 Rozhraní TDM

Přestože označení TDM (Time Division Multiplex) obecně označuje komunikaci s časovým dělením (multiplexem), ve spojitosti s MEMS mikrofony značí specifické rozhraní, jež některé mikrofony (např. InvenSense ICS-52000) využívají. Jedná se v principu o modifikaci I²S, kdy mikrofony mají signál WS (někdy také značený FSYNC – Frame Synchronization) zapojený kaskádně. Toto rozhraní podporuje až 16 takto zřetězených mikrofonů [8]. Obrázek 1.5 ukazuje zapojení řetězce čtyř TDM mikrofonů k obvodu, který od nich přijímá zvuková data.

Průběh TDM komunikace v čase je znázorněn na obrázku 1.6. Na začátku tzv. *rámce* (frame) – rámec označuje skupinu vzorků, ve které je z každého mikrofonu v řetězci obsažen právě jeden vzorek – vyšle nadřazený obvod (master) pulz na signál WS o délce alespoň jedné periody hodin. Tímto je první mikrofon v řetězci aktivován, přičemž odešle v následujících dvaatřiceti hodinových cyklech (jeden tzv. *slot*) vzorek.



Obrázek 1.5: Ukázka řetězce čtyř mikrofonů na rozhraní TDM



Obrázek 1.6: Časové průběhy signálů TDM [8]

Na konci časového slotu je vyslán pulz z výstupu prvního mikrofonu s názvem WSO (Word Select Output) na vstup WS druhého mikrofonu, čímž dochází k přenesení druhého vzorku. Takovýmto způsobem pulz na WS doputuje až na konec kaskády, kde zanikne. Nadřazený obvod poté zahájí další rámec vysláním pulzu na WS. Aby nadřazený obvod správně určil, kdy má znovu vyslat pulz, musí mít informaci o počtu mikrofonů zapojených za sebou.

Výhodou tohoto rozhraní je malý počet vodičů – celý řetězec mikrofonů lze připojit třemi signály. Nevýhodou tohoto řešení je to, že úměrně s počtem mikrofonů v řetězci roste potřebný hodinový kmitočet k dosažení žádané vzorkovací frekvence. Dle datového listu [8] se kmitočet hodinového signálu určí nikoliv podle skutečného počtu mikrofonů, ale zaokrouhlením počtu mikrofonů v řetězci nahoru na nejbližší mocninu dvou. To znamená, že řetězec dlouhý například 5 mikrofonů má shodný kmitočet hodin SCK jako řetězec 8 mikrofonů.

V případě čtení dvou 32bitových slotů vzorkovací frekvencí $f_{\mathrm{vz}} = 48$ kHz je požadovaný kmitočet hodin

$$f_{\mathrm{SCK}} = 2 \cdot 32 \cdot f_{\mathrm{vz}} = 3,072 \text{ MHz},$$

stejně jako u rozhraní I²S. V případě čtení šestnácti slotů požadovaný kmitočet hodin vzroste:

$$f_{\mathrm{SCK}} = 16 \cdot 32 \cdot f_{\mathrm{vz}} = 24,576 \text{ MHz}.$$

Přenos dat přes digitální rozhraní zvyšuje oproti analogovému odolnost proti okolnímu rušení. Další výhodou je jednodušší obvodová realizace, jelikož není nutné zesilovat signál. Limitující jsou však vysokofrekvenční vlastnosti přívodních vodičů, které bez impedančního přizpůsobení při vysokých hodinových kmitočtech digitální signál zkreslují. V tomto ohledu je rozhraní I²S vhodnější pro použití v mikrofonním poli než TDM, jelikož pracuje na nižších kmitočtech.

Kapitola 2

Komunikace s počítačem přes rozhraní USB

Před započetím této práce jsem USB znal pouze z uživatelského hlediska, nikoliv implementačního. Abych byl schopen implementovat USB zařízení, studoval jsem charakteristiky tohoto rozhraní na více úrovních jeho komunikačního protokolu. Relevantní informace týkající se implementace USB zařízení v této kapitole stručně uvedu. Ve druhé části kapitoly je popsán čip FX2LP, jehož hardware implementuje USB komunikaci. Poslední část je věnována analýze USB komunikace ze strany počítače a možnosti hledání případných problémů.

2.1 Rozhraní USB

USB (Universal Serial Bus) je univerzální sériové rozhraní pro přenos dat. Tvoří jej právě jedno zařízení v roli řídicí (*hostitelský systém*, anglicky *host*), což je obvykle počítač, a jedno nebo více podřízených zařízení (*periférie*, v angličtině *device*). Každé zařízení má svou vlastní adresu přidělenou řídicím zařízením. Žádné podřízené zařízení nemůže odesílat data do počítače, dokud k tomu není vyzváno [9].

Rozhraní USB je standardizované a jeho specifikace je volně přístupná. Tato práce pracuje s USB verze 2.0 [10]. Dnes se na počítačích již běžně setkáváme s USB verze 3.0, které je však zpětně kompatibilní s verzí 2.0, tudíž není problém nižší verzi používat i nadále.

2.1.1 Typy USB přenosů

USB podporuje 4 různé druhy přenosu dat [9]:

- *Řídicí přenos* (Control transfer) – používá se k ovládání USB zařízení pomocí řídicích dotazů (Control requests).
- *Přenos při přerušení* (Interrupt transfer) – používá se k periodickému přenosu krátkých dat. Řídicí systém se periodicky dotazuje na nová data.
- *Hromadný přenos* (Bulk transfer) – určen k přenosu velkého objemu dat s kontrolou chyb, nemá však garantovanou dobu zpoždění. Tento přenos má nízkou prioritu.

- *Isochronní přenos* (Isochronous transfer) – přenáší velký objem dat definovanou přenosovou rychlostí (která je zaručena). Používá se tam, kde je vyžadováno malé zpoždění (např. zvukové karty) na úkor opravy chyb, jež se neprovádí.

2.1.2 Deskriptor

Deskriptor je hierarchická datová struktura, která popisuje vlastnosti daného USB zařízení. Základem je *deskriptor zařízení* (Device Descriptor), který uvádí obecné vlastnosti zařízení. Jeho součástí jsou i parametry identifikátor dodavatele (VID) a identifikátor produktu (PID). Identifikátor VID uděluje organizace USB Implementers Forum a stojí 5000 amerických dolarů ročně [11].

Zařízení může mít jednu či více konfigurací, každou popsanou *deskriptorem konfigurace* (Configuration Descriptor). Každá konfigurace má jedno či více rozhraní, popsané *deskriptorem rozhraní* (Interface Descriptor). U každého rozhraní se dále specifikují koncové body *deskriptorem koncového bodu* (Endpoint Descriptor).

2.1.3 Struktura deskriptoru

Deskriptor se skládá z několika *datových polí*, jejichž počet a datový typ závisí na konkrétním deskriptoru. Datová pole mají před názvem předponu (prefix), která určuje datový typ pole. Například pole `bLength` je osmibitové číslo (byte) – může tedy nabývat hodnot od 0 do 255 – a udává délku deskriptoru. Obdobně prefix `w` (word) značí šestnáctibitové číslo.

Speciální význam má pole s předponou `i`. To udává index textového řetězce v seznamu řetězců umístěném na samém konci deskriptoru. Kupříkladu pole `iProduct` ukazuje na řetězec obsahující název USB zařízení. Pokud má `iProduct` hodnotu větší než 0, přečte se ze seznamu řetězců odpovídající řetězec. Jinak, pokud by měl `iProduct` hodnotu 0, název by byl prázdný.

2.1.4 Koncové body

Koncové body jsou vázány ke konkrétnímu (logickému) rozhraní, které jich využívá. Jeden koncový bod již představuje jednosměrný datový tok mezi řídicím zařízením a periférií. Koncové body jsou číslovány indexem od 0 do 15 a značí se zkratkou EP (např. EP6 označuje koncový bod číslo 6).

Jsou rozlišovány dva typy koncových bodů: typ OUT posílá data ven z řídicího zařízení (host) do periférie (device), typ IN posílá data z periférie do řídicího zařízení [12]. Jedno zařízení může mít až 15 vstupních a 15 výstupních koncových bodů. Koncové body EP0 IN a EP0 OUT jsou vyhrazeny pro přenos řídicích dotazů, a nejsou do tohoto počtu zahrnuty [10].

2.1.5 Pakety a mikrorámce

Elementární jednotkou USB datového přenosu je (datový) *paket* (packet). Maximální velikost paketu je určena standardem. Co se týče isochronního koncového bodu v režimu high-speed, ten umožňuje přenést v jednom paketu až 1024 bajtů.

Standard USB high-speed definuje tzv. *mikrorámec* (microframe) jako základní časové okno pro přenos datových paketů. Jeden mikrorámec je dlouhý 125 µs. Zda budou přeneseny během tohoto rámce data je určeno druhem přenosu daného koncového bodu.

Například koncový bod s isochronním typem přenosu má možnost přenášet datový paket každých N mikrorámců. Proměnná N se vypočte z parametru `bInterval`, který je součástí deskriptoru isochronního koncového bodu, následujícím vztahem:

$$N = 2^{\texttt{bInterval}-1}$$

Parametr `bInterval` může nabývat hodnot od 1 do 16 [10].

Isochronní koncový bod s vysokou přenosovou rychlostí (*high-bandwidth endpoint*) dovoluje posílat až 3 pakety v jednom mikrorámci [13]. Tím se maximální objem dat přenesený za jeden mikrorámec zvyšuje na 3072 bajtů. Tento druh koncového bodu jsem však nevyužil, jelikož velikost paketu do 1024 bajtů byla v praxi dostačující.

2.2 Integrovaný obvod FX2LP pro USB komunikaci

Integrovaný obvod CY7C68013A, s označením FX2LP™, někdy také nazývaný EZ-USB® (Easy USB), je obvod speciálně určený pro USB aplikace. Tento čip vyrábí společnost Infineon (v minulosti Cypress). Obsahuje mikrořadič s osmibitovou architekturou 8051[1] běžící na taktu až 48 MHz [14]. Čip je vyráběn v několika různých pouzdrech pro povrchovou montáž, a to s 56, 100 nebo 128 vývody [15]. K dosažení cílů této práce vystačuje provedení s 56 vývody.

2.2.1 USB komunikace

Mikrořadič podporuje USB rozhraní verze 2.0 v režimu full-speed (12 Mb/s) a high-speed (480 Mb/s). Integrovaný mikrořadič slouží ke konfiguraci a řízení USB komunikace. Data, která se mají posílat či číst z USB, jsou posílána obousměrnou paralelní sběrnicí – značenou USB FIFO (First In First Out). Zkratka FIFO značí paměť typu fronta, která je v obvodu přítomna a slouží jako vyrovnávací paměť při komunikaci. Její velikost pro jednotlivé koncové body závisí na konfiguraci čipu ve firmwaru.

[1] Architektura 8051 byla původně vyvinuta společností Intel.

Tento obvod se může tvářit jako libovolné USB zařízení, což je určeno takzvaným *deskriptorem*. Deskriptor je obsažen v paměti programu mikrořadiče 8051 (je zkompilován jako součást firmwaru).

2.2.2 Firmware integrovaného mikroprocesoru

Vyvíjet firmware (tj. program běžící na interním procesoru) pro tento čip lze v jazyce C a assembleru. Zvolil jsem kompilátor `sdcc` (Small Device C Compiler)[2], který má otevřený zdrojový kód a bývá dostupný jako balíček v linuxových distribucích.

S použitím knihovny `fx2lib`[3], která je licencována pod svobodnou licencí GNU Lesser General Public License (LGPL), jsem nemusel začínat psaní kódu od nuly, ale vycházel jsem z příkladů v této knihovně. Knihovna `fx2lib` se stará o definici názvů registrů procesoru, vektorů přerušení, maker a pomocných funkcí, které usnadňují obsluhu čipu a USB komunikace.

Firmware procesoru může být uložen na externí EEPROM paměti s rozhraním I²C, ale postačuje jej nahrát do paměti RAM (Random Access Memory) přes samotné USB rozhraní, čehož jsem také využíval. Tato interní paměť je velikosti 16 kB a je společná pro data i program. Firmware v paměti RAM je vymazán vynulováním (resetem) čipu nebo odpojením napájení.

2.3 Ladění USB komunikace na PC s operačním systémem Linux

Tato práce je cílena tak, aby byla kompatibilní především s operačním systémem Linux. Pro Linux existují nástroje, které pomáhají řešit problémy s USB zařízeními, a to z různých úrovní pohledu.

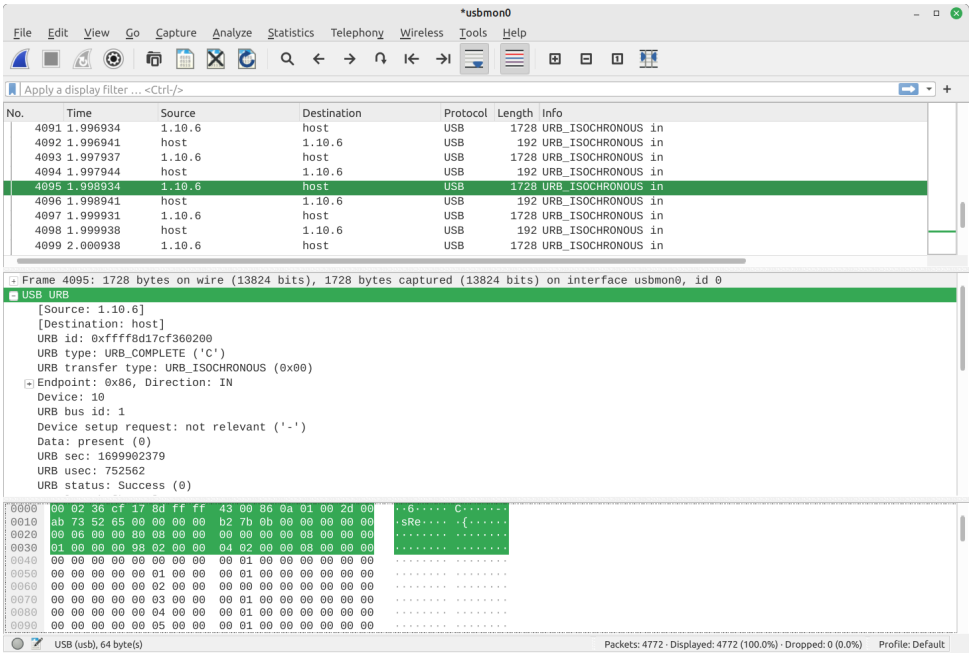
Pro základní kontrolu, zda USB zařízení bylo rozpoznáno systémem, lze použít příkaz `lsusb`, jenž vypíše seznam právě připojených (a rozpoznaných) USB zařízení. Informace o rozpoznání nově připojených USB zařízení a přiřazení ovladačů je možné sledovat v záznamu systému, například příkazem `journalctl -f`.

Pokud se zařízení má chovat jako vstupní zvuková karta, lze toto ověřit příkazem `arecord -l`, který vypíše detekované vstupní audio zařízení. Nástroj `arecord` dále umožňuje pořídit zvukový záznam a je součástí zvukového subsystému ALSA (Advanced Linux Sound Architecture)[4]. Ověřit tuto skutečnost lze většinou i v grafickém prostředí, tento způsob se však mezi linuxovými distribucemi liší.

2.3.1 Sledování USB paketů pomocí programu Wireshark

Probíhající USB komunikaci lze ladit pomocí programu Wireshark, jenž je dostupný zdarma a má otevřený zdrojový kód. Wireshark poskytuje rozbor

[2] `http://sdcc.sourceforge.net/`
[3] `https://github.com/djmuhlestein/fx2lib`
[4] `https://www.alsa-project.org`



Obrázek 2.1: Snímek okna programu Wireshark

jednotlivých paketů v grafickém prostředí. Tímto způsobem lze odhalit například chybný deskriptor, který způsobí, že zařízení není systémem rozpoznáno. Princip činnosti spočívá v tom, že po spuštění záznamu je zachytáván každý USB paket na vybraném USB monitoru (tj. komponentě, která se stará o sledování paketů). Na obrázku 2.1 je snímek okna tohoto programu.

Následující příkazy ke zprovoznění sledování USB komunikace na Linuxu byly převzaty z dokumentace programu Wireshark [16]. Před spuštěním tohoto programu je zapotřebí (s právy superuživatele) načíst modul `usbmon` do jádra systému, jenž zprostředkovává USB monitory.

```
sudo modprobe usbmon
```

Aby měl Wireshark přístup k USB monitorům, je vyžadováno nastavit oprávnění číst z nich pro přihlášeného uživatele. Tento krok spolu s předchozím je nutné provést po každém restartu systému.

```
sudo setfacl -m u:$USER:r /dev/usbmon*
```

Následujícím příkazem v terminálu se již otevře okno s programem.

```
wireshark &
```

V otevřeném okně už zbývá vybrat jeden z USB monitorů s názvem `usbmon*`, kde `*` značí číslo monitoru, a spustit záznam kliknutím na ikonu modré ploutve. Záznam se ukončí kliknutím na vedlejší červený čtverec. Správný monitor, který sleduje USB port s laděným zařízením, lze zjistit vypojením a zapojením tohoto zařízení, čímž se objeví na odpovídajícím monitoru inicializační pakety.

Kapitola 3

Konfigurace programovatelného hradlového pole

Programovatelné hradlové pole, často značeno anglickou zkratkou FPGA (Field Programmable Gate Array), je uživatelsky konfigurovatelný logický integrovaný obvod. Obsahuje řadu logických funkčních bloků, jejichž chování a vzájemné propojení je možné naprogramovat.

Dvěma hlavními výrobci programovatelných hradlových polí na trhu jsou v současnosti Intel (dříve Altera) a AMD (dříve Xilinx). S hradlovým polem Xilinx® jsem pracoval již v rámci své bakalářské práce [3], proto jsem tohoto výrobce zvolil i pro tuto práci. Při tvorbě této diplomové práce jsem své znalosti prohloubil, především v tom směru, kdy se FPGA umístí na vlastní plošný spoj, namísto toho aby se použila již hotová vývojová deska.

Klíčovou znalostí je zde způsob, jakým FPGA vůbec naprogramovat (ve smyslu přenést na něj svůj [syntetizovaný] návrh). Tuto problematiku, kterou jsem pro potřeby praktické realizace desky studoval, popíši v následujících sekcích.

FPGA Xilinx Spartan-3E, které v práci používám, má podle datového listu [17] šest různých způsobů načtení konfigurace. Z nich jsem vybral dva, které byly pro mou aplikaci nejpraktičtější – sběrnice JTAG a paměť typu flash s rozhraním SPI (Serial Peripheral Interface). V této kapitole popíši rozhraní JTAG a problematiku související s konfigurací FPGA. Implementační hledisko je rozebráno, spolu s připojením SPI flash paměti, v sekci 7.4 praktické části.

3.1 Sběrnice JTAG

JTAG (Joint Test Action Group) je standardizovaná testovací sběrnice určená pro integrované obvody (značená také IEEE 1149.1). Tato sběrnice vznikla z toho důvodu, aby bylo možné jednoduše přistupovat k vývodům čipu – například stimulovat vstupy a odečítat odezvu na výstupech, což není u čipu připájeném na plošném spoji vždy snadné.

Informace o této sběrnici jsem čerpal z knihy Testability Primer od Texas Instruments [18]. Princip JTAG je založen na tzv. „skenu okrajů“ (*boundary scan*). Tento název poukazuje na způsob činnosti (boundary, neboli okraj čipu zde označuje digitální vstupy a výstupy integrovaného obvodu). Sběrnice

využívá sériových posuvných registrů umístěných na čipu. Skenem se rozumí nasouvání dat do těchto registrů a současné vyčítání z druhého konce.

Tyto posuvné registry jsou přítomny u některých vstupů a výstupů čipu, čímž je umožněno k nim jednotlivě přistupovat. Dnes se používají i např. k přístupu do interní paměti čipu (tak je tomu u hradlových polí Xilinx). Posuvné registry jsou průchozí, čipy lze tudíž spojovat do řetězců. Na sběrnici může být teoreticky neomezený počet čipů spojených do řetězce.

Základem sběrnice JTAG je tzv. *Test Access Port* (TAP), jehož součástí jsou čtyři signálové vodiče:

- TDI (Test Data Input),
- TDO (Test Data Output),
- TMS (Test Mode Select),
- TCK (Test Clock).

Tyto signály jsou řízeny ve standardu definovaným stavovým automatem.

Uvnitř čipu jsou dvě datové cesty – jedna přes instrukční registr a jedna přes zvolený datový registr. To, jaká cesta bude zvolena určuje stav signálu TMS a stavového automatu JTAG při náběžné hraně hodin TCK. Signál TDI nese bitovou informaci, která se nasune do zvolené datové cesty. Vodič TDO je výstup z konce registru, do kterého se nasouvají data, jeho hodnota se mění sestupnou hranou hodinového signálu TCK.

3.2 Konfigurace hradlového pole Xilinx pomocí sběrnice JTAG

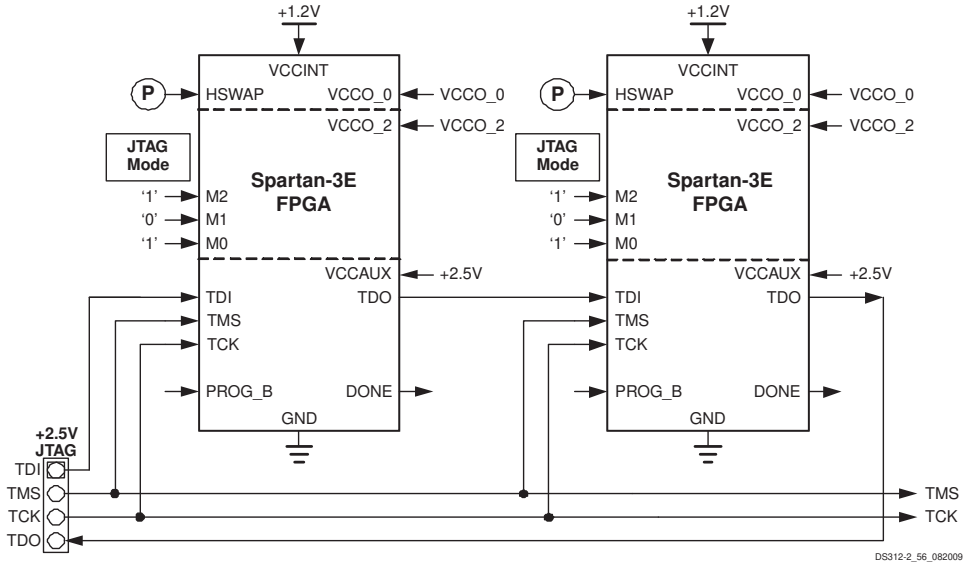
Co se týče hradlových polí Xilinx, lze do nich skrze JTAG port nahrát bitovou konfiguraci. *Bitovou konfigurací* je myšlen binární formát dat popisující vnitřní konfiguraci čipu FPGA (anglicky *bitstream*). Bitová konfigurace je výsledkem syntézy návrhu z jazyka popisujícího hardware (např. VHDL či Verilog) pro dané hradlové pole. Kromě zdrojových souborů je také vyžadováno určit přiřazení vývodů – v Xilinx ISE se toto provede souborem UCF (User Constraints File).

Konfigurací hradlového pole se může myslet jak bitová konfigurace, tak vlastní akt nahrávání této konfigurace na hradlové pole (tj. „naprogramování“ FPGA). Nahraná konfigurace je na FPGA uložena pouze dočasně – po vynulování čipu (reset) či odpojení napájení je ztracena.

Příklad, jak může vypadat zapojení dvou hradlových polí Spartan-3E na jedné sběrnici JTAG ukazuje obrázek 3.1. Pozornosti by neměla ujít skutečnost, že vývody JTAG očekávají logické napěťové úrovně CMOS 2,5 V, což komplikuje použití v obvodech s běžně používanými úrovněmi 3,3 V.

3.2.1 Formát souboru SVF

Formát souboru SVF (Serial Vector Format) je standardní textový formát určený k přenositelnému popisu operací sběrnice JTAG. Soubor tohoto formátu



Obrázek 3.1: Schéma konfigurace hradlového pole Spartan-3E pomocí rozhraní JTAG [17]

se obvykle značí příponou `.svf`. V souboru jsou obsažena data, která se mají po sběrnici JTAG nasunout do cílového řetězce čipů. Operace v takovémto souboru lze reprodukovat tzv. SVF interpretem, který posloupnost operací na sběrnici JTAG provede. Nástroj určen primárně pro nahrávání hradlových polí Xilinx – iMPACT, který je součástí vývojového prostředí Xilinx ISE, podporuje export tohoto formátu [19]. To umožňuje použít libovolný hardware, který dokáže přehrát SVF soubor, k nahrání konfigurace na FPGA přes JTAG port.

3.3 Syntéza VHDL projektu v Xilinx ISE

Níže je uveden postup, jakým lze syntetizovat VHDL projekt ve vývojovém prostředí Xilinx ISE. Výsledkem syntézy je bitová konfigurace FPGA (binární soubor `.bit`). Bude využito grafického prostředí Xilinx Project Navigator.

Postup syntézy je následující:

1. Otevření projektu v programu Project Navigator – v horní liště se zvolí File → Open Project…
2. V otevřeném dialogovém okně prohlížeče souborů se otevře soubor `.xise` příslušející vybranému VHDL projektu. Výběr se potvrdí tlačítkem Open.
3. V levém horním postranním okně by měla být výběrem zvýrazněna hlavní entita VHDL návrhu. Pod tímto oknem, kde je nabídka různých návrhových kroků, dvojklikem spustit položku Generate Programming File. Tím se spustí řetězec syntézy, který generuje soubor bitové konfigurace ze zdrojového kódu.
4. Po úspěšném dokončení všech operací vznikne příslušný soubor `.bit` v kořenovém adresáři VHDL projektu.

Část II

Praktická část

Kapitola 4

Praktické cíle práce

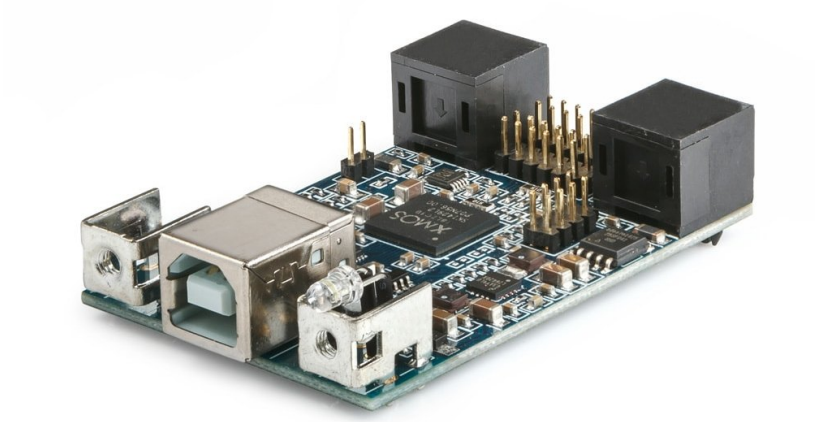
Cílem této diplomové práce bylo seznámit se s principy funkce MEMS mikrofonů a navrhnout řešení pro čtení vzorků z vícekanálového mikrofonního pole (až 16 mikrofonů) pomocí FPGA a následné posílání těchto vzorků do počítače přes rozhraní USB. Každý zvukový kanál je tvořen jedním MEMS mikrofonem, který má digitální rozhraní I²S nebo TDM. Navržený systém by měl zvládnout posílat vzorky získané vzorkovacím kmitočtem až 48 kHz.

4.1 Obdobné řešení dostupné na trhu

Zařízení, které zastává obdobnou úlohu jako deska vyvinutá v rámci této práce je USBStreamer Kit [20] – na obrázku 4.1. USBStreamer byl před několika měsíci ještě dostupný, v době dokončování tohoto textu (prosinec 2023) je však v oficiálním internetovém obchodě vyprodán. Přípravek USBStreamer bylo možné zakoupit za cenu 105 amerických dolarů (bez daně z přidané hodnoty).

USBStreamer využívá signálový procesor XMOS, který zajišťuje jak USB komunikaci, tak zpracování zvukových dat. Přípravek umí jak přijímat, tak vysílat zvukové signály. Podporovaná rozhraní pro přenos zvuku jsou Toslink, ADAT, I²S a TDM. Rozhraní TDM tohoto přípravku zvládne přijímat či vysílat až 16 kanálů při vzorkovací frekvenci 48 kHz, I²S až 8 kanálů při 192 kHz. Oficiálně jsou podporovány operační systémy Windows 10, Linux a macOS.

Oproti tomuto komerčnímu řešení využívá mnou navržené zařízení programovatelné hradlové pole spolu s USB mikrořadičem. Flexibilita FPGA umožňuje přizpůsobení vnitřní implementace na úrovni logických hradel, signálový procesor takovouto úroveň přizpůsobení neposkytuje. Připojení mikrofonního pole je u mnou navržené desky snazší, protože byl pro každý mikrofon vyhrazen vlastní konektor. Přípravek USBStreamer by v tomto případě vyžadoval tvorbu vlastního adaptéru na dvouřadou kolíkovou lištu (2×6 kolíků), která sdružuje všech 8 I²S rozhraní.



Obrázek 4.1: Komerční přípravek USBStreamer Kit [20]

4.2 Rozbor problematiky komunikace s počítačem

Posílání vícekanálového zvukového signálu klade vyšší nároky na přenosovou rychlost, než přenos běžného stereofonního zvuku. Požadavek na minimální přenosovou rychlost R_p pro tok dat z 16 MEMS mikrofonů do počítače, se vzorkovací frekvencí 48 kHz:

$$R_\mathrm{p} = 16 \cdot 32 \cdot 48000 = 24,576 \text{ Mb/s}.$$

Z nejrozšířenějších rozhraní PC se vzhledem k požadavku na rychlost jako vhodné jeví buď USB, nebo síťové rozhraní Ethernet. Kupříkladu rozhraní UART (Universal Asynchronous Receiver/Transmitter), s nímž jsem pracoval v rámci své bakalářské práce [3], vzhledem k potřebné přenosové rychlosti nepřipadá v úvahu, jelikož s ním lze dosáhnout rychlosti maximálně několik jednotek Mb/s.

Z možných rozhraní jsem vybral USB verze 2.0, a to z několika důvodů:

- Více než dostatečná přenosová rychlost (až 480 Mb/s v režimu *high-speed*).
- Konektor USB přivádí také napájení 5 V, zatížitelné proudem až 500 mA.
- Pro přenos zvuku je v USB standardu třída audio zařízení (USB Audio Class), jejíž implementace by odstranila potřebu speciálních ovladačů.
- Jednodušší fyzická implementace na plošném spoji.
- Jedná se o často používané rozhraní pro lokální přenos dat do počítače.

Za účelem realizace USB komunikace byl vybrán mikrořadič FX2LP, jenž lze využít pro vysokorychlostní USB komunikaci přes vlastní paralelní rozhraní (USB FIFO), a to s podporou standardu USB 2.0 high-speed.

Kapitola 5

Implementace na FPGA

Logický návrh byl původně cílen na vývojovou desku Digilent Nexys 3™, která obsahuje hradlové pole Xilinx Spartan®-6. Později byl návrh převeden na vlastní desku s FPGA Spartan®-3E. Na desce Nexys 3 byl oscilátor o kmitočtu 100 MHz, což nebyl vhodný kmitočet pro cílenou vzorkovací frekvenci 48 kHz, jelikož na 48 kHz nelze převést vydělením celým číslem. To byl jeden z hlavních důvodů přechodu.

Návrh jsem psal v jazyce VHDL (Very High Speed Integrated Circuit Hardware Description Language) v souladu se specifikací z roku 1993. Přechod na jinou (příbuznou) architekturu FPGA nebyl z hlediska kódu problém. Hlavní změna spočívala ve změně souboru přiřazení vývodů `.ucf`. Zdrojové soubory jsou opatřeny komentáři, které VHDL kód vysvětlují.

Syntéza a simulace logického návrhu byla prováděna ve vývojovém prostředí[1] Xilinx ISE® Design Suite verze 14.7 se základní licencí WebPack™, která je dostupná zdarma (po registraci na webu výrobce). Projekt se zdrojovými soubory je vytvořen v součásti *Project Navigator*, kde lze také provést syntézu návrhu.

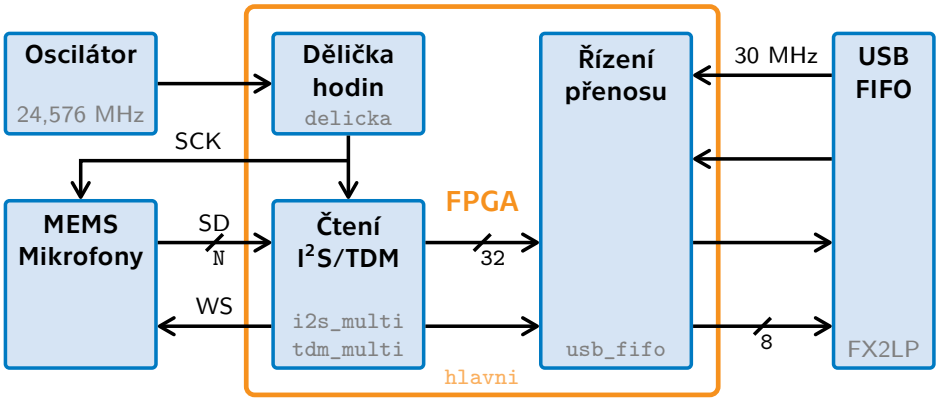
5.1 Analytický pohled na implementaci

Na FPGA byl implementační návrh rozdělen do tří hlavních funkčních celků, jak je znázorněno na obrázku 5.1. Nejsou zde zobrazeny všechny signály, ale jen naznačeny ty zásadní, z důvodu úspory místa. Jednotlivé bloky mají ve spodní části uveden název VHDL entity, která daný blok implementuje. Zdrojové VHDL soubory (s příponou `.vhd`) mají název shodný s entitou, která je v nich implementována.

Přeškrtnutí šipky signálu (na obrázku) indikuje vícebitový signál. Číslo pod přeškrtnutím udává bitovou šířku tohoto signálu. U signálu SD je místo čísla proměnná `N`, jež vyjadřuje počet připojených audio rozhraní.

Speciální význam má (hlavní) entita `hlavni`, která zastřešuje všechny entity návrhu a propojuje je mezi sebou. Entita `hlavni` také specifikuje vstupy a výstupy návrhu, které se následně přiřadí souborem `MicArrayBoard.ucf` fyzickým vývodům hradlového pole Spartan-3E.

[1] `https://www.xilinx.com/products/design-tools/ise-design-suite.html`



Obrázek 5.1: Blokové znázornění obvodu realizovaného na FPGA

Zdrojem hodinového kmitočtu návrhu je oscilátor o kmitočtu 24,576 MHz, jenž je osazen na desce MicArrayBoard. Vstupní kmitočet je v hradlovém poli následně dělen podle požadavků na celkový počet mikrofonů a vzorkovací kmitočet. Blok „Čtení I2S/TDM“ ve schématu reprezentuje funkční celek, který neustále čte vzorky z daného rozhraní pro přenos zvuku. Přečtený vzorek je poté předán do bloku „Řízení přenosu“, který implementuje předávání dat po sběrnici USB FIFO.

Specificky pro čtení několika I²S rozhraní současně byl vytvořen VHDL projekt `MicArrayBoard_I2S_to_USB`. Obdobně pro čtení několik TDM rozhraní vznikl projekt `MicArrayBoard_TDM_to_USB`. Toto jsou dva hlavní návrhy, které cílí na desku MicArrayBoard. Kromě rozdílné implementace bloku „Čtení I2S/TDM“ se ostatní bloky těchto dvou návrhů liší pouze drobnostmi, proto budou popsány právě jednou. Návrh pro čtení dat z TDM mikrofonů lze navíc v Xilinx ISE simulovat zabudovaným simulátorem iSim.

V souboru `konstanty.vhd`, který neobsahuje VHDL entitu, je definována vlastní knihovna konstant, kterou lze použít v celé hierarchii VHDL návrhu. Konstanty volí např. vzorkovací kmitočet, počet aktivních mikrofonních rozhraní a velikost USB paketu. Změna hodnot obsažených konstant vyžaduje určitou obezřetnost. Na jaká omezení je třeba dát zřetel je uvedeno v sekci 9.6.

5.2 Dělení kmitočtu

Vstupní kmitočet z oscilátoru je v bloku „Dělička“ (entita `delicka`) dělen celým sudým číslem. Hodinový signál je nejdříve přiveden na budič vstupního hodinového signálu `IBUFG`. Tato komponenta je specifická pro hradlová pole Xilinx, což omezuje přenositelnost kódu.

Dělička kmitočtu je implementována čítačem, který se každou půlperiodu výstupního taktu vynuluje a překlopí stav hodinového signálu. Výstupní kmitočet je potom použit jako hodinový signál I²S či TDM rozhraní. V bloku `delicka` je také generován globální nulovací signál `reset` aktivní v log. 1, a to posuvným registrem s přednastavením.

Dělicí poměr mezi vstupním kmitočtem oscilátoru a výstupním kmitočtem

udává konstanta `DELITEL`. Hodnota této konstanty je nastavena v knihovně `konstanty.vhd` funkcí, jež mapuje vzorkovací kmitočet na celočíselnou hodnotu dělitele. U rozhraní TDM je mapování dvouúrovňové, jelikož záleží i na délce řetězce mikrofonů.

Pokud vyjde hodnota dělitele 1, je blok s čítačem opomenut a vstupní hodinový signál je beze změny vyveden na výstup entity. Toho je docíleno podmínkovou konstrukcí `if generate` jazyka VHDL. Není-li dělitel sudý (nebo 1), syntéza selže díky příkazu `ASSERT`, který ověřuje sudost dělitele.

Například 2kanálové I²S, vzorkovací kmitočet 48 kHz vyžaduje hodinový kmitočet 3,072 MHz. Kmitočet se tedy bude dělit číslem:

$$\texttt{DELITEL} = \frac{24,576}{3,072} = 8.$$

TDM rozhraní o délce řetězce 4, vzorkováno kmitočtem 16 kHz bude mít dělitel hodnotu:

$$\texttt{DELITEL} = \frac{24,576 \cdot 10^6}{16000 \cdot 32 \cdot 4} = 12.$$

5.3 Čtení I²S mikrofonů

Projekt `MicArrayBoard_I2S_to_USB` implementuje čtení I²S mikrofonů. Blok „Čtení I²S“, jenž představuje entitu `i2s_multi`, zastává funkci přijímacího nadřazeného obvodu (role master) na rozhraní I²S, a to pro několik paralelních I²S rozhraní. Blok je taktován samotným kmitočtem hodinového taktu I²S.

Čtení vzorků je řízeno jednoduchým stavovým automatem, který periodicky střídá levý a pravý kanál. Signál `stav_i2s` udává aktuální stav I²S rozhraní z hlediska čteného kanálu. Může nabývat hodnoty buď `levy` nebo `pravy`.

Každou vzestupnou hranu hodin je přečten jeden bit zvukového vzorku z každého rozhraní. Bity jsou čteny ze vstupního bitového vektoru `SD`, jehož délka je dána počtem čtených rozhraní, do pole posuvných registrů (jeden pro každé rozhraní). Pokud bylo přečteno všech 32 bitů kanálu, je přepnut stav automatu a je čten opačný kanál. Vzorky z I²S mikrofonů zabírají pouze horních 24 bitů [2][6], zbylé bity nejsou využity. Mikrofony Knowles SPH0645LM4H-B mají navíc z těchto 24 bitů platných pouze 18 [2].

Signál `FSYNC`, společný pro všechna rozhraní, je řízen v samostatném VHDL procesu reagujícím na sestupnou hranu hodin. Podle budoucího stavu signálu `stav_i2s`, kterého nabyde po následující vzestupné hraně hodin je zapsána log. 0 v případě levého nebo log. 1 v případě pravého kanálu.

5.3.1 Předávání dat

Po přečtení 32 bitů jednoho kanálu jsou tato data sekvenčně předávána entitě `usb_fifo`, která řídí sběrnici USB FIFO. Najednou lze entitě předat pouze jeden vzorek, proto je předávání sekvenční. Pokaždé, když jsou připraveny vzorky ze všech levých nebo pravých kanálů, zahájí se předávací sekvence.

Předávací sekvence probíhá následovně:

1. Vzorek rozhraní s indexem 0 se zapíše na výstup `DATA`. Na výstup `DATA_zapis` je zapsán jedničkový pulz, který oznamuje nově přečtený vzorek. Entita `usb_fifo` na pulz zareaguje a zahájí zápis do USB FIFO, signál `pozadavek` z této entity přejde do úrovně log. 1.

2. Čeká se, až vstupní signál `pozadavek`, vyjadřující probíhající zápis do USB FIFO, poklesne zpět k nule.

3. Pokud se tak stane, přejde se na vzorek následujícího rozhraní. Odeslal-li se vzorek posledního rozhraní, je sekvence ukončena.

Předpokládá se to, že předávání dat komponentě `usb_fifo` proběhne dříve, než se přečtou vzorky z opačného kanálu. Vzhledem k tomu, že komponenta `usb_fifo` pracuje na 30 MHz, lze toto s přehledem zajistit, vezmeme-li v úvahu, že z I²S rozhraní jsou čteny vzorky frekvencí desítek kHz.

Tři bajty s nejvyšší vahou, které jsou obsaženy ve vzorku, obsahují 24bitovou hodnotu vzorku. Poslední bajt má pouze pomocnou úlohu v této implementaci. Z toho horních 7 bitů udává index mikrofonu (ve formátu binárního čísla). Reálně se využijí nanejvýš 4 bity, což odpovídá rozsahu indexů 0 až 15 (zbylé bity jsou tedy nulové). Do nejnižšího bitu vzorku je zapsán bit o hodnotě 0, pokud byl přečten z levého kanálu, popřípadě 1, pokud byl přečten z pravého. Tyto informace slouží pro případnou identifikaci kanálu v dalším zpracování.

5.3.2 Ověření činnosti v praxi

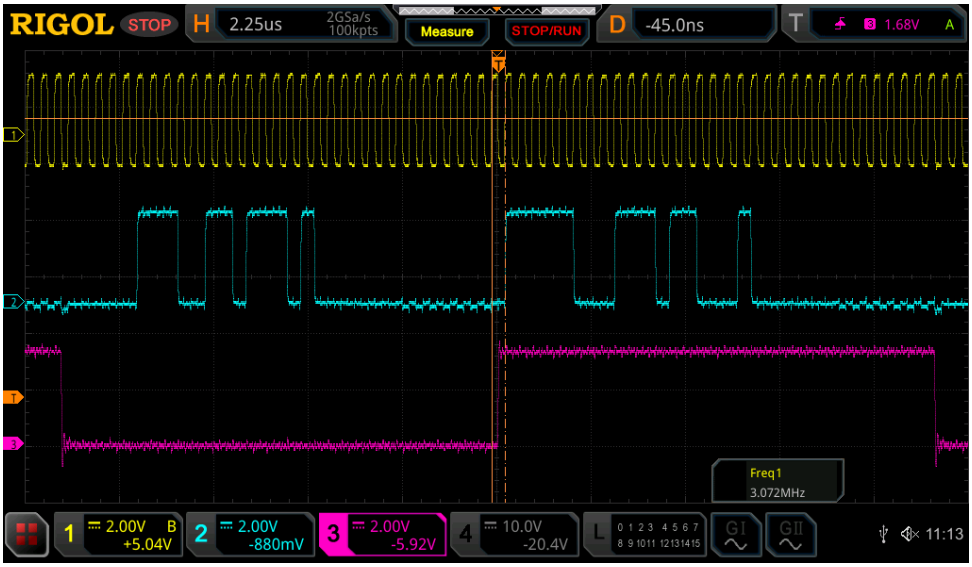
Funkčnost řešení byla ověřena jak v simulaci, tak v praxi. Obrázek 5.2 zobrazuje snímek probíhající I²S komunikace zachycený na čtyřkanálovém osciloskopu Rigol MSO5074. Žlutou barvou je vykreslen hodinový signál SCK, modrou barvou datový signál SD a fialový je signál FSYNC. MEMS mikrofony jsou na kruhovém mikrofonním poli a jsou připojeny k desce MicArrayBoard, která tyto mikrofony čte vzorkovacím kmitočtem 48 kHz, což je horní limit pro mikrofony Knowles SPH0645LM4H-B [2].

5.4 Čtení TDM mikrofonů

Obdobně jako I²S funguje i čtení TDM rozhraní, s tím rozdílem, že zde nejsou dva kanály, ale až 16 slotů a signál WS posílá pulz na začátku 1. slotu. Sloty se po vyslání pulzu postupně čtou, po přečtení všech se sekvence opakuje.

Počet slotů (tj. počet mikrofonů v TDM řetězci) se nastaví v souboru `konstanty.vhd` konstantou `MIKROFONU`. Aby bylo možné připojit více TDM řetězců, jsou pro čtení TDM mikrofonů využity konektory I2S-0 až I2S-7[2].

[2]Konektor TDM tedy zůstává nevyužitý, ale je možné na něj vývody přemapovat v souboru `.ucf`, pokud uživatel bude požadovat jediné TDM rozhraní.



Obrázek 5.2: Probíhající I²S komunikace MEMS mikrofonů s FPGA zachycená osciloskopem

5.4.1 Simulace TDM mikrofonů

Jelikož jsem měl k dispozici lehce nespolehlivé mikrofonní pole s TDM mikrofony (stávalo se, že mikrofony z řetězce vypadávaly), byla raná verze návrhu testována na přípravku se signálovým procesorem ADAU1452 od společnosti Analog Devices (takto bylo testováno čtení jednoho TDM rozhraní).

Signálový procesor byl programován ve vývojovém prostředí Sigmastudio verze 4.6 na stolním počítači s operačním systémem Windows 10. Principem tohoto prostředí je propojování různých předem daných funkčních bloků. Přes jednoduchost tohoto programu však byl problém například zadat přesnou hodnotu vzorku v šestnáctkové soustavě, která by byla neustále posílána.

Simulaci TDM mikrofonů jsem později, z důvodu flexibility a možnosti vytvoření několika paralelních TDM řetězců, implementoval vlastními VHDL komponentami přímo na hradlovém poli. Simulované mikrofony odesílaly předem definovaný periodický průběh. Připojení simulovaných mikrofonů (komponenty `tdm_slave`) je implementováno v alternativní architektuře hlavní entity, nazvané `Simul`.

V Xilinx ISE se mi nepodařilo naleznout způsob, jakým mezi dvěma architekturami nejvyšší entity přepínat. Proto jsou změny kódu, které způsobí zastínění výchozí architektury `Structural` architekturou `Simul`, aplikovány ve složce `MicArrayBoard_TDM_to_USB_simul`, která jinak obsahuje shodný VHDL projekt. Pokud by uživatel chtěl takto pozměněný projekt použít, stačí překopírovat obsažené soubory do složky bez „`_simul`“ na konci.

Průběhy byly implementovány jako pole vzorků dané délky, k vygenerování vzorků jsem vytvořil skript v jazyce `lua`. Generovaná pole vzorků byla vložena do knihovny `prubehy.vhd`, kde jsou k dispozici tři druhy průběhů: obdélníkový, pilový a sinusový.

5.4.2 Čtení vícero TDM rozhraní

Návrh `MicArrayBoard_TDM_to_USB_simul` byl odzkoušen v praxi, kdy byly zkoušeny různé kombinace vzorkovacího kmitočtu, počtu TDM rozhraní a délky řetězců. Při 24 kHz se podařilo přijmout až 64 kanálů (8 rozhraní×8 mikrofonů), což je mezní konfigurace, kdy velikost paketu nepřekročí 1024 bajtů.

Při zvýšení vzorkovací frekvence na 48 kHz jsem narazil na omezení při délce řetězce nad 4 mikrofony, kdy přestávala navržená struktura předávat všechny vzorky. Omezení se mi po zkoušení různých možných řešení přeci jen podařilo odstranit.

Řešení spočívá ve vytvoření pomocného hodinového signálu `clk_data`, který vznikne vydělením kmitočtu TDM rozhraní podle hodnoty dělitele. Pokud je `DELITEL` = 2, pak se kmitočet vydělí na polovinu, pokud má dělitel hodnotu 1, pak se kmitočet dělí na čtvrtinu. V ostatních případech signál není využit.

Přechody stavů signálu `DATA_zapis` a několika dalších pomocných signálů jsou omezeny stavem hodin `clk_data`. K předání dalšího vzorku (jedničkový pulz na `DATA_zapis`) může dojít pouze tehdy, je-li `clk_data` v logické 1. Návrat signálu `DATA_zapis` do log. 0 je omezen na případ, kdy je `clk_data` v log. 0. Je tedy vyžadována opačná polarita pomocných hodin oproti předchozí změně stavu.

Takto zpomalené přechody učiní chování stavového automatu dostatečně spolehlivé na to, aby předávané vzorky byly skutečně přebrány entitou `usb_fifo`. Při vzorkovacím kmitočtu 48 kHz lze tedy délku řetězce nastavit na 16 mikrofonů a číst až dvě TDM rozhraní, či na 8 mikrofonů a číst až 4 rozhraní (celkem 32 zvukových kanálů).

5.5 Řízení sběrnice USB FIFO

Blok „Řízení přenosu“, jenž předává vzorky do mikrořadiče FX2LP po sběrnici USB FIFO, je realizován VHDL entitou `usb_fifo`. Sběrnice USB FIFO, blíže popsaná v sekci 6.2, je řízena pomocí signálů ze strany hradlového pole. Předávání jednoho vzorku do fronty je v hradlovém poli implementováno jako sekvence čtyř zápisů osmibitových hodnot. Jelikož USB očekává vzorky ve formátu little-endian, zapisují se bajty v pořadí od nejméně významného po nejvýznamnější.

Výstup `DATA_zapis` komponenty, jež čte data z mikrofonů, je připojen na vstup `DATA_WRITE` entity `usb_fifo`. Analogicky jsou mezi sebou propojeny 32bitové signály `DATA` obou komponent.

5.5.1 Popis stavového automatu

Předávací sekvence je řízena stavovým automatem, který řídí signály sběrnice USB FIFO. Ve výchozím stavu, `cekani`, se čeká na nový vzorek, indikovaný stavem logické 1 signálu `DATA_WRITE`.

Následující stav – `prodleva` – provádí ošetření pozice vzorků, a to pokud je obdržený vzorek prvním v paketu. Paket může být zahájen pouze vzorkem z levého kanálu 1. rozhraní (I2S-0) v případě I²S, v případě TDM prvním slotem 1. rozhraní.

Pokud je obdržený vzorek vyhovující, předávací sekvence pokračuje za podmínky, že příznak FLAGB není nastaven, tj. paměť fronty koncového bodu EP6 není plná. V takovém případě se na sběrnici zapíše USB FIFO první (nejméně významný) bajt vzorku, automat přechází do stavu `bajt1`.

Ve stavu `bajt1` je odeslán 2. bajt a dochází k přechodu do stavu `bajt2` kde se odešle 3. bajt. Obdobný je význam stavu `bajt3`. Indexy těchto stavů neodpovídají bajtům, které jsou odesílány, ale těm, které byly právě odeslány.

Po přechodu do koncového stavu `bajt4` je ukončen zápis bajtů. Pokud počet předaných vzorků právě odpovídá délce paketu (dané konstantou `VZORKU_NA_PAKET`), je signálem `PKTEND` ukončen paket. Tento signál vyhodnotí logika USB mikrořadiče a připraví právě ukončený paket na odeslání do počítače.

5.5.2 Synchronizace signálů

Jak je znázorněno na blokovém schématu 5.1, je entita `usb_fifo` taktována frekvencí 30 MHz. Tento takt je generován v čipu FX2LP, a je proto asynchronní vůči zbylým logickým blokům, které mají jako zdroj hodin oscilátor. Z tohoto důvodu je vzorek vstupující z čtecího obvodu (signál `DATA`) synchronizován dvoustupňovým synchronizátorem.

Synchronizátor sestává z kaskády několika registrů, které mají společné hodiny. To znamená, že asynchronní vstupní data jsou vzorkována cílovým kmitočtem. Každým cyklem hodin se do vstupu kaskády nasune nový vzorek a z druhé strany se vysune synchronizovaný signál.

Signál `DATA_WRITE` je synchronizován kaskádou registrů o parametrické délce, kterou udává konstanta `ZPOZDENI`. Název konstanty vyjadřuje fakt, že synchronizátor zpozdí signál o čas úměrný délce kaskády. Konstantu `ZPOZDENI` jsem nastavil na hodnotu 4, což způsobí mírné zpoždění signálu `DATA_WRITE` oproti signálu `DATA`, čímž se zajišťuje čtení již synchronizovaných dat kdykoliv je signál `DATA_WRITE` aktivován.

Kapitola 6

Komunikace desky MicArrayBoard s PC

Na desce MicArrayBoard se nachází USB čip FX2LP, který je připojen k FPGA Spartan-3E osmibitovou sběrnicí USB FIFO. V této kapitole bude věnována pozornost právě tomuto čipu, který zprostředkovává USB komunikaci desky s počítačem. Chování čipu FX2LP je určeno firmwarem jeho vnitřního mikroprocesoru, jenž je založen na modifikované osmibitové architektuře 8051. Budou zde rozebrány implementační podrobnosti firmwaru určeného pro posílání zvukových dat do počítače – `FX2LP_audio_ISO`.

6.1 Firmware čipu FX2LP

Vnitřní mikroprocesor 8051 má poměrně jednoduchou architekturu, proto je kód firmwaru psán na nízké úrovni – přímým přístupem k registrům. Detailní popis registrů obsahuje technický manuál FX2LP [14].

Firmware byl v minulosti nastaven tak, že vzorky byly přenášeny hromadným USB přenosem (typ *bulk*). Tento přenos byl nejjednodušší na implementaci vzhledem k dostupnému příkladu v knihovně. Protože však tyto pakety nemají zaručenou přenosovou rychlost, nebyl tento přenos příliš vhodný pro přenos zvuku. Toto jsem zjistil, když se v přenosu referenční sinusovky objevovaly výpadky vzorků (jejich četnost rostla se vzorkovacím kmitočtem).

Nynější firmware – `FX2LP_audio_ISO` – využívá isochronní přenos, a v deskriptoru je implementována třída USB audio zařízení verze 1.0. Implementaci této třídy USB zařízení jsem převzal z návodu od společnosti Silicon Labs [21], který se zabývá implementací USB zařízení třídy audio. Převzatý deskriptor jsem poupravil pro potřeby tohoto projektu. Zdrojový kód `FX2LP_audio_ISO` je zveřejněn pod licencí GNU GPL (General Public License) v2[1].

Během vývoje firmwaru jsem využíval program Wireshark ke sledování USB komunikace (popsáno v sekci 2.3.1). Pomohlo mi to v počátku správně vytvořit strukturu deskriptoru a nastavit parametry isochronního přenosu pro koncový bod EP6.

[1] `https://gitlab.com/micarray-fpga/fx2lp-audio-iso`

6.1.1 USB deskriptor

Dle USB standardu, datové struktury o velikosti několika bajtů jsou strukturovány způsobem *little endian* ([10], strana 195). To znamená, že struktury jsou odesílány od nejméně významného bajtu po nejvýznamnější. Toto jsem bral v potaz při tvorbě datových struktur deskriptoru, kdy některé hodnoty mají velikost 2 či 4 bajty.

Grafické znázornění hierarchické struktury USB deskriptoru implementované ve firmwaru `FX2LP_audio_ISO` je na obrázku 6.2. Jednotlivé části jsou zde stručně popsány, přičemž názvy struktur byly volně přeloženy do češtiny. Deskriptor je v rámci kódu definován datovou strukturou v assembleru, jež je obsažena v souboru `descriptor.a51`.

Identifikátory dodavatele (VID) a produktu (PID) jsem neměl žádné k dispozici, proto jsem (provizorně) zvolil takové, které se dobře pamatují. Jako VID jsem zvolil hexadecimální hodnotu `cafe` a jako PID hodnotu `1337`. Nutno podotknout, že tyto zvolené identifikátory nelze legálně používat jinak než pro vlastní laboratorní účely. Identifikátory jsou definovány konstantami `VID` a `PID` v souboru deskriptoru a lze je zde změnit.

6.1.2 Popis činnosti

Při zahájení běhu firmwaru je nastavena sběrnice USB FIFO – synchronní režim slave, šířka datové sběrnice na 8 bitů, a frekvence výstupních hodin IFCLK na hodnotu 30 MHz. Řídicí signály a příznaky jsou aktivní v nule. Význam příznakových signálů je nastaven následovně: FLAGB indikuje plnou frontu EP6, FLAGC indikuje prázdnou frontu EP6. Příznaky FLAGA a FLAGD zůstávají nevyužity.

Dále je nastaven koncový bod EP6 IN (směrem do počítače) na isochronní přenos. Ostatní koncové body mikrořadiče jsou deaktivovány. Vnitřní vyrovnávací paměť (buffer) je nastavena na dvě úrovně po 1024 bajtech – jedna úroveň vyrovnávací paměti se plní, zatímco druhá je odesílána. Automatické odeslání paketu, pokud je naplněna fronta, je deaktivováno, protože odeslání paketu je úlohou FPGA. Probíhá zde také nulování vnitřní vyrovnávací paměti.

Plánoval jsem využít výstup PA0 pro signalizaci, že zvuková karta je právě používána. Detekce používání je implementována tak, že v obsluze řídicího příkazu přepnutí rozhraní do alternativní konfigurace 1 (s isochronním koncovým bodem) se přepne stav PA0 do stavu log. 0. Naopak, po přepnutí rozhraní do alternativní konfigurace 0 (bez datového toku), se přepne PA0 do log. 1. Tato implementace se neukázala být příliš spolehlivá, proto nakonec signál PA0 nebyl hradlovým polem využit.

Po počátečním nastavení čeká procesor ve smyčce na řídicí pakety (odesílané z počítače řídicím přenosem). Mimo obsluhy těchto paketů a jiných přerušení je procesor nečinný, přenos dat zajišťuje přímo hardware FX2LP. Knihovna `fx2lib` zajišťuje obsluhu potřebných přerušení. Například odesílá deskriptor, když si jej hostitelský počítač vyžádá řídicím příkazem `GET_DESCRIPTOR`.

Zápis do fronty koncového bodu EP6 IN probíhá prostřednictvím sběrnice USB FIFO. Tuto úlohu zastává hradlové pole, které zapíše vzorek do fronty,

jakmile jej přijme z mikrofonu (pokud není fronta plná). Každý vzorek se posílá jako 4 bajty (32bitová hodnota), nejméně významný bajt označuje pořadí vzorku. U rozhraní I²S se značí kanál (levý či pravý) a číslo rozhraní, u TDM číslo slotu a číslo rozhraní. Tento bajt však není ovladačem na počítači nijak interpretován (jeho obsah je ignorován).

6.1.3 Kompilace a nahrání firmwaru

Firmware je kompilován kompilátorem `sdcc` do binárního formátu Intel hex (`.ihx`). Tento binární soubor je potom na desku nahrán skrze USB kabel připojený k desce. Postup naprogramování čipu FX2LP je upřesněn v sekci 8.4. Aby byla kompilace a nahrání firmwaru jednoduchá činnost, je použit soubor `Makefile`, který je zpracován nástrojem `make`, když je zavolán v adresáři projektu s tímto souborem. Využívaná knihovna `fx2lib` je hledána v adresáři `fx2lib` umístěném vedle adresáře s tímto zdrojovým kódem. Knihovna se zkompiluje automaticky během první kompilace firmwaru.

Velikost zkompilovaného programu `FX2LP_audio_ISO` je podle informací získaných z vygenerovaného souboru `.mem` přibližně 2650 bajtů. Nároky programu na kapacitu paměti jsou tedy velice nízké.

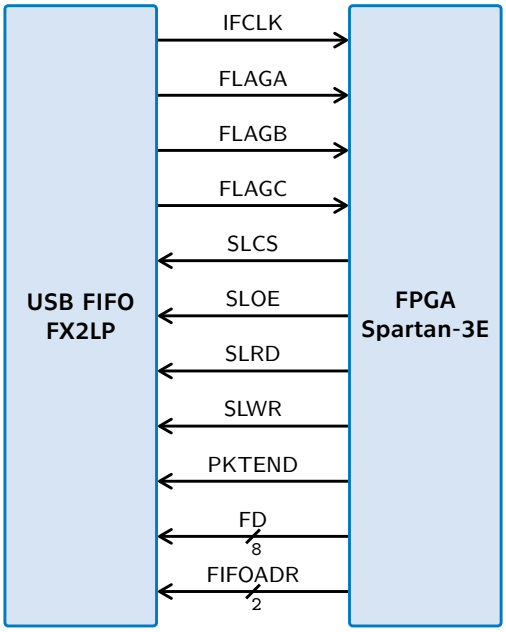
6.2 Paralelní sběrnice USB FIFO

Sběrnice USB FIFO má celou řadu signálů, jak lze vidět na blokovém schématu 6.1. Signál IFCLK (Interface Clock) je hodinový signál sběrnice, a v tomto případě je konfigurován tak, aby na něj čip FX2LP přivedl kmitočet 30 MHz. Příznaky FLAGA, FLAGB a FLAGC mají konfigurovatelný význam. V logickém návrhu využívám pouze příznak FLAGB, který indikuje plnou frontu FIFO pro koncový bod EP6.

Následuje 5 řídicích vodičů, všechny mají nastavenu aktivní úroveň v logické nule (u každého je v závorce uvedeno, jak je využit ze strany FPGA):

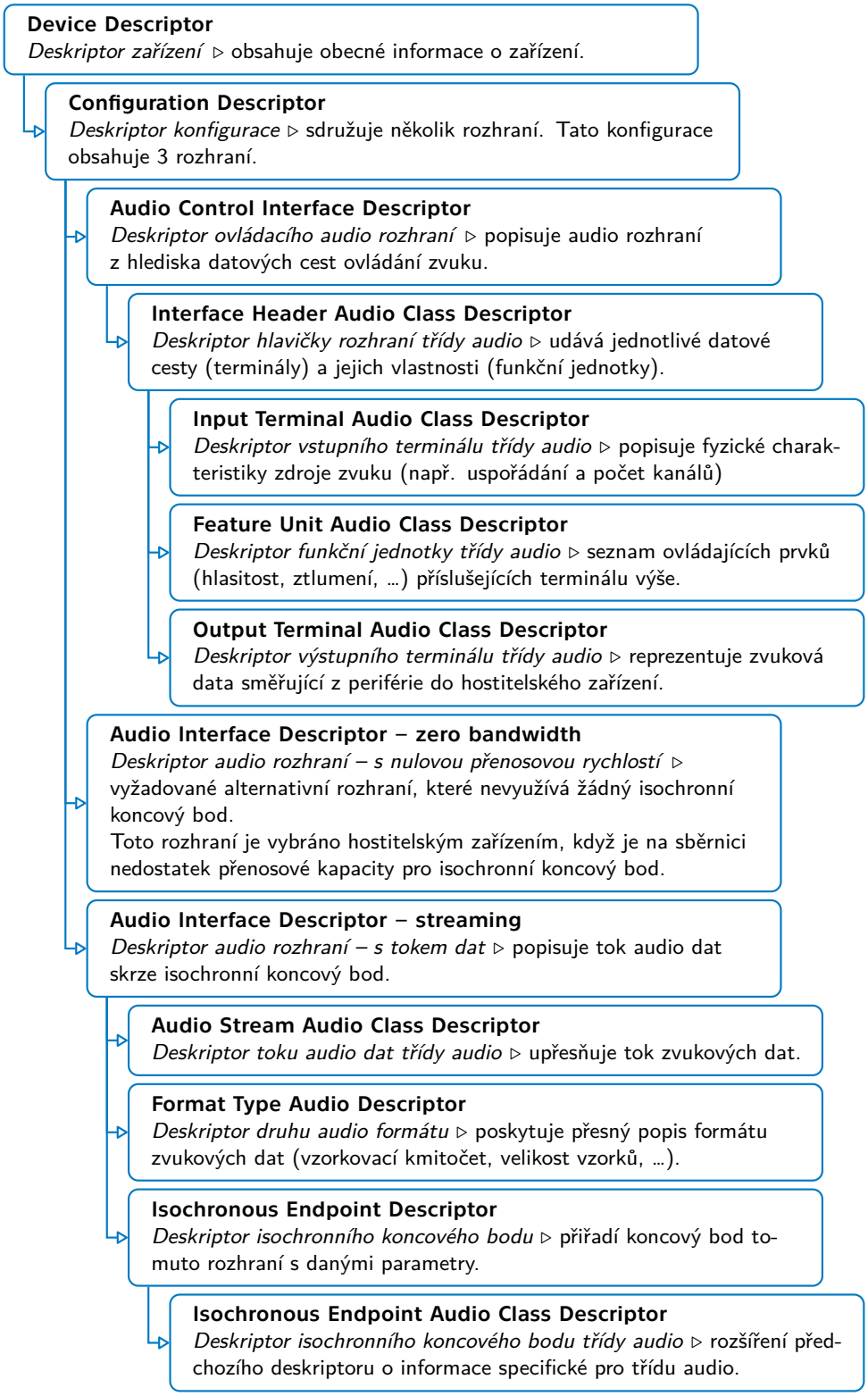
- SLCS (Slave Chip Select) – aktivuje sběrnici (trvale připojen k log. nule).
- SLOE (Slave Output Enable) – aktivuje se, pokud má být z FX2LP čteno (trvale deaktivován).
- SLRD (Slave Read) – pokud je aktivován, je každou periodu hodin čten bajt z FX2LP (trvale deaktivován).
- SLWR (Slave Write) – pokud je aktivován, je každou periodu hodin zapsán bajt do fronty FX2LP (aktivován podle potřeby).
- PKTEND (Packet End) – vynutí odeslání USB paketu, bez ohledu na jeho délku (aktivován na konci paketu).

Zbývá uvést osmivodičový signál FD (FIFO Data) – na něj se přivádí samotná data a dvouvodičovou adresu FIFOADR. Adresa FIFOADR slouží k přepínání mezi čtyřmi koncovými body, které obvod FX2LP podporuje. Protože využívám jen koncový bod EP6, je na tyto vodiče přivedena konstantní adresa `10`, jež odpovídá koncovému bodu EP6.



Obrázek 6.1: Blokové znázornění paralelní sběrnice USB FIFO

USB FIFO sběrnice pracuje v synchronním režimu, tzn. každou vzestupnou hranu hodin je zapsán jeden bajt dat FD do fronty, pokud je aktivní signál SLWR. Zápis jednoho vzorku, který má formu dvaatřicetibitové hodnoty, je v hradlovém poli řízen stavovým automatem jako sekvence osmibitových zápisů.



Obrázek 6.2: Struktura deskriptoru implementovaného USB zařízení [21]

Kapitola 7
Navržená deska plošných spojů

V rámci této práce jsem navrhl vlastní desku plošných spojů s FPGA Xilinx řady Spartan-3E. Návrh, nazvaný *FPGA Microphone Array Board* (zkráceně *MicArrayBoard*), byl vytvořen v programu KiCAD verze 7 a následně vyroben čínskou společností JLCPCB[1]. Program KiCAD byl zvolen z toho důvodu, že je dostupný zdarma bez omezení. Osazení součástek provedl David Vagner, který také navrhoval modulární mikrofonní pole, jehož účelem bylo připojení k této desce.

Náhled na 3D model desky pořízený v programu KiCAD je na obrázcích 7.1 (vrchní strana) a 7.2 (spodní strana). Fotografie osazené desky s osmi zapojenými konektory je potom na obrázku C.1 v příloze C. Schéma desky je rozděleno na čtyři listy formátu A4 a je součástí této práce v příloze D. Rozpiska součástek (kusovník) pro desku MicArrayBoard je v příloze E.

7.1 Obecný popis desky

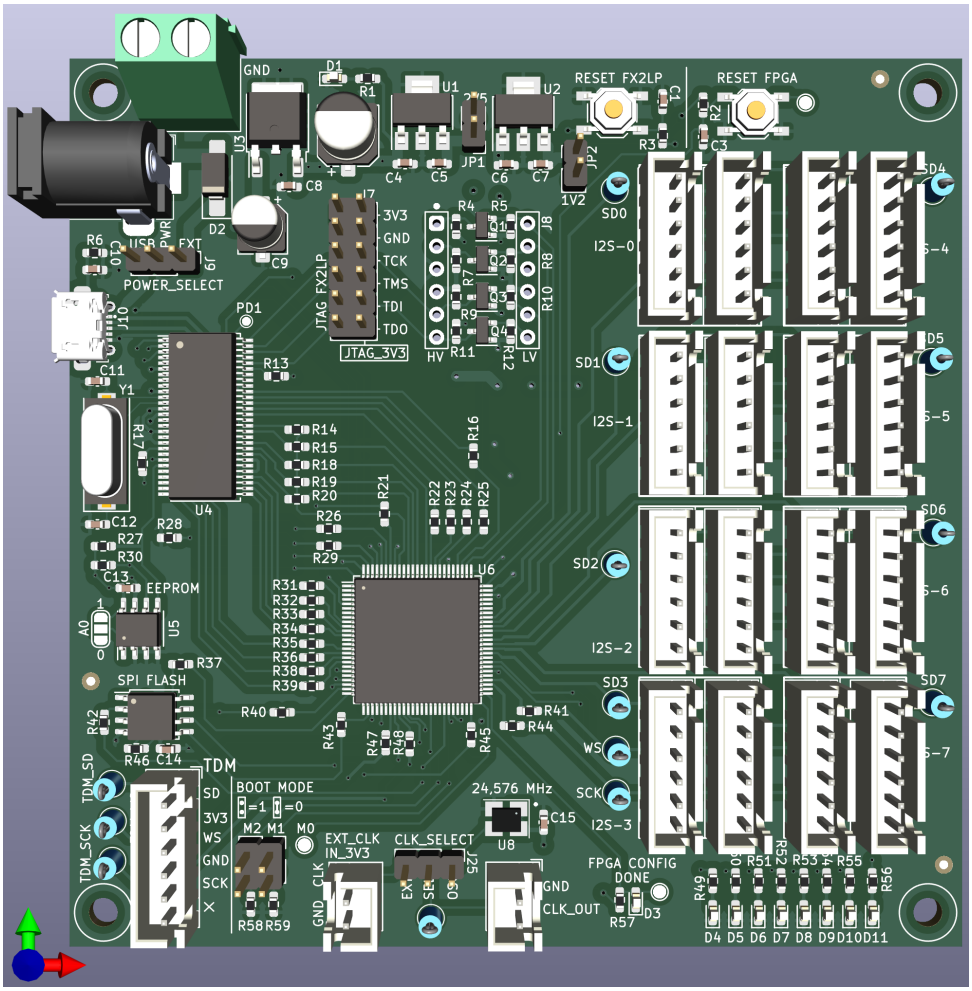
Na desce jsou využity čtyři vrstvy mědi, kdy prostřední dvě vrstvy slouží pro rozvod země a napájení. Rozměry desky jsou zaokrouhleně 99,5×99,1 mm. Tímto se plošný spoj vejde do mezního rozměru 100×100 mm, kdy je cena výroby ještě zvýhodněná. Celková tloušťka desky plošných spojů je 1,6 mm, tloušťka vnějších měděných vrstev je 35 µm a vnitřních 15,2 µm. Za povrchovou úpravu byla zvolena zelená nepájivá maska s bílými popisky. Pájecí plošky jsou pokoveny olovnatým cínem.

Většina součástek je v provedení pro povrchovou montáž – SMD (Surface Mount Device). Rezistory i keramické kondenzátory jsou velikosti 0603 v imperiálních jednotkách (plošný rozměr 1,6×0,8 mm), které lze pomocí vhodného vybavení (kterým není např. trafo páječka[2]) zapájet i ručně. Téměř všechny součástky jsou umístěny na vrchní straně desky, na spodní straně jsou pouze blokovací kondenzátory, které by bylo obtížné dávat na vrchní stranu, a tři rezistory.

Ve středu desky je umístěno hradlové pole Xilinx XC3S100E-4VQG100C v pouzdře VQFP (Very Small Quad Flat Package) se 100 vývody, které bylo

[1] `https://jlcpcb.com/`

[2] Z osobní zkušenosti není vhodná ani pájecí stanice „Pro'skit".

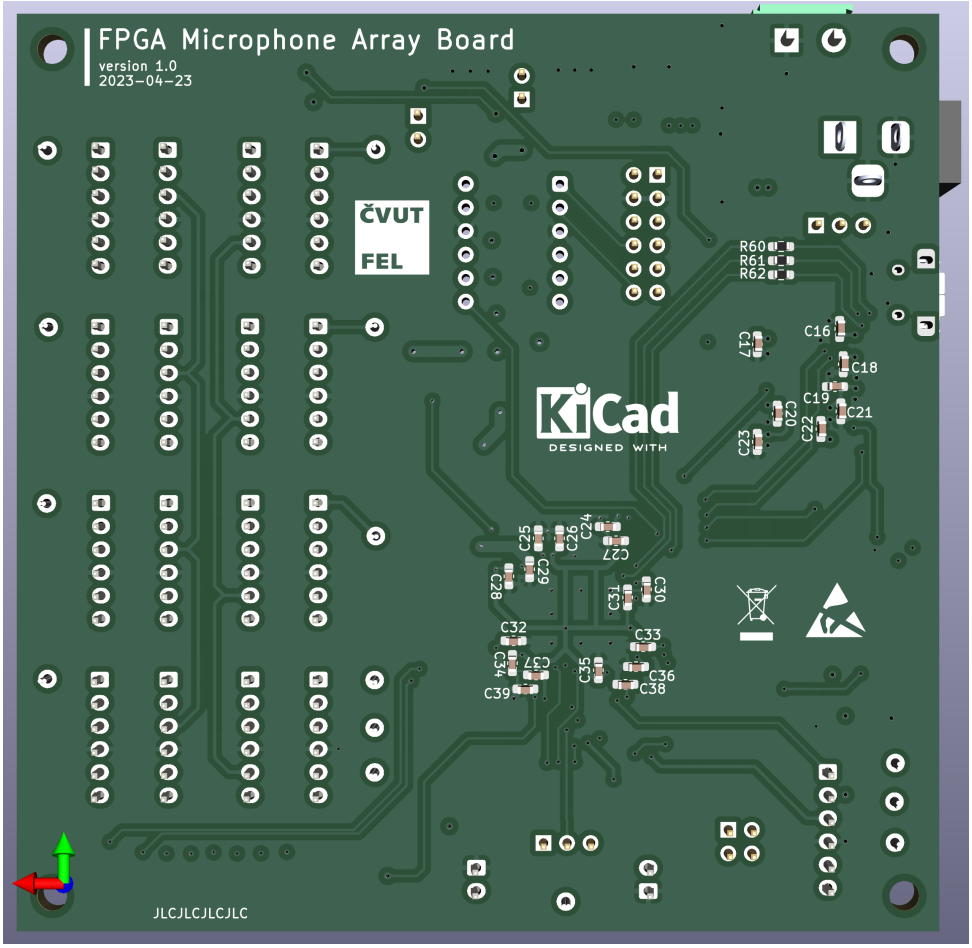


Obrázek 7.1: Náhled na 3D model desky z vrchní strany

zvoleno z toho důvodu, že v době navrhování to bylo jedno z ekonomicky nejdostupnějších hradlových polí Xilinx. To je doplněno USB mikrořadičem FX2LP, který zajišťuje tok zvukových dat do počítače. K připojení mikrofonního pole slouží konektory JST XH s šesti vodiči. V následujících sekcích bude každá část schématu podrobněji popsána.

7.2 Napájení desky

Pro napájení obvodů na desce byly z důvodu jednoduchosti zvoleny lineární napěťové regulátory. FPGA Spartan-3E vyžaduje tři různé úrovně napájecího napětí: 3,3 V, 2,5 V a 1,2 V. Datový list [17] doporučuje nejvýše 5% toleranci pro tyto hodnoty napětí. Dále je zde uvedeno, že nezáleží na pořadí, jakým zdroje těchto tří hodnot napětí po připojení napájení naběhnou, čehož jsem využil tím, že jsem regulátory napětí zapojil kaskádně a snížil tak výkonové ztráty. Ostatní obvody na desce si vystačí s nominálním napětím 3,3 V.



Obrázek 7.2: Náhled na 3D model desky ze spodní strany

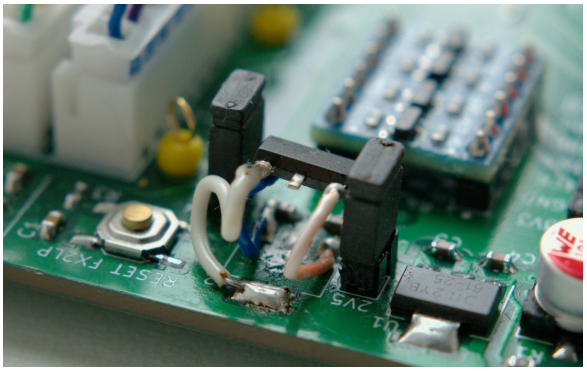
Navržená deska má 3 možné vstupy napájení:

- 5 V z mikroUSB konektoru, jenž zároveň slouží ke komunikaci,
- síťový zdroj se souosým válcovým konektorem (kladný střed),
- vlastní zdroj přivedený na svorkovnici.

Napájecí vstup z válcového konektoru a ze svorkovnice je opatřen schottkyho diodou jakožto ochranou proti přepólování.

7.2.1 Regulátory napětí

Zvolený 3,3V regulátor s nízkým úbytkem napětí (LDO – low-dropout) ROHM BD33FC0 dovoluje na svůj vstup připojit napětí 4,3 až 26,5 V, přičemž výstupní proudový odběr by neměl překročit 1 A [22]. Na vstupu a výstupu tohoto regulátory jsou umístěny elektrolytické kondenzátory o kapacitě 22 µF respektive 100 µF, které omezují kolísání napětí na vstupu a výstupu. Tyto hodnoty kapacity byly zvoleny podle grafu doporučených hodnot v datovém listu regulátoru [22]. Na výstup tohoto regulátoru je připojena svítivá dioda v sérii s 1kΩ rezistorem, která indikuje, zda je deska



Obrázek 7.3: Regulátor napětí 1,2 V připájený na drátcích

napájena.

Napětí 2,5 V je následně sníženo z 3,3 V LDO regulátorem AP7361C-25E. Na vstupní a výstupní větvi jsou podle doporučeného zapojení [23] keramické kondenzátory o kapacitě 4,7 µF. Obdobně je zapojen regulátor AP7361-12E s výstupním napětím 1,2 V, který bere vstup z regulátoru 2,5 V. Na výstupech těchto regulátorů jsou pro účel měření výstupního proudu (či napětí) umístěny odnímatelné propojky na kolíkové liště.

Při prvním přivedení napájení na čerstvě spájenou desku se objevil problém, že regulátor napětí 1,2 V nedodával na výstup 1,2 V, ale pouze několik stovek milivoltů. Navíc se obvod začal poněkud rychle zahřívat. Původní podezření bylo, že na desce je zkrat. Avšak multimetr v režimu měření vodivosti tuto domněnku vyvrátil. Následoval pokus regulátor vyměnit za další, což nepřineslo žádné zlepšení.

Při dalším zkoumání datového listu regulátoru [23] jsem si všiml, že daný typ regulátoru má dvě různé varianty rozmístění vývodů v tom samém pouzdře – SOT223. Mou nepozorností byla objednána varianta pouzdra „R“ (AP7361-12ER), který má namísto země výstup (na chladiči pouzdra) a namísto vstupního vývodu zem. Tímto vzniklo reverzní zapojení čipu a to způsobilo nadměrný odběr proudu a zahřívání.

Aby se nemusela kvůli regulátoru dělat další objednávka, byl dočasně připájen na desku otočen a volně visící na drátcích s pevným měděným jádrem, jak je ukázáno na obrázku 7.3. Později byl zakoupen a osazen regulátor AP2114HA-1.2TRG1 (pro účel použití je zaměnitelný s AP7361), který má již rozmístěny vývody tak, jak bylo původně zamýšleno.

7.3 Integrovaný obvod FX2LP

Zapojení integrovaného obvodu FX2LP na desce bylo inspirováno tím na vývojové desce Digilent Nexys 3 [24]. Čip je napájen napětím 3,3 V, na každém napájecím vývodu má umístěn blokovací keramický kondenzátor o kapacitě 100 nF. Zdrojem hodin čipu je externí krystalový rezonátor o kmitočtu 24 MHz. Na každém z obou vývodů jsou zapojeny keramické zatěžovací kondenzátory o kapacitě 22 pF.

Na nulovacím vstupu RESET je umístěn sériový RC článek, k jehož kondenzátoru je paralelně připojeno tlačítko RESET FX2LP. Stiskem tohoto tlačítka se čip FX2LP dostane do počátečního stavu, kdy se pokouší znovu načíst program z externí paměti EEPROM.

Většina vývodů čipu přísluší rozhraní USB FIFO, ty jsou všechny připojeny k hradlovému poli. Některé další vývody (PD5, PD6, PD7, PA0, PA1 a PA3), jejichž význam lze určit firmwarem procesoru, jsou k FPGA také přivedeny. Propojit vývody FX2LP a FPGA byla při návrhu desky největší výzva, neboť bylo zapotřebí se vyhnout některým spojům v cestě napříč, přičemž jsem se snažil omezit počet přechodů mezi vrchní a spodní stranou plošného spoje průchodkami.

Po vzoru vývojové desky Nexys 3 jsem na každý vývod mezi FPGA a čipem FX2LP umístil rezistor o velikosti 75 Ω. Domnívám se, že zde byly umístěny za účelem omezení odrazů vysokofrekvenčních signálů, které jsou zdrojem rušení.

Čtyři vývody čipu jsou využity pro emulaci rozhraní JTAG, přes které je konfigurováno FPGA. Jejich přiřazení odpovídá tomu na desce Nexys 3 – zvolil jsem tak za účelem kompatibility. Přiřazení signálů JTAG k vývodům mikrořadiče je v tabulce 7.1.

Tabulka 7.1: Přiřazení vývodů JTAG portu na mikrořadiči FX2LP

FX2LP	JTAG
PD0	TDO
PD2	TDI
PD3	TMS
PD4	TCK

7.3.1 Vedení datových vodičů USB

Datové vodiče rozhraní USB (DATA+ a DATA-) vyžadovaly zvláštní pozornost, jelikož bylo zapotřebí je vést jako diferenční pár o definované impedanci. Specifikovaná impedance je 45 Ω mezi datovým vodičem a zemí a 90 Ω mezi oběma datovými vodiči. Na plošném spoji lze tento požadavek realizovat vytvořením dvojitého mikropáskového vedení [25].

Parametry mikropásku jsem zjistil použitím „*nástrojů kalkulačky*“ v programu KiCAD, přičemž bylo využito parametrů čtyřvrstvé desky JLCPCB. V návrhu má tento diferenční pár vlastní třídu spojů USB data, kde je zadána vypočtená šířka mikropásku 0,42 mm (parametr DP šířka) a vzdálenost mezi oběma mikropásky 0,5 mm (parametr Mezera DP). Realizované vedení datových vodičů USB na plošném spoji je z velké části obklopeno zemní plochou, aby se zamezilo rušení od okolních signálů.

7.3.2 Paměť EEPROM s rozhraním I²C

Obvod FX2LP má uložen firmware v externí paměti EEPROM, jež je k čipu připojena skrze sériové rozhraní I²C (Inter-Integrated Circuit)[3]. Jsou využity dva vývody vyhrazené právě pro tuto funkci: SDA (Serial Data) a SCL (Serial Clock). Na každém z obou vodičů I²C je upínací rezistor na napájecí napětí o odporu 4,7 kΩ. Paměť má také vlastní blokovací 100nF kondenzátor.

Použitý typ paměti EEPROM má tři vývody určující 3 nejnižší bity sedmibitové adresy, která této paměti na sběrnici I²C přísluší. Vývody A2 a A1 jsou trvale uzemněny, odpovídající bity adresy mají tudíž hodnotu 0. Dle referenčního manuálu čipu FX2LP [14] se má nastavit adresový vodič A0 podle toho, jakou šířku interní adresy použitá paměť očekává, což se odvíjí od její kapacity. Pro osmibitovou adresu má být A0 připojen na logickou nulu, pro šestnáctibitovou interní adresu má být připojen na log. 1. Z tohoto důvodu jsem na desku umístil přepájitelnou propojku, kterou lze přemostit vývod A0 buď na napájení (log. 1) nebo na zem (log. 0).

Na desce jsem použil paměť M24128 výrobce STMicroelectronics, jež má kapacitu 16 kB a vyžaduje šestnáctibitovou interní adresu. Propojka má v tomto případě být připájena na napájení. Kdyby však z nějakého důvodu chtěl uživatel zabránit načítání firmwaru z této paměti, stačí přepájet propojku na opačnou stranu, čímž se načtení zamezí[4].

7.4 Hradlové pole Spartan-3E

Hradlové pole Xilinx XC3S100E-4VQG100C jsem zapojoval podle informací uvedených v datovém listu pro rodinu hradlových polí Spartan-3E [17]. Dbal jsem na to, aby se na desce spoje co nejméně křížily, takže toto zapojení prošlo několika revizemi před tím, než dosáhlo finální podoby. Využil jsem toho, že u většiny uživatelských vývodů lze zvolit, zda vývod použít jako vstup či výstup.

FPGA je umístěno poblíž středu desky. Je propojeno s mikrořadičem FX2LP a konektory na mikrofony. Pomocí FPGA lze také řídit stav osmi indikačních LED diod, každá je připojena anodou skrze 470Ω rezistor na jeden vývod hradlového pole; katoda je uzemněna.

7.4.1 Napájení hradlového pole

Hradlové pole Spartan-3E má několik oddělených bloků se svým vlastním napájením – čtyři nezávislé banky vývodů VCCO, napájení vnitřního jádra VCCINT a napájení pomocných obvodů VCCAUX. Všechny banky vývodů jsou napájeny napětím 3,3 V. Vnitřní jádro vyžaduje napětí 1,2 V a pomocné obvody 2,5 V. U každého napájecího vývodu jsem umístil blokovací 47nF keramický kondenzátor, který má za úkol lokálně vyhlazovat kolísání napětí

[3] Toto rozhraní, přestože má podobný název, nemá mnoho společného s rozhraním I²S

[4] Firmware je v EEPROM uložen se záhlavím, jehož čtení selže, pokud procesor FX2LP odešle pouze 8bitovou interní adresu namísto 16bitové.

z důvodu špičkového odběru proudu při činnosti logického obvodu (zejména překlápění hradel).

7.4.2 Konfigurační rozhraní hradlového pole

Některé vývody použitého hradlového pole jsou vyhrazeny pro načítání bitové konfigurace a nastavení chování při tomto procesu. Ve schématu (příloha D, 3. list na straně 84) jsem tyto vývody oddělil do samostatného bloku. Význam a použití těchto vývodů je popsán v již zmíněném dokumentu [17].

Trojice konfiguračních vývodů VS2, VS1 a VS0 (Variant Select) vybírá variantu SPI paměti – nastavuje se, jaký čtecí příkaz připojená paměť podporuje. Vybraná sériová SPI paměť podporuje příkaz „FAST READ“, což odpovídá kombinaci VS2 = VS1 = VS0 = 1. Tyto vývody nejsou na desce nikam připojeny, během konfigurace jsou totiž interně připojeny k upínacímu rezistoru na napájení.

Upnutí konfiguračních vývodů k napájení v době, kdy FPGA načítá konfiguraci, je v návrhu pevně nastaveno připojením vývodu HSWAP přes 470Ω rezistor na zem.

Vývody M[2:0][5] vybírají zdroj, ze kterého FPGA načítá bitovou konfiguraci. Vývod M0 je ponechán plovoucí, vyveden je pouze na testovací bod. Během načítání konfigurace jsou, obdobně jako vývody VS[2:0], i tyto vývody upnuty na úroveň logické 1. Nastavitelnost je pro vývody M2 a M1 řešena možností upnout vybranou množinu vývodů přes 470Ω rezistor k zemi pomocí propojek nasunutých na kolíkovou lištu.

Deska MicArrayBoard byla navržena se dvěma možnými zdroji konfigurace hradlového pole: buďto sběrnice JTAG, nebo paměť typu flash s rozhraním SPI. Tomu odpovídají stavy vývodů M[2:0] dané následující tabulkou:

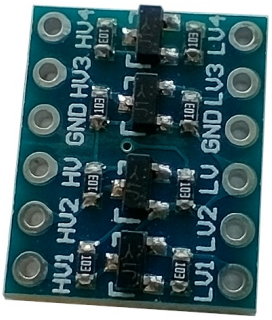
Tabulka 7.2: Výběr zdroje konfigurace hradlového pole pomocí vývodů M[2:0]

Zdroj konfigurace	M2	M1	M0
JTAG	1	0	1
SPI flash	0	0	1

Výstup DONE je typu otevřený kolektor/drain. Logická 1 na tomto výstupu signalizuje, že načítání konfigurace FPGA bylo dokončeno. Na DPS MicArrayBoard je k tomuto výstupu připojen upínací rezistor 470 Ω na hladinu 2,5 V a svítivá dioda na zem. Pokud není načtena konfigurace, LED dioda nesvítí, jelikož vývod DONE je stažen k zemi. Po dokončení načítání konfigurace je výstup DONE plovoucí, a dioda tudíž svítí.

Vstup PROG_B je nulovací signál – aktivní je v logické nule. Přivedením na krátkou dobu na nízkou úroveň se vymaže načtená konfigurace FPGA. Na desce je vstup PROG_B připojen k tlačítku FPGA RESET, jehož stisknutím se provede nulování hradlového pole. Vývody DOUT a INIT_B neplní na

[5] Označena je tak množina vývodů M2, M1 a M0.



Obrázek 7.4: Modul obousměrného převodníku logických úrovní

desce žádnou užitečnou funkci. První z nich zůstal nezapojen, druhý je upnut k 3,3V napájení podle vzoru v datovém listu.

7.4.3 Konfigurační JTAG port

Vývody JTAG pracují na čipu Spartan-3E s logickými úrovněmi CMOS 2,5 V. Je-li nezbytné připojení JTAG programátorů, pracujících na 3,3 V, navrhuje výrobce využít sériově zapojených rezistorů omezujících proud (pod 10 mA) [26]. Tím se „využijí“ vnitřní ochranné diody na vývodech uvnitř čipu, navíc je vzniklým reverzním proudem zatěžován regulátor 2,5 V. Takovéto řešení mi připadá spíše jako improvizace, je tu totiž riziko zničení některé části vnitřní struktury čipu.

Namísto řešení navrhovaného výrobcem jsem zvolil převodník napěťových úrovní. Každý signál JTAG je převeden obousměrným převodníkem úrovní, jenž je realizován zapojením s jedním tranzistorem MOSFET. Lze zvolit i již hotový modul a zapájet jej do připravených otvorů (pomocí kolíkových lišt s roztečí 2,54 mm), jeden takový modul je vyfocen na obrázku 7.4.

Počítal jsem i s možností připojení externího JTAG programátoru, proto byly JTAG signály (spolu s napájením 3,3 V a zemí) vyvedeny na kolíkovou lištu. Kolíkovou lištu lze propojkami přemostit k vývodům USB mikrořadiče FX2LP (na sousední řadě kolíků), vhodným firmwarem lze potom nahrát konfiguraci FPGA přes USB bez připojování externího programátoru.

7.4.4 Paměť SPI flash

Pro trvalé uchování bitové konfigurace FPGA slouží na desce MicArrayBoard sériová paměť typu flash – Winbond W25Q32BV[6] [27] v osmivývodovém pouzdře SOIC-8 (široká varianta[7]). Tato paměť má kapacitu 32 Mb (4 MB) a je připojena k FPGA pomocí rozhraní SPI (Serial Peripheral Interface). Tuto konkrétní paměť jsem zvolil z toho důvodu, že je na seznamu podporovaných sériových pamětí v nahrávacím nástroji iMPACT, jenž je součástí Xilinx ISE.

[6] Varianta W25Q32BV se již dále nevyrábí, nahradila ji varianta W25Q32JV. Na desku MicArrayBoard byla však osazena paměť W25Q32BV, kterou jsem měl k dispozici.

[7] Standardní šířka pouzdra SOIC se udává jako 150 mil (3,8 mm), širší varianta má udávanou šířku 208 mil (5,3 mm).

Paměť W25Q32BV je napájena napětím 3,3 V, poblíž je umístěn blokovací 100nF keramický kondenzátor.

Propojení s FPGA je provedeno čtyřmi standardními signály SPI:

- MISO (Master Input-Slave Output),
- MOSI (Master Output-Slave Input),
- CLK (Clock),
- $\overline{\text{CS}}$ (Chip Select).

Řídicí vstupy $\overline{\text{WP}}$ (Write Protect) a $\overline{\text{HOLD}}$ nejsou využívány, na plošném spoji jsou upnuty rezistory na napájení. Na každém signálovém vodiči rozhraní SPI je umístěn sériový 22Ω rezistor pro případné utlumení strmých náběžných hran, které by mohly rušit okolní vodiče.

Co se týče velikosti bitové konfigurace, ta je pevně dána konkrétním modelem FPGA. Pro model XC3S100E je tato velikost 581,344 bitů [17], není tedy žádný problém s nedostatkem kapacity flash paměti.

7.5 Mikrofony

K připojení mikrofonů jsou na desce umístěny šestivodičové konektory JST XH (typ samec) s roztečí 2,5 mm. Jako protikus je na plochém svazku vodičů každého mikrofonu konektor typu samice, který byl manuálně nakrimpován pro tento typ konektoru určenými krimpovacími kleštěmi. Výhodou tohoto typu konektoru je, že díky mechanickému zajištění nedochází k nahodilému odpojení.

Rozmístění signálů na konektoru bylo navrženo tak, aby odpovídalo tomu na modulárním mikrofonním poli navrženém Ing. Davidem Vagnerem. Na konektorech je využito pouze 5 vodičů z 6, poslední nakonec využit nebyl. Šestivodičové konektory však již byly objednány, proto zůstaly i v návrhu.

Pro I²S mikrofony je zde 16 konektorů, z nichž každá dvojice představuje jednu I²S sběrnici – levý a pravý kanál. Těchto 8 dvojic konektorů je na desce označených pomocí indexovaných názvů: I2S-0 až I2S-7. Signály na konektorech jsou přímo připojeny na vývody FPGA, všechny signály operují na úrovních 3,3V logiky CMOS. Pro napájení mikrofonů slouží vodič s napětím 3,3 V a zemní vodič.

Hodinový signál SCK a signál WS jsou společné pro všech 8 rozhraní – tím je docíleno synchronního čtení mikrofonů (zpoždění signálů na přívodních vodičích lze zanedbat). Samostatně od ostatních potom stojí jeden konektor pro TDM rozhraní, jež má přiřazeno vlastní vývody na FPGA.

Každý signál I²S a TDM vede na testovací bod, kam lze zapájet drátovou smyčku, do které lze zaháknout sondu osciloskopu. Opomenul jsem však přidání testovacího bodu na zemní potenciál, takže zemní vodič sondy je odkázán na improvizované připojení, například k zemním kolíkům konektoru JTAG_3V3.

Pro účely vzorkování audio signálů je na DPS osazen oscilátor o nominálním kmitočtu 24,576 MHz. Tento kmitočet byl zvolen z toho důvodu, že je celočíselným násobkem běžně používané vzorkovací frekvence 48 kHz. Výstup osci-

látoru je přiveden na vstup hradlového pole označený IO_L06N_2/D1/GCLK3. Vývody označené funkcí GCLK jsou připojeny na vstup vnitřního budiče globálního hodinového signálu. V FPGA potom lze takový kmitočet vydělit pomocí čítače na potřebný celočíselný podíl.

Do FPGA lze přivést i externí hodinový signál namísto hodin z oscilátoru. Měl by dosahovat logické úrovně 3,3 V, jelikož je to napájecí napětí zvolené pro všechny banky vývodů tohoto hradlového pole. Pokud je však kmitočet externího signálu rozdílný oproti tomu z oscilátoru, musí se zohlednit v číslicovém návrhu na FPGA. Zdroj hodinového signálu se vybírá zkratovací propojkou na kolíkovém konektoru CLK_SELECT.

7.6 Poznámky k osazení součástkami

Jak uvádí referenční manuál obvodu FX2LP ([14], strana 193), vývod A0 paměti EEPROM má mít logickou úroveň 1, pokud používá interní adresu o délce dvou bajtů. Vodič A0 je připojen na desce k pájitelné propojce, jejímž přemostěním k napájení či zemi se zvolí hodnota tohoto vývodu paměti. Pro zvolený model EEPROM 24LC128 je tedy vyžadováno připájení propojky do polohy „1“.

Programovací port JTAG hradlového pole vyžaduje logické úrovně CMOS 2,5 V. Aby bylo možné využít často využívané úrovně 3,3 V, byl přidán převodník logických úrovní pro čtyři signály. Tento převodník lze osadit vlastními tranzistory MOSFET a rezistory, nebo je možné připájet již hotový modul, který se prodává například na internetovém tržišti Aliexpress.

Kapitola 8

Programové vybavení pro desku FPGA MicArrayBoard

8.1 Výběr nástroje sloužícího k naprogramování čipu FX2LP

Volně dostupný nástroj `fxload` (ten je dostupný jako balíček ve většině linuxových distribucí) vyžaduje k naprogramování připojené I²C EEPROM paměti zavaděč druhého stupně (*second stage loader*), který je nejdříve načten do interní paměti RAM. Takovýto speciální zavaděč, který podporuje USB příkaz dodavatele (*vendor command*) `0xA2`, však není s nástrojem `fxload` distribuován, pravděpodobně z důvodu licencování (dodával jej výrobce čipu FX2LP Cypress ve svém vývojovém prostředí) [28].

Pokusil jsem se pro tento účel použít ukázkový firmware „eeprom“ v knihovně `fx2lib`, ale bez úspěchu. Kvůli těmto neduhům jsem místo nástroje `fxload` hledal alternativní řešení. Jako alternativu jsem nalezl nástroj `fx2loader` (se svobodnou licencí), který měl být schopen zapsat firmware do EEPROM. Oficiální repozitář na Githubu[1] však byl archivován v roce 2020, což znamená, že autor tohoto nástroje, Chris McClelland[2], jej přestal vyvíjet. Odkaz na autorův web[3] je již také neplatný.

Na autorově profilu na Githubu jsem však našel repozitář `libfpgalink`[4], který je novější a obsahuje nástroj `flcli`, který nejen dokáže programovat EEPROM procesoru FX2LP, ale dokonce i konfigurovat FPGA přes sběrnici JTAG (Joint Test Action Group). Knihovna `libfpgalink` je zveřejněna pod licencí LGPL verze 3. Knihovnu jsem nakonec použil i pro vytvoření několika pomocných skriptů v jazyce Python pro nahrávání desky. Použití těchto nástrojů bude v následujících sekcích popsáno.

[1] `https://github.com/makestuff/fx2loader`
[2] `https://github.com/makestuff`
[3] `http://www.makestuff.eu/wordpress/?page_id=343`
[4] `https://github.com/makestuff/libfpgalink`

8.2 Uvolnění USB zařízení obsazeného ovladačem

Když na mikrořadiči byl právě nahrán firmware tvářící se jako zvuková karta, tak se při pokusu o nahrání nového firmwaru do EEPROM objevil problém – skript selhal s chybou `LIBUSB_ERROR_BUSY`. Významem takovéto chyby je to, že USB zařízení je aktuálně používáno jiným procesem (a pravděpodobně je k němu přiřazen ovladač).

Tento problém jsem vyřešil pomocí `libusb_detach_kernel_driver`, což je funkce knihovny `libusb` k odebrání ovladače. Tuto funkci volám v knihovně `libusbwrap` při otevírání zařízení, pokud je právě přiřazen ovladač žádanému rozhraní USB zařízení. Program, který volá tuto funkci musí mít oprávnění zapisovat do zařízení, jinak by k odebrání ovladače nedošlo. Z tohoto důvodu by měl uživatel nastavit `udev` pravidla pro veškeré kombinace identifikátorů VID a PID, kterých deska může nabývat. Řešení není efektivní, pokud má USB zařízení vícero rozhraní, kterým je přiřazen ovladač, což však není případ desky MicArrayBoard.

Další, identická chyba, se vyskytla o několik řádků níže při nastavování konfigurace zařízení. Pokud má totiž ovladač přiděleno více než jedno rozhraní, řešení popsané výše neodebere ovladač z jiných rozhraní než z toho požadovaného. To je nedostatek tohoto jednoduchého řešení.

Ve firmwaru `FX2LP_audio_ISO` využívám pouze jednu konfiguraci (konfigurace číslo 1). Rozhodl jsem se tudíž této chybě předejít tak, že knihovna přepíná konfiguraci pouze tehdy,je-li rozdílná od té stávající. Pokud je konfigurace zařízení předem známa, lze se takto vyhnout chybě `LIBUSB_ERROR_BUSY`.

Provedené změny v knihovně `libusbwrap` jsou na Gitlabu[5] a jsou zahrnuty i při sestavování knihovny `libfpgalink` postupem, který bude dále popsán.

8.3 Kompilace knihovny `libfpgalink`

Původně nebylo úplně zřejmé, jak zdrojový kód knihovny `libfpgalink` zkompilovat. Zdrojové soubory totiž odkazují na externí knihovny mimo repozitář, navíc instrukce v souboru `README` již nejsou platné, protože systém kompilace byl změněn a nově využívá program `cmake`. Po několika pokusech jsem nakonec došel k funkčnímu řešení, které jsem umístil na Gitlab[6]. Klíčem bylo vytvoření hierarchicky nadřazeného souboru `CMakeLists.txt`, jenž zahrnuje veškeré knihovny a definice, na kterých kompilace závisí.

8.3.1 Potřebné programové vybavení

Knihovna `libfpgalink`, jejíž součástí je nástroj `flcli`, má kompilaci řízenou pomocí programů `cmake` a `make`, jež jsou vyžadovány. Dále je zapotřebí kompilátor `gcc`. K naklonování (stažení) repozitáře slouží systém pro správu

[5] `https://gitlab.com/micarray-fpga/fx2lp-tools/makestuff/libusbwrap`
[6] `https://gitlab.com/micarray-fpga/fx2lp-tools/fpgalink-build`

verzí `git`. Z knihoven je vyžadována knihovna `libusb` verze 1.0 (společně s hlavičkovými soubory[7]). Vlastní knihovny (v repozitáři) se nainstalují spolu s nástrojem po zkompilování.

Na linuxových distribucích založených na Debianu a Ubuntu lze potřebný software stáhnout a nainstalovat správcem balíčků `apt`. Před samotnou instalací je vhodné aktualizovat databázi balíčků. Aktualizace databáze balíčků a jejich instalace se provede pomocí následujících příkazů:

```
sudo apt update
sudo apt install build-essential cmake libusb-1.0-0-dev git
```

8.3.2 Postup kompilace

1. Klonování „obalovacího“ repozitáře, který má vloženy závislosti formou takzvaných podmodulů (submodules) v rámci nástroje pro správu verzí `git`. Důležitý je zde parametr `--recurse-submodules`, který zvolí tzv. rekurzivní klonování. To znamená, že se naklonují i adresáře vložené jako podmoduly, které nejsou součástí tohoto repozitáře, ale odkazují na jiný repozitář.

   ```
   git clone --recurse-submodules \
   https://gitlab.com/micarray-fpga/fx2lp-tools/fpgalink-build.git
   ```

2. Vstoupení do právě naklonovaného adresáře a vytvoření složky pro zkompilované soubory. Následně se nakonfiguruje a spustí kompilace nástrojem `cmake` v podadresáři `build`.

   ```
   cd fpgalink-build
   mkdir build
   cmake -B build
   cmake --build build
   ```

3. Instalace knihovny `libfpgalink`, nástroje `flcli` a příslušných knihoven. Příkaz `ldconfig` vyhledá nově nainstalované knihovny, takže budou viditelné pro spouštěné programy.

   ```
   sudo cmake --build build -- install
   sudo ldconfig
   ```

8.4 Programování čipu FX2LP

Pokud je výše uvedeným postupem nainstalována knihovna `libfpgalink` obsahující nástroj `flcli`, jejíž součástí je knihovna `libfx2loader`, je možné tyto využít k nahrávání programu na mikrořadič FX2LP.

Přístup k USB zařízení běžně vyžaduje superuživatelská oprávnění, proto se příkazy volají s privilegovaným přístupem pomocí předsazeného příkazu `sudo`. Oprávnění pro běžného uživatele lze zařídit vytvořením konfigurace správce zařízení `udev`, což je popsáno v sekci 9.2.

[7] U distribucí založených na Debianu/Ubuntu končí název balíčku knihovny s hlavičkovými soubory příponou `-dev`.

8.4.1 Nahrávání firmwaru do paměti RAM

Pro (dočasné) načtení programu mikrořadiče do interní paměti RAM se použije nástroj `fx2cli`, jenž je součástí knihovny `libfx2loader`, která se instaluje společně s `libfpgalink`.

K tomuto účelu není vhodný nástroj `flcli`, jelikož ten po nahrání očekává, že firmware bude implementovat řídicí příkazy specifické pro interní použití knihovnou `libfpgalink`. Přestože se program načte do RAM, `flcli` selže s chybou:

```
flOpen(): getStatus(): usbControlRead(): LIBUSB_ERROR_PIPE
```

Takové chování není vhodné, chceme-li použít nástroj ve skriptech, které automatizují nahrávání, neboť výskyt jakékoli chyby (a nenulové návratové hodnoty) signalizuje selhání příkazu.

Uvažujme, že pracovní adresář v aktuálním terminálovém okně je složka programu `FX2LP_audio_ISO`. Potom následující příkaz nahraje zkompilovaný program ve formátu Intel hex do RAM mikrořadiče:

```
sudo fx2cli --vidpid 04b4:8613 build/FX2LP_audio.ihx ram
```

8.4.2 Nahrávání programu mikrořadiče do paměti I²C EEPROM

Nahrání programu mikrořadiče FX2LP na externí paměť EEPROM se provede nástrojem `flcli` spolu s přepínačem `--eeprom` a s příslušným binárním souborem pro mikrořadič. Přepínač `--ivp` zadává původní ID dodavatele a ID produktu (VID:PID) – tedy takové, kterými se čip FX2LP ohlásí počítači po připojení. Další přepínač `--vp` potom udává cílovou hodnotu VID:PID. Zde se jedná o VID:PID zavaděče druhého stupně. Firmware zavaděče druhého stupně je integrován do knihovny, což usnadňuje práci s tímto mikrořadičem.

Vzorový příkaz, jenž nahraje zkompilovaný program `FX2LP_audio_ISO` do paměti EEPROM, vypadá následovně:

```
sudo flcli --ivp 04b4:8613 --vp 1d50:602b --eeprom build/FX2LP_audio.ihx
```

Pokud je připojena EEPROM se správně nastavenou I²C adresou, tak po resetu mikrořadiče se z ní načte program `FX2LP_audio_ISO` do programové paměti.

8.4.3 Mazání paměti I²C EEPROM

Během experimentování s deskou MicArrayBoard jsem narazil na potřebu obsah paměti EEPROM připojené k mikrořadiči vymazat. Tuto možnost nástroj `flcli` nenabízí. Součástí knihovny `libfpgalink` je soubor `fl.py`, jenž definuje vazbu knihovny `libfpgalink` na jazyk Python. Rozhodl jsem se tak vytvořit skript v jazyce Python, který by umožnil vymazat paměť EEPROM. Po investigaci jsem zjistil, že ani knihovna `libfpgalink` takovou funkcionalitu neimplementuje. Implementuje pouze možnost nahrát firmware, tímto se však do paměti zapíše i binární hlavička, kterou upravit nelze.

Zápis libovolných binárních dat do EEPROM je implementován v knihovně

`fx2loader`, kterou `libfpgalink` využívá. Jelikož však tato knihovna nemá vytvořené vazby na jazyk Python, bylo nutné je vytvořit. Tímto jsem se seznámil s knihovnou `ctypes`, která vytváří vazby na knihovny napsané v jazyce C. Vazby byly vytvořeny jen pro využívané funkce. Současně byly na Python navázány funkce knihovny `usbwrap`, která pomáhá navázat komunikaci s USB zařízeními, konkrétně zde s mikrořadičem FX2LP.

Mazání paměti EEPROM jsem tímto způsobem implementoval ve skriptu `fx2erase.py`, jenž je pod licencí GNU LGPL v3 zveřejněn na Gitlabu[8]. Povinnými argumenty jsou identifikátor dodavatele a identifikátor produktu, které aktuálně deska používá. Vymazání EEPROM spočívá v zápisu souvislého bloku nulových bajtů. Velikost tohoto bloku lze změnit parametrem `--size`, výchozí hodnota je 2048 bajtů. Příklad zavolání skriptu:

```
sudo python3 fx2erase.py -v 04b4 -p 8613 --size 4096
```

8.5 Nahrávání konfigurace FPGA přes sběrnici JTAG

Otázka nahrávání mikrořadiče FX2LP již byla probrána, nyní uvedu postup, jakým nakonfigurovat hradlové pole. Ve spolupráci s nástrojem `flcli` je možno FPGA nakonfigurovat skrze jeho JTAG port. Takovýmto způsobem nahraná konfigurace nezůstane trvale v hradlovém poli, ale je vymazána resetem či vypnutím napájení.

Pro provedení této akce vyžaduje program `flcli` soubor formátu `.svf` vytvořený z bitové konfigurace FPGA `.bit`. Dále je zapotřebí, aby na FX2LP byl nahrán standardní firmware knihovny `libfpgalink` – to je provedeno automaticky.

8.5.1 Generování SVF souboru

Pro ukázku předpokládejme soubor `hlavni.bit` ve VHDL projektu pro čtení I²S mikrofonů `MicArrayBoard_I2S_to_USB`, jenž je uložen v podadresáři `DIP` domovského adresáře přihlášeného uživatele. Absolutní cesta k projektu je tudíž: `${HOME}/DIP/MicArrayBoard_I2S_to_USB`. Dále předpokládejme, že uživatel má nainstalovanou 64bitovou linuxovou distribuci a vývojové prostředí Xilinx ISE verze 14.7 je v adresáři `/opt/Xilinx`.

Klíč k tomu, jak vygenerovat `.svf` soubor z `.bit` souboru jsem nalezl ve skriptu `nexys2prog` [29] napsaném v jazyce Perl. Podstatou je vytvoření skriptu pro nástroj iMPACT (součást Xilinx ISE), který je následně proveden. Program iMPACT vyžaduje ve skriptu absolutní cesty k souborům, proto je skript vytvořen příkazem `cat` v příkazové řádce, kde se provede substituce výrazu `${HOME}` za domovský adresář[9].

[8] `https://gitlab.com/micarray-fpga/fx2lp-tools/micarrayboard-scripts`
[9] Předpokladem je, že proměnná `HOME` je v prostředí nastavena, což bývá běžné.

Skript přizpůsobený pro desku MicArrayBoard lze vytvořit tímto víceřádkovým příkazem:

```
cat > /tmp/gen_svf << END
setMode -bs
setCable -port svf -file ${HOME}/DIP/MicArrayBoard_I2S_to_USB/hlavni.svf
addDevice -p 1 -file ${HOME}/DIP/MicArrayBoard_I2S_to_USB/hlavni.bit
program -p 1
closeCable
quit
END
```

Vlastní skript leží mezi oběma výskyty ukončující sekvence znaků `END`. Tento skript se zapíše textového souboru (v příkladě jako dočasný soubor `/tmp/gen_svf`), který se jako parametr předá samotnému programu `impact` s přepínačem `-batch` takto:

```
/opt/Xilinx/14.7/ISE_DS/ISE/bin/lin64/impact -batch /tmp/gen_svf
```

Stručně vysvětlím princip výše uvedeného skriptu. Vytvoří se virtuální JTAG programátor (v terminologii Xilinx ISE: *cable*), který je přesměrován do `.svf` souboru. Programátoru je přiděleno JTAG zařízení (FPGA), jehož konfigurace je dána souborem `.bit`. Příkaz `program` provede nahrání konfigurace, což vyústí ve vytvoření obsahu `.svf` souboru.

8.5.2 Přístup k JTAG portu s využitím mikrořadiče FX2LP

Standardní firmware knihovny `libfpgalink` dokáže emulovat sběrnici JTAG na vývodech mikrořadiče FX2LP. Nástroj `flcli` zprostředkovává příkazy, které se mají na JTAG portu provést, zároveň zajistí nahrání standardního firmwaru do paměti RAM mikrořadiče.

Ověřit, zda JTAG řetězec (obsahující FPGA) je detekován, lze příkazem:

```
sudo flcli --ivp 04b4:8613 --vp 1d50:602b --query "D0D2D3D4"
```

Řetězec `D0D2D3D4` vyjadřuje přiřazení vývodů sběrnice JTAG k vývodům mikrořadiče, což odpovídá tabulce 7.1.

Přehrát SVF soubor vytvořený skriptem v předchozí sekci lze následujícím příkazem (zalomen na dva řádky pro přehlednost):

```
sudo flcli --ivp 04b4:8613 --vp 1d50:602b --program \
"J:D0D2D3D4:${HOME}/DIP/MicArrayBoard_I2S_to_USB/hlavni.svf"
```

Provedením tohoto příkazu se nakonfiguruje hradlové pole na desce MicArrayBoard.

8.6 Nahrávání paměti SPI flash

Původně bylo zamýšleno programovat flash paměť také pomocí sběrnice JTAG, ve skriptu pro iMPACT by to znamenalo pouze přidat dalších pár příkazů. Když jsem se o to pokusil, objevila se na konzoli hláška:

```
File generation is not currently available for SPI devices attached to a
Spartan-3E FPGA.
```

Není tedy podporováno generování SVF souborů, pokud je k hradlovému poli Spartan-3E přiřazena SPI flash paměť, jež se má naprogramovat.

Podařilo se mi naštěstí přijít na jiný způsob, jakým nahrát SPI flash paměť – pomocí USB mikrořadiče FX2LP, který bude emulovat rozhraní SPI. Vytvořil jsem k tomu pythonovský skript `spiflash.py` využívající knihovnu `libfpgalink`, která zajistí emulaci SPI rozhraní na vývodech mikrořadiče FX2LP. Toto rozhraní lze ovládat ze strany PC přes USB. Zdrojový kód tohoto skriptu je umístěn na Gitlabu[10] pod svobodnou licencí GNU LGPL verze 3.

8.6.1 Emulace rozhraní SPI na čipu FX2LP

Naprogramování flash paměti emulací SPI rozhraní vyžaduje nahrání – k tomuto účelu určené – konfigurace do hradlového pole, která interně propojí SPI vývody flash paměti s USB mikrořadičem. K tomu slouží jednoduchý Xilinx ISE projekt `MicArrayBoard_SPI_durch`[11] napsaný ve VHDL. Ten je před programováním paměti přes JTAG port nahrán v binární podobě do FPGA.

Jelikož se jedná o sběrnici SPI, jsou mezi sebou propojeny celkem 4 páry vývodů. K tomuto účelu jsem použil vývody PD5 až PD7 a PA0 mikrořadiče FX2LP, které jsou na desce připojeny k vývodům FPGA. Tyto vývody jsou poté interně propojeny s SPI vývody FPGA.

Ve VHDL se propojení realizuje operátorem přiřazení signálu „`<=`“. Přiřadit tímto operátorem lze vstup k výstupu (`výstup <= vstup`), nikoliv však naopak – příkladem může být vodič MOSI, jenž je jako jediný SPI signál veden ve směru z flash paměti do mikrořadiče, a proto je u něj přiřazení opačné. Zde bylo využito té vlastnosti FPGA Spartan-3E, že SPI vývody FPGA lze po načtení konfigurace používat jako uživatelské vývody. Mapování vývodů mikrořadiče je v tabulce 8.1.

Tabulka 8.1: Mapování vývodů mikrořadiče FX2LP na emulované SPI rozhraní

FX2LP	SPI flash
PD5	MISO
PD6	MOSI
PD7	CLK
PA0	CS

8.6.2 Skript `spiflash.py`

Ve skriptu `spiflash.py` je knihovně `libfpgalink` předána informace o mapování SPI vývodů pomocí textového řetězce. Význam řetězce je zdokumentován v souboru definujícím vazbu knihovny `libfpgalink` na jazyk Python – `fl.py`. Soubor `fl.py` je součástí zdrojového kódu knihovny. Kromě `fl.py` vyžaduje skript `spiflash.py` pouze standardní knihovny jazyka Python verze 3.

[10] `https://gitlab.com/micarray-fpga/fx2lp-tools/micarrayboard-scripts`
[11] Projekt je nazván podle německého durch ≡ skrze.

Knihovna `libfpgalink` poskytuje emulovaný SPI port přes USB, ale příkazy pro komunikaci s pamětí, které jsou součástí vyšší komunikační vrstvy, jsem implementoval ve vlastním skriptu. Aby byl SPI port emulován, je vyžadováno, aby na mikrořadiči byl nahrán standardní firmware knihovny, což je ve skriptu provedeno. Jednotlivé SPI instrukce pro komunikaci s pamětí jsou specifikovány v datového listu W25Q32BV [27]. Z dostupných instrukcí jsem implementoval jen ty nutné pro naprogramování a případné ověření správnosti obsahu paměti.

Po nahrání konfigurace[12] `SPI_durch` lze skriptem `spiflash.py` do paměti nahrát zvolený soubor `.bit`. Jako parametr se skriptu zadá cesta k `.bit` souboru a pro rozpoznání desky identifikátor dodavatele a produktu (ty se mohou lišit podle programu mikrořadiče). Příklad volání skriptu (v adresáři, kde je umístěn):

```
sudo ./spiflash.py -v 04b4 -p 8613 ${HOME}/DIP/MicArrayBoard_I2S_to_USB/hlavni.bit
```

Skript `spiflash.py` se následně postará o nahrání bitové konfigurace (obsažené v souboru `.bit`) na flash paměť připojenou k FPGA. Deska se takto může trvale chovat jako zvuková karta bez další potřeby přehrávání FPGA nebo FX2LP.

[12] Skript neověřuje, zda tento krok byl skutečně proveden.

Kapitola 9

Používání desky FPGA MicArrayBoard

Vymezil jsem dva rozdílné přístupy k používání (a nahrávání) desky FPGA MicArrayBoard. Přístup β („beta“) je určen spíše pro experimentování s hardwarem desky. Firmware čipu FX2LP i konfigurace hradlového pole je nahraná pouze dočasně, po vypnutí či resetu dojde k vrácení do původního stavu.

Oproti tomu přístup ψ („psí“) slouží k použití spolu s mikrofonním polem, kdy je již ověřeno, že návrh funguje správně. Firmware je permanentně (až do přepsání) uložen v paměti EEPROM a konfigurace hradlového pole je uložena v paměti typu flash s rozhraním SPI.

Oživením desky je myšleno počáteční připravení desky k následnému nahrání vlastního firmwaru a vlastní konfigurace FPGA, určených především ke čtení zvukových dat z mikrofonního pole. Cílem oživení je především zajistit příhodné podmínky pro identifikaci desky během nahrávacích operací bez ohledu na přístup používání.

Nejdříve bude popsáno oživení spolu s oběma vymezenými přístupy k používání, následuje návod, jakým způsobem lze měnit parametry mikrofonního pole připojeného k desce. Nakonec je popsáno praktické použití desky MicArrayBoard k nahrávání zvukových dat z mikrofonního pole, což bylo v rámci měření v akustické komoře provedeno.

9.1 Požadované programové vybavení

Pro oživení a následné nahrávání desky MicArrayBoard je nezbytné mít k dispozici počítač s operačním systémem Linux (jsou doporučeny distribuce založené na Debianu), na kterém je nainstalováno potřebné programové vybavení, jež je uvedeno níže. Pro instalaci programového vybavení jsou vyžadována práva superuživatele (`root`).

Potřebné programové vybavení je následující:

- vývojové prostředí pro hradlová pole Xilinx ISE WebPack,
- kompilátor `sdcc` pro mikroprocesor FX2LP,
- knihovna `libfpgalink` spolu s jejími závislostmi,
- nástroj `flcli`, jenž slouží pro nahrání programu čipu FX2LP do připojené paměti EEPROM (je součástí knihovny `libfpgalink`),

- obdobně nástroj `fx2cli` – pro nahrání firmwaru do RAM čipu FX2LP (také v knihovně `libfpgalink`),
- program pro automatizaci kompilace GNU `make`,
- interpret jazyka Python verze 3 (doporučena alespoň verze 3.10),
- sada nástrojů `usbutils` – je využíván příkaz `lsusb`, který vypíše připojená USB zařízení k počítači.
- Emulátor terminálu (například Konsole nebo Terminál GNOME) s příkazovým řádkem `bash` (Bourne-Again Shell).

Veškeré programové vybavení zmíněné výše, vyjma Xilinx ISE a knihovny `libfpgalink`, bývá dostupné k instalaci pomocí správce balíčků (může se lišit podle dané linuxové distribuce). Vývojové prostředí Xilinx ISE je nutné nainstalovat manuálně. Knihovna `libfpgalink` se musí sestavit ze zdrojového kódu, postup je popsán v sekci 8.3. Nahrání paměti flash se SPI rozhraním zajistí skript `spiflash.py` napsaný v jazyce Python, který volá funkce knihovny `libfpgalink`.

9.2 Nastavení oprávnění `udev`

Doporučeným krokem, který usnadňuje práci s deskou, je nastavení práv přístupu k tomuto USB zařízení pro běžné systémové uživatele. Tento krok spočívá ve vytvoření konfiguračního souboru pro správce zařízení `udev` (toto vytvoření vyžaduje superuživatelská oprávnění).

Konfigurační soubor `40-fx2lp.rules`[1], určený pro čip FX2LP bez vlastního firmwaru či se standardním firmwarem knihovny `libfpgalink`, se vytvoří tímto víceřádkovým příkazem:

```
sudo tee /etc/udev/rules.d/40-fx2lp.rules << END
# Cypress Semiconductor Corp. CY7C68013 EZ-USB FX2 USB 2.0 Development Kit
SUBSYSTEM=="usb", ATTR{idVendor}=="04b4", ATTR{idProduct}=="8613", MODE:="0666"
# OpenMoko, Inc. FPGALink; USB to JTAG firmware
SUBSYSTEM=="usb", ATTR{idVendor}=="1d50", ATTR{idProduct}=="602b", MODE:="0666"
END
```

Při práci s deskou je výhodné mít zvýšená oprávnění, i v případě, kdy je na ní nahrán program `FX2LP_audio_ISO`. Její USB identifikátory může uživatel změnit, proto je vhodné pravidla uložit do samostatného souboru. Následující příkaz vytvoří soubor s `udev` pravidly pro desku FPGA MicArrayBoard:

```
sudo tee /etc/udev/rules.d/42-micarrayboard.rules << END
# FPGA MicArrayBoard
SUBSYSTEM=="usb", ATTR{idVendor}=="cafe", ATTR{idProduct}=="1337", MODE:="0666"
END
```

Pro okamžitý účinek změn se musí znovu načíst pravidla správce zařízení, jinak se vytvořený konfigurační soubor projeví po restartu systému. Pravidla správce zařízení se znovu načtou následujícím příkazem:

```
sudo udevadm control --reload-rules
```

Nově připojenou desku je nyní možno obsluhovat bez práv superuživatele (tj. bez privilegovaného příkazu `sudo`).

[1] Číselná předpona souboru udává pořadí pravidel při jejich načítání – čím nižší číslo, tím dříve budou načteny.

9.3 Počáteční kroky před oživením

Předpokladem pro oživení je plošný spoj osazený potřebnými součástkami. Navíc k tomu deska vyžaduje několik zkratovacích propojek na kolíkové lišty o rozteči 2,54 mm (anglicky se značí „header jumper“).

Pro připojení desky k počítači slouží kabel s koncovkami USB A samec a USB mikro B samec. Před prvním zapojením desky by měly být na kolíkové konektory umístěny propojky níže uvedeným způsobem:

- Propojka na J9 vybírá USB jako zdroj napájení.
- Vývod M2 bez propojky, M1 zkratován propojkou (FPGA se bude zavádět z rozhraní JTAG).
- Na konektoru JTAG_3V3 zkratovat propojkami signály TDI, TDO, TMS a TCK (připojí se JTAG signály z FX2LP k hradlovému poli).
- Propojky jsou umístěny na konektorech JP1 a JP2 (výstupy regulátorů 2,5 a 1,2 V).
- Přemostit propojkou na konektoru CLK_SELECT vodiče SEL a OSC (vybere oscilátor jako zdroj hodin FPGA).

Při rozmisťování propojek se lze řídit fotografií osazené desky C.1. Na fotografii lze vidět, že na JTAG konektoru je umístěna čtyřnásobná propojka v červeném plastovém pouzdře namísto čtyř jednoduchých propojek.

Při prvním přivedení napájení je doporučeno ověřit multimetrem, zda na výstupech regulátorů je přítomno požadované napětí. Měla by svítit LED dioda D1 indikující napájení.

9.4 Proces oživení desky

1. Připojte desku k počítači pomocí USB kabelu.

2. Ověřte příkazem `lsusb`, že se USB čip ohlásil systému (výchozími identifikátory). Ve výpisu by se měl objevit následující řádek (čísla za `Bus` a `Device` se mohou lišit):

   ```
   Bus 001 Device 031: ID 04b4:8613 Cypress Semiconductor Corp. CY7C68013
       EZ-USB FX2 USB 2.0 Development Kit
   ```

   Pokud se ve výpisu objevuje takovýto řádek, USB čip FX2LP se chová tak, jak by měl.

3. Součástí linuxového jádra je modul `usbtest`, který se automaticky načte pro USB zařízení s identifikátory odpovídajícími čipu FX2LP (ID dodavatele `04b4` a ID produktu `8613`). Pro oživení desky je však `usbtest` nežádoucí jelikož s čipem v takovém případě nelze komunikovat, dokud je vlastněn tímto modulem. Modul se odstraní příkazem (vyžaduje oprávnění superuživatele):

   ```
   sudo rmmod usbtest
   ```

4. Nahrání firmwaru pro nástroj `flcli`, který emuluje JTAG rozhraní přes USB, do paměti EEPROM:

   ```
   sudo flcli --ivp 04b4:8613 --vp 1d50:602b --eeprom=std
   ```

   Přepínač `--ivp` zadává původní ID dodavatele a ID produktu (VID:PID), přepínač `--vp` potom hodnotu VID:PID zavaděče 2. stupně. Tento firmware je v rámci knihovny `libfpgalink` zabudován, proto je označen jako standardní (`std`).

Provedením těchto kroků je deska MicArrayBoard připravena k používání. Při každém přivedení napájení se do USB mikroprocesoru FX2LP načte standardní firmware knihovny `libfpgalink`, jenž umožňuje s deskou dále pracovat, například konfigurovat hradlové pole přes rozhraní JTAG.

9.5 Nahrávání desky

Má-li uživatel potřebu změnit na desce firmware FX2LP či konfiguraci FPGA, měl by mít k dispozici odpovídající binární soubory, které se následně nahrají na desku. Vytvoření binárních souborů firmwaru ze zdrojových souborů je automatizováno souborem `Makefile`. Oproti tomu, syntéza VHDL projektů je stále odkázána na manuální provedení, přičemž automatizace by byla možná.

Nahrání binárních souborů je automatizováno celé, pokud uživatel využije pracovní prostor, který byl pro zjednodušení nahrávání desky MicArrayBoard vytvořen. V této sekci bude popsána struktura tohoto pracovního prostoru a jeho použití s deskou.

9.5.1 Struktura pracovního prostoru

Pro nahrávání firmwaru FX2LP a konfigurace hradlového pole byla připravena pracovní složka (neboli pracovní prostor) se všemi potřebnými zdrojovými kódy. Pracovní prostor zabalený v archivu je součástí elektronické přílohy. Kořenový soubor `Makefile` definuje potřebné sekvence příkazů (tzv. *cíle*), které usnadňují nahrávání desky.

Stromová struktura pracovního prostoru (pojmenovaného `DIP`) je následující (názvy složek jsou zakončeny lomítkem):

- `DIP/` ..... Složka pracovního prostoru
  - `Makefile` ..... Soubor pro nástroj GNU `make`
  - `fx2lib/` ..... Zdrojový kód knihovny `fx2lib`
  - `FX2LP_audio_ISO/` ..... Zdrojový kód firmwaru mikrořadiče FX2LP implementujícího USB zařízení třídy audio (git repozitář)
  - `MicArrayBoard_I2S_to_USB/` VHDL projekt pro čtení I²S mikrofonů
  - `MicArrayBoard_SPI_durch/` .. VHDL projekt pro nahrávání paměti SPI flash
  - `MicArrayBoard_TDM_to_USB/` ..... VHDL projekt pro čtení TDM mikrofonů
  - `micarrayboard-scripts/` ..... Pomocné skripty pro desku MicArrayBoard (git repozitář)

V pracovním prostoru je navíc složka `MicArrayBoard_TDM_to_USB_simul`, ale ta obsahuje pozměněný zdrojový kód TDM projektu pro simulaci TDM mikrofonů na FPGA a není zahrnuta v automatizaci nahrávání.

9.5.2 Příprava pracovního prostoru

V souboru `Makefile` je definována proměnná `XILINX`, jež obsahuje cestu k vývojovému prostředí Xilinx ISE verze 14.7. Instaloval-li uživatel prostředí Xilinx ISE do jiného adresáře, nechť změní obsah této proměnné. Ostatní proměnné v souboru smí uživatel ponechat beze změn.

Nejdříve by měl být syntetizován VHDL projekt, který si uživatel přeje používat, v programu Xilinx Project Navigator. V případě čtení I²S mikrofonů je to projekt `MicArrayBoard_I2S_to_USB`. Budou-li čteny TDM mikrofony, potom je namístě VHDL projekt `MicArrayBoard_TDM_to_USB`. Projekty jsou umístěny ve stejnojmenných složkách. Návod, jak syntetizovat projekt v Xilinx ISE je v sekci 3.3.

V kořenovém adresáři pracovním prostoru by následně měl být otevřen příkazový řádek. Je náležité, aby deska MicArrayBoard byla připojena k počítači. Následné příkazy se liší podle přístupu k používání.

9.5.3 Nahrávání desky – přístup β

Soubor bitové konfigurace FPGA, pokud byl generován syntézou, lze následně automaticky převést na formát SVF a nahrát přes JTAG port na FPGA. Tyto kroky provede nástroj `make` při zavolání jednoho z následujících cílů.

Návrhu `MicArrayBoard_I2S_to_USB`, který čte I²S mikrofony, odpovídá cíl `I2S` a tudíž příkaz:

```
make I2S
```

Obdobně návrhu čtoucímu TDM mikrofony odpovídá příkaz:

```
make TDM
```

Cíl `audio` zkompiluje a nahraje firmware, jenž se chová jako zvuková karta, do paměti RAM mikrořadiče:

```
make audio
```

Nahraný mikrořadič se poté ve výpisu `lsusb` objeví takto:

```
Bus 001 Device 015: ID cafe:1337 ČVUT FPGA MicArrayBoard
```

9.5.4 Nahrávání desky – přístup ψ

Bitová konfigurace FPGA (soubor `.bit`), v případě přístupu ψ, lze bez dalšího zpracování nahrát na připojenou paměť typu flash přes SPI rozhraní. Konfigurace FPGA se bude načítat z paměti SPI flash, budou-li na desce zkratovány vývody M2 a M1 propojkou na zem.

Projekt `MicArrayBoard_SPI_durch`, který umožňuje přístup ke SPI rozhraní flash paměti, byl již předem syntetizován do binárního souboru konfigurace `SPI_durch.bit`. Nadto byl tento binární soubor konvertován do formátu SVF, tudíž je připraven k nahrání na hradlové pole přes JTAG port.

Zápis bitové konfigurace projektu `MicArrayBoard_I2S_to_USB` na flash paměť se provede příkazem:

```
make I2S_SPI
```

Bitová konfigurace projektu `MicArrayBoard_TDM_to_USB` se analogicky nahraje zavoláním cíle `TDM_SPI`.

Nahrání programu mikrořadiče FX2LP do paměti I²C EEPROM se provede následovně:

```
make audio_EEPROM
```

Interně je v příkazu volán nástroj `flcli` s předurčenými parametry.

9.6 Změna parametrů mikrofonního pole

Mnou připravené projekty pro desku MicArrayBoard lze dále přizpůsobit, aby například byla použita jiná vzorkovací frekvence nebo jiný počet I²S či TDM rozhraní. K tomuto účelu slouží parametry definované v příslušných zdrojových kódech. Zdrojový kód FX2LP firmwaru `FX2LP_audio_ISO` má tyto parametry definované v deskriptoru – soubor `descriptor.a51`. Vzhledem k tomu, že čip FX2LP má univerzální paralelní sběrnici, není třeba rozlišovat firmware podle použitého mikrofonního rozhraní, takže stačí jediná varianta s několika proměnnými parametry.

9.6.1 Parametry firmwaru `FX2LP_audio_ISO`

Klíčové parametry firmwaru jsou dva – `F_SAMP` a `CHANNELS`. Parametr `F_SAMP` určuje vzorkovací kmitočet v jednotkách Hz. Tento kmitočet by měl být totožný s tím, který je vybrán v používaném FPGA návrhu. Druhý parametr, `CHANNELS`, nastavuje celkový počet zvukových kanálů posílaných do počítače. Hodnota tohoto parametru, pokud jsou čteny I²S mikrofony, je dvojnásobek aktivních I²S rozhraní. V případě čtení TDM mikrofonů je to násobek počtu rozhraní a délky TDM řetězců (implementace pracuje se shodnou délkou pro všechna rozhraní).

Není doporučeno měnit hodnotu parametru `AUDIO_PACKET_SIZE`. Ten stanovuje, jaká je maximální velikost paketu obsahujícího vzorky zvuku. Aktuální velikost paketu vzorků může být i menší, proto je nastavena maximální velikost na limit isochronního koncového bodu – to jest paket o velikosti 1024 B.

9.6.2 Parametry I²S rozhraní

Parametry ve VHDL projektu `MicArrayBoard_I2S_to_USB` jsou definované jakožto VHDL konstanty v souboru `konstanty.vhd`, který je koncipován jako knihovna s definicí konstant. Vzorkovací kmitočet se vybírá konstantou `VYBRANA_FVZ`. Konstanta je definována jako výčtový typ, může tedy nabývat jen definovaných hodnot. Namísto číselných hodnot jsou definovány hodnoty jako `fvz_8k`, `fvz_48k`, `fvz_96k`, atd. Za podtržítkem je hodnota vzorkovací frekvence – např. `48k` odpovídá frekvenci 48 kHz.

Další parametr, `ROZHRANI`, vybírá počet aktivních I²S rozhraní, ze kterých se budou číst data. Na desce MicArrayBoard lze takto aktivovat až 8 I²S rozhraní. Aktivní rozhraní jsou vždy ta s nižším indexem (ten je uveden vedle konektorů na desce). Pokud jsou tedy aktivována 4 rozhraní, jsou využity porty I2S-0, I2S-1, I2S-2 a I2S-3.

Třetí z důležitých konstant je `VZORKU_NA_PAKET`. Tou je určeno, jak veliké pakety se zvukovými daty mají být odesílány. Jelikož FPGA řídí odesílání paketů, je hodnota této konstanty klíčová. Hodnotu lze vypočíst ze vzorkovacího kmitočtu a celkového počtu zvukových kanálů. Předpokládáme-li interval mezi jednotlivými pakety 125 µs, lze dojít ke vztahu

$$\texttt{VZORKU\_NA\_PAKET} = f_{\mathrm{vz}} \cdot K \cdot 125 \cdot 10^{-6},$$

kde f_{vz} je vzorkovací frekvence v Hz a K je celkový počet zvukových kanálů.

Příklad výpočtu pro 4 aktivní I²S rozhraní (8 zvukových kanálů):

$$\texttt{VZORKU\_NA\_PAKET} = 48000 \cdot 8 \cdot 125 \cdot 10^{-6} = 48.$$

To odpovídá, uvážíme-li, že velikost vzorku je 4 bajty, paketu o velikosti 192 B. Konstanta `VZORKU_NA_PAKET` musí maximálně dosahovat délky 256 vzorků. Navíc aby zvukové kanály obsahovaly příslušející vzorky, musí být celočíselným násobkem celkového počtu kanálů[2].

9.6.3 Skript pro nastavení I²S rozhraní

Aby nastavení všech souvisejících parametrů návrhu bylo pohodlnější, pro I²S rozhraní jsem vytvořil `bash` skript `i2sconfig.sh`. Po jeho spuštění může uživatel v textovém menu vybrat žádanou vzorkovací frekvenci a počet I²S rozhraní. Skript lze spustit v pracovním prostoru příkazem:

```
make I2S_CONF
```

Menu je vytvořeno přímo v textovém rozhraní terminálu pomocí programu `dialog`. Mezi položkami v menu se pohybuje šipkami nahoru a dolů. Šipky vlevo a vpravo vybírají mezi volbou „Budiž“ nebo „Storno“. Stisk klávesy Enter potvrdí volbu.

Potvrzením změn v posledním okně dialogu jsou všechny potřebné parametry vyhledány a změněny ve zdrojových souborech nástrojem pro transformaci textu `sed` (Stream Editor). K aplikování změn ve zdrojových souborech je vyžadováno znovu syntetizovat návrh FPGA a zkompilovat firmware mikrořadiče FX2LP.

9.6.4 Parametry pro TDM mikrofony

VHDL projekt pro čtení TDM mikrofonů, `MicArrayBoard_TDM_to_USB`, má obdobné řešení definice parametrů v knihovně `konstanty.vhd` jako má projekt pro I²S rozhraní.

Oproti I²S projektu má `VYBRANA_FVZ` méně možných hodnot vzorkovací frekvence – lze vybrat 8, 16, 24 nebo 48 kHz. Obdobně jako u I²S projektu

[2] Pokud tato podmínka není dodržena, dojde k „cestování“ přijatých vzorků mezi kanály.

určuje konstanta `ROZHRANI` počet aktivně čtených TDM rozhraní. Konstanta `MIKROFONU` zde specifikuje počet TDM mikrofonů v řetězci, který je očekáván shodný *na každém* čteném rozhraní. Celkový počet zvukových kanálů je tudíž násobkem konstant `MIKROFONU` a `ROZHRANI`. Vzorkovací kmitočet 16 kHz s řetězcem delším než 8 mikrofonů není podporován, protože vychází lichý dělitel kmitočtu, což implementace na FPGA nepodporuje.

Pro nastavení parametrů TDM není vytvořen skript, proto je doporučeno dbát zvýšené opatrnosti při jejich změně. Je důležité, aby hodnoty parametrů ve firmwaru `FX2LP_audio_ISO` reflektovaly provedené změny v FPGA návrhu. Navíc je počet vzorků v jednom paketu omezen na 256, což omezuje celkový počet kanálů. Například při vzorkovacím kmitočtu 48 kHz je maximum 32 kanálů (volíme-li počet kanálů jako mocninu dvou).

9.7 Nahrávání zvuku

Díky implementaci standardního USB zařízení třídy audio lze, alespoň v případě operačního systému Linux, využít již integrovaných standardních ovladačů (subsystému ALSA). To umožňuje desku MicArrayBoard, která se tváří jako zvuková karta, otevřít v jakémkoli programu využívajícím architekturu ALSA.

Nahrávat zvukovou kartou MicArrayBoard jsem odzkoušel v programu Audacity[3]. Podařilo se mi tak nahrát až 128 kanálů současně, ale s tím omezením, že exportovat do zvukového souboru (například WAV – Waveform Audio File Format) lze nanejvýš 32 kanálů (to je omezení uvnitř Audacity, nikoliv formátu).

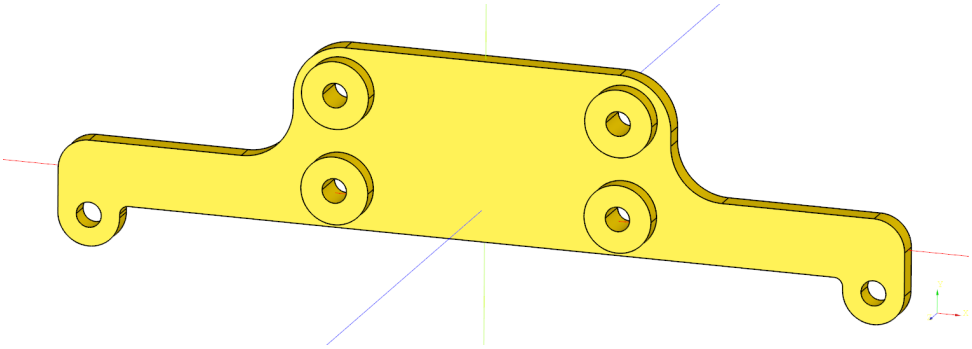
9.7.1 Systémová kompatibilita desky MicArrayBoard

Deska MicArrayBoard s firmwarem `FX2LP_audio_ISO`, s nímž se deska tváří jako USB zvuková karta, byla vyzkoušena na několika přístrojích s různými operačními systémy. Systém macOS 13.4.1 Ventura USB zařízení detekuje, avšak k němu nenačte vhodný ovladač, takže nelze použít jako osmikanálová zvuková karta. Podobný problém nastává i na operačním systému Windows 10.

Linux byl odzkoušen na více strojích, například na Raspberry Pi 4 s distribucí Raspberry Pi OS lze desku vidět jako zvukové zařízení i nahrávat zvuk, avšak po připojení se z nějakého důvodu v grafickém prostředí LXDE zhroutí panel `lxpanel`. Pád panelu se uskutečnil i po připojení komerční USB zvukové karty M-AUDIO Fast Track Pro, problém je tudíž v programovém vybavení distribuce.

Na počítači s distribucí Linux Mint 21.1 a verzí jádra systému 5.15.0 pracuje tato deska bez problémů. Na dalších počítačích s distribucemi Xubuntu 20.04 (jádro verze 5.13) a Ubuntu 22.04 (jádro verze 6.2) byla deska také funkční.

[3] `https://www.audacityteam.org`



Obrázek 9.1: 3D model držáku desky MicArrayBoard na rameno krokového motoru

9.8 Měření kruhového mikrofonního pole s deskou FPGA MicArrayBoard

Deska FPGA MicArrayBoard posloužila praktické realizaci měření modulárního kruhového mikrofonního pole, navrženého Ing. Davidem Vagnerem, v akustické bezodrazové komoře v budově FEL. Jak bylo uspořádáno a jak probíhalo měření již bylo popsáno Ing. Davidem Vagnerem v jeho diplomové práci [30], kde byly získané zvukové nahrávky také zpracovány. Doplním k tomu informaci, že měření bylo ovládáno notebookem s linuxovou distribucí Linux Mint 21.1.

Změřené mikrofonní pole má celkem 8 I²S mikrofonů rovnoměrně rozmístěných po obvodu, střídají se mikrofony odesílající levý a pravý kanál. Pro otáčení mikrofonním polem posloužil přípravek Bc. Davida Ringsmutha s krokovým motorem umístěným na stojanu [31]. Na hřídel krokového motoru bylo kruhové pole nasazeno přes adaptér se závitem a závěsným ramenem vytištěný na 3D tiskárně. Tento adaptér vymodeloval Ing. David Vagner, a je součástí příloh k jeho diplomové práci.

9.8.1 Držák desky

Deska MicArrayBoard byla k adaptéru během měření improvizovaně přivázána drátem. Později jsem vymodeloval přídavný držák, který lze k ramenu adaptéru přišroubovat pomocí metrických šroubů M3 se speciálně vymodelovanými podložkami. Deska je s držákem spojena přišroubováním skrze otvory v jejích rozích (také šrouby M3). Závity pro šrouby M3 se do plastového držáku vyřežou šroubem samotným.

Držák jsem vymodeloval v parametrickém modelovacím systému CadQuery[4], výsledný model je na obrázku 9.1. Fotografie desky uchycené tímto držákem ke krokovému motoru je na obrázku C.3. Na fotografii lze také zpozorovat, že deska již je osazena 1,2V regulátorem se správným rozvržením vývodů.

[4] `https://github.com/CadQuery/cadquery`

9.8.2 Skript řídící měření

K zajištění součinnosti otáčení krokového motoru s nahráváním osmikanálového zvuku jsem napsal skript v jazyce Python – `krokac.py`. Skript je umístěn spolu s pomocnými skripty na Gitlabu[5]. Ovládání krokového motoru je řešeno komunikací s přípravkem, který řídí krokový motor, přes sériový port emulovaný rozhraním USB. Na desce Arduino® UNO, která je uvnitř přípravku, byl nahrán firmware `StandardFirmata`, jenž je součástí příkladů knihovny Firmata[6] ve vývojovém prostředí Arduino IDE. Knihovna Firmata je součástí výchozí instalace Arduino IDE.

Ze strany skriptu je komunikace s přípravkem řešena knihovnou `pyfirmata`. Nahrávání zvuku řeší knihovna `sounddevice`, ke zpracování vzorků slouží knihovny `numpy` a `scipy`. Jako testovací zvukový signál byl zvolen tzv. *sine sweep*, což je sinusový signál, u kterého se v čase zvyšuje frekvence. Ten byl přehráván ze studiového reproduktoru připojeného k USB zvukové kartě.

Na některých linuxových distribucích se stává, že deska MicArrayBoard v knihovně `sounddevice` není registrována jako samostatné zařízení, když je vybrána systémem jako výchozí vstupní zařízení. Namísto toho na ní odkazuje zařízení `default`. To považuji za limitaci knihovny PortAudio[7] (napsané v jazyce C), na níž je knihovna `sounddevice` postavená.

Ve skriptu není implementován žádný výběr zvukových zařízení. To je z toho důvodu, že výstupní i vstupní zvuková karta byla zvolena jako výchozí v systémovém dialogu `cinnamon-settings`.

9.8.3 Zvukové nahrávky

Výstupem skriptu je složka se zvukovými soubory (název složky začíná slovem `measurement`, na konci názvu je datum a čas měření) v bezeztrátovém formátu WAV. Každý zvukový soubor je pojmenován `krok_M`, kde parametr `M` udává pořadí kroku motoru od nuly do 199 (plná otáčka odpovídá 200 krokům u použitého krokového motoru). Z tohoto parametru lze přímo určit úhel natočení mikrofonního pole.

Pořadí kanálů v nahraných zvukových souborech je oproti pořadí, které do počítače posílá deska MicArrayBoard, upraveno tak, aby reprezentovalo fyzickou polohu mikrofonů (ve směru hodinových ručiček při pohledu na mikrofonní pole shora). Mikrofony byly na kruhovém poli kolem dokola označeny písmenem značícím odesílaný kanál (L – levý, P – pravý) a číslem pořadí páru mikrofonů. Číslo páru mikrofonů odpovídá číslu I²S portu podle označení na desce, avšak je k němu přičtena jednička. Např. mikrofon P1 je připojen na rozhraní I2S-0 a mikrofon L3 na I2S-2.

Pořadí kanálů ve zvukovém záznamu je tedy P1, L1, P2, …, P4, L4. V počáteční poloze byl nasměrován mikrofon P1 směrem k protějšímu reproduktoru. Krokový motor mikrofonním polem postupně otáčel ve směru hodinových ručiček, každé dva kroky (3,6 °) se nahrál zvukový záznam.

[5] `https://gitlab.com/micarray-fpga/recording-tools`
[6] `https://github.com/firmata/arduino`
[7] `https://github.com/PortAudio/portaudio`

Závěr

Výsledky diplomové práce

V rámci této diplomové práce bylo vyvinuto řešení pro příjem zvukových dat z MEMS mikrofonů skrze I²S a TDM rozhraní pomocí programovatelného hradlového pole Xilinx Spartan-3E. Na hradlové pole byl navržen logický obvod v jazyce VHDL pro čtení I²S respektive TDM mikrofonů.

Teoretická část práce obsahuje seznámení s MEMS mikrofony spolu s jejich digitálními rozhraními I²S a TDM. V praktické části je popsána implementace a činnost jednotlivých součástí přenosového řetězce.

Čtení zvukového signálu rozhraním I²S bylo odzkoušeno na reálných MEMS mikrofonech v laboratoři. Současně bylo čteno 8 I²S mikrofonů (na celkem čtyřech rozhraních) vzorkovacím kmitočtem 48 kHz. Vyšší vzorkovací kmitočet, například 96 kHz, je možné zvolit, nebyl však ověřen v praxi, jelikož to nedovolily použité I²S mikrofony.

Implementace čtení TDM rozhraní byla v praxi ověřena na řetězci pěti TDM mikrofonů ICS-52000 vzorkovacím kmitočtem 24 kHz (delší řetězec se nepodařilo zprovoznit). Kromě toho bylo úspěšně čteno osm TDM rozhraní simulovaných na FPGA, kdy každé mělo připojených osm mikrofonů, vzorkovacím kmitočtem 24 kHz. Současně bylo tudíž posíláno do počítače 64 zvukových kanálů, čímž se vyznačuje málokterá zvuková karta.

Přečtené vzorky zvuku z mikrofonů jsou pomocí USB mikrořadiče FX2LP posílány do počítače pomocí isochronního přenosu. Na mikrořadič byl napsán firmware v jazyce C spolu s USB deskriptorem v assembleru. Deskriptor implementuje USB zařízení třídy audio verze 1.0, takže mikrořadič se chová jako USB zvuková karta. Tato implementace USB zařízení je kompatibilní s operačním systémem Linux.

Pro realizaci navrženého systému byla vyrobena vlastní deska plošných spojů nazvaná MicArrayBoard. Popisu jejího návrhu je věnována celá kapitola. Výzvou bylo naleznout programové řešení, které by bylo schopné na desku dostat bitovou konfiguraci hradlového pole, především na připojenou paměť typu flash.

Knihovna `libfpgalink`, která má svobodnou licenci zdrojového kódu GNU LGPL v3, nakonec poskytla jednotné řešení problematiky nahrávání jak firmwaru mikrořadiče FX2LP, tak konfigurace FPGA na desku MicArray-

Board. Způsob a programové vybavení, kterým se tyto součástky nahrávají, je v práci také popsán.

Deska MicArrayBoard byla využita při měření modulárního kruhového mikrofonního pole s osmi I²S mikrofony v akustické bezodrazové komoře pro účely diplomové práce Ing. Davida Vagnera.

K realizaci mikrofonního pole se ukázalo rozhraní I²S vhodnější než TDM, a to z několika důvodů. Jedním z důvodů je nižší kmitočet hodin, a tím i menší náchylnost na impedanční nepřizpůsobení přívodních vodičů. Další, zcela praktický důvod, je výrazně širší výběr MEMS mikrofonů s I²S rozhraním dostupných na trhu.

Návrh na další postup

Do budoucna by bylo vhodné pro desku MicArrayBoard vyvinout ovladač pro operační systém Windows, což by rozšířilo možnosti uplatnění této desky. S tím souvisí také skutečnost, že by bylo žádoucí, aby deska měla oficiální identifikátor produktu (PID) pod některým z registrovaných dodavatelů USB zařízení.

Kromě modulárního kruhového mikrofonního pole by mohlo být vymodelováno a k desce připojeno pole kulové s 16 I²S mikrofony. Případně pro tvarování směrových charakteristik by mohlo být vytvořeno lineární mikrofonní pole. Přímo na hradlovém poli by také mohlo probíhat číslicové zpracování přečtených zvukových signálů. Zpracovaná data by poté byla posílána do počítače.

Literatura

[1] VAGNER, David. *Návrh sférického mikrofonního pole*. Praha, 2021. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra radioelektroniky. Vedoucí práce Ing. Petr Honzík, Ph.D. Dostupné také z: https://dspace.cvut.cz/handle/10467/94677

[2] *SPH0645LM4H-B: I2S Output Digital Microphone* [online]. Knowles, 2017 [cit. 2024-01-05]. Dostupné z: https://www.knowles.com/docs/default-source/model-downloads/sph0645lm4h-b-datasheet-rev-c.pdf

[3] ŠEDIVÝ, Jan. *Komunikace s přípravkem Spartan3E pomocí rozhraní RS232*. Praha, 2021. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra telekomunikační techniky. Vedoucí práce Ing. Pavel Lafata, Ph.D. Dostupné také z: https://dspace.cvut.cz/handle/10467/94715

[4] *MP23DB01HP Datasheet* [online]. STMicroelectronics, 2022 [cit. 2023-05-02]. Dostupné z: https://www.st.com/resource/en/datasheet/mp23db01hp.pdf

[5] LEWIS, Jerad. *Inter-IC Digital Audio Interfaces* [online]. InvenSense, 2015 [cit. 2023-05-02]. Dostupné z: https://invensense.tdk.com/download-pdf/inter-ic-digital-audio-interfaces/

[6] *INMP441: Omnidirectional Microphone with Bottom Port and I²S Digital Output* [online]. InvenSense, 2014 [cit. 2023-01-24]. Dostupné z: https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf

[7] *I²S bus specification* [online]. Philips Semiconductors, 1986 [cit. 2023-01-19]. Dostupné z: https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf

[8] *ICS-52000: Low-Noise Microphone with TDM Digital Output* [online]. InvenSense, 2017 [cit. 2023-01-19]. Dostupné z: https://invensense.tdk.com/download-pdf/ics-52000-datasheet/

[9] KAINKA, Burkhard. *USB – měření, řízení a regulace pomocí sběrnice USB*. 1. vyd. Praha: BEN – technická literatura, 2002. ISBN 80-7300-073-3.

[10] *Universal Serial Bus Specification, Revision 2.0* [online]. USB Implementers Forum, 2000 [cit. 2023-02-11]. Dostupné z: https://www.usb.org/document-library/usb-20-specification

[11] Getting a Vendor ID. *USB-IF* [online]. USB Implementers Forum [cit. 2023-05-24]. Dostupné z: https://www.usb.org/getting-vendor-id

[12] USB Endpoints. *Microchip Developer Help* [online]. Microchip Technology, 2021 [cit. 2023-02-11]. Dostupné z: https://microchipdeveloper.com/usb:endpoints

[13] VAKKANTULA, Rama Sai Krishna. *AN4053: Streaming Data Through Isochronous or Bulk Endpoints on EZ-USB® FX2™ and FX2LP™* [online]. Cypress Semiconductor, 2017 [cit. 2023-12-19]. Dostupné z: https://www.infineon.com/dgdl/?fileId=8ac78c8c7cdc391c017d07353e5b5885

[14] *EZ-USB® Technical Reference Manual* [online]. San Jose: Cypress Semiconductor, 2019 [cit. 2023-05-24]. Dostupné z: https://www.infineon.com/dgdl/Infineon-EZ-USB_TECHNICAL_REFERENCE_MANUAL-EN.pdf?fileId=8ac78c8c7d0d8da4017d0f9093657d61

[15] *CY7C68013A/14A/15A/16A EZ-USB FX2LP USB Microcontroller High-Speed USB Peripheral Controller* [online]. San Jose: Cypress Semiconductor, 2021 [cit. 2023-05-24]. Dostupné z: https://www.infineon.com/dgdl/Infineon-CY7C68013A-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ec9f7974252

[16] USB capture setup – Linux. *Wireshark Wiki* [online]. Wireshark Foundation, 2020 [cit. 2023-11-13]. Dostupné z: https://wiki.wireshark.org/CaptureSetup/USB#Linux

[17] *DS312: Spartan-3E FPGA Family Data Sheet* [online]. Xilinx, 2018 [cit. 2023-05-16]. Dostupné z: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

[18] *IEEE Std 1149.1 (JTAG) Testability Primer* [online]. Texas Instruments, 1997 [cit. 2023-12-01]. Dostupné z: https://www.ti.com/lit/an/ssya002c/ssya002c.pdf

[19] CAMMON, Justin a Brendan BRIDGFORD. *XAPP503: SVF and XSVF File Formats for Xilinx Devices (ISE Tools)* [online]. Xilinx, 2017 [cit. 2023-12-13]. Dostupné z: https://docs.xilinx.com/v/u/en-US/xapp503

[20] *USB Audio Streaming: USBStreamer Kit* [online]. miniDSP, 2023 [cit. 2023-12-06]. Dostupné z: https://www.minidsp.com/products/usb-audio-interface/usbstreamer

[21] *AN295: USB Audio Class Tutorial* [online]. Silicon Laboratories, 2006 [cit. 2023-11-18]. Dostupné z: https://www.silabs.com/documents/public/application-notes/AN295.pdf

[22] *BDxxFC0 series: Single-Output LDO Regulators* [online]. ROHM, 2017 [cit. 2023-05-16]. Dostupné z: https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/linear_regulator/bdxxfc0wefj-e.pdf

[23] *AP7361: 1A LOW DROPOUT ADJUSTABLE AND FIXED-MODE REGULATOR WITH ENABLE* [online]. Diodes Incorporated, 2020 [cit. 2023-08-18]. Dostupné z: https://www.diodes.com/assets/Datasheets/AP7361.pdf

[24] *Nexys 3 Schematic Rev. B* [online]. Digilent, 2016 [cit. 2023-01-24]. Dostupné z: https://digilent.com/reference/_media/reference/programmable-logic/nexys-3/nexys3_sch.pdf

[25] HYMEL, Shawn. How to Route Differential Pairs in KiCad (for USB). *Maker.io® powered by Digi-Key* [online]. DigiKey, 2020 [cit. 2023-11-06]. Dostupné z: https://www.digikey.cz/en/maker/projects/how-to-route-differential-pairs-in-kicad-for-usb/45b99011f5d34879ae1831dce1f13e93

[26] *XAPP453: The 3.3V Configuration of Spartan-3 FPGAs* [online]. Xilinx, 2008 [cit. 2023-12-10]. Dostupné z: https://docs.xilinx.com/v/u/en-US/xapp453

[27] *W25Q32BV: 3V 32M-BIT SERIAL FLASH MEMORY WITH DUAL AND QUAD SPI* [online]. Winbond, 2013 [cit. 2023-12-12]. Dostupné z: https://www.laskakit.cz/user/related_files/w25q32bv_datasheet.pdf

[28] Ubuntu Manpage: fxload – Firmware download to EZ-USB devices. *Ubuntu Manpage Repository* [online]. Canonical, 2019 [cit. 2023-09-19]. Dostupné z: https://manpages.ubuntu.com/manpages/lunar/en/man8/fxload.8.html

[29] ROSS, Andrew. nexys2prog - Easy programming of Digilent's Nexys(2). *ixo.de USB JTAG pod* [online]. SourceForge, 2010 [cit. 2023-11-27]. Dostupné z: https://ixo-jtag.sourceforge.net/#nexys2prog

[30] VAGNER, David. *Návrh HW pro mikrofonní pole s MEMS mikrofony.* Praha, 2023. Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra radioelektroniky. Vedoucí práce Ing. Petr Honzík, Ph.D. Dostupné také z: https://dspace.cvut.cz/handle/10467/111299

[31] RINGSMUTH, David. *Kondenzátorový mikrofon s dělenou pevnou elektrodou*. Praha, 2023. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra radioelektroniky. Vedoucí práce Ing. Petr Honzík, Ph.D. Dostupné také z: `https://dspace.cvut.cz/handle/10467/109281`

Publikace autora související s prací

[1] ŠEDIVÝ, Jan. Komunikace s MEMS mikrofony pomocí FPGA. In: *4. studentský akustický seminář*. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2023, str. 9–10. ISBN 978-80-01-07156-4.

[2] ŠEDIVÝ, Jan a David VAGNER. Spherical MEMS microphone array with FPGA communication unit. In: *Proceedings of the International Student Scientific Conference Poster – 27/2023*. Czech Technical University in Prague, Faculty of Electrical Engineering, 2023, str. 57–59. ISBN 978-80-01-07140-3.

Přílohy

Příloha A

Seznam použitých zkratek

atd.	a tak dále
log.	logická (hodnota)
např.	například
tj.	to jest
tzn.	to znamená
tzv.	takzvaně
`bash`	Bourne-Again Shell
`gcc`	GNU Compiler Collection
`sdcc`	Small Device C Compiler
`sed`	Stream Editor
ALSA	Advanced Linux Sound Architecture
CMOS	Complementary Metal Oxide Semiconductor
EEPROM	Electrically Erasable Programmable Read-Only Memory
FEL	Fakulta elektrotechnická
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GND	Ground
GNU	GNU's Not Unix
GPL	General Public License
ID	Identifikátor
LDO	Low-Dropout
LED	Light Emitting Diode
LGPL	Lesser General Public License
LSB	Least Significant Bit
MEMS	Micro Electro-Mechanical System
MOSFET	Metal Oxide Semiconductor Field Effect Transistor

MSB	Most Significant Bit
PC	Personal Computer
PID	Product Identifier
RAM	Random Access Memory
SMD	Surface Mount Device
SOIC	Small Outline Integrated Circuit
SOT	Small-Outline Transistor
SVF	Serial Vector Format
UCF	User Constraints File
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VID	Vendor Identifier
VQFP	Very Thin Quad Flat Package
WAV	Waveform Audio File Format

Zkratky týkající se komunikačních rozhraní

Audio rozhraní

FSYNC	Frame Synchronization
I²S	Inter-Integrated Circuit Sound
LRCLK	Left-Right Clock
SCLK	Serial Clock
SD	Serial Data
TDM	Time Division Multiplex
WSO	Word Select Output
WS	Word Select

JTAG

JTAG	Joint Test Action Group
TAP	Test Access Port
TCK	Test Clock
TDI	Test Data Input
TDO	Test Data Output
TMS	Test Mode Select

SPI

CLK	Clock
CS	Chip Select
MISO	Master Input-Slave Output
MOSI	Master Output-Slave Input
SPI	Serial Peripheral Interface

Ostatní rozhraní

I²C	Inter-Integrated Circuit
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

Použité jednotky

zkratka	**název**	**veličina**
A	*ampér*	elektrický proud
b	*bit*	množství informace
B	*bajt* (*byte*) = 8 bitů	množství informace
b/s	*bity za sekundu*	přenosová rychlost
F	*farad*	elektrická kapacita
Hz	*hertz*	kmitočet
Ω	*ohm*	elektrický odpor
V	*volt*	elektrické napětí

Příloha B

Obsah elektronické přílohy

`FPGA_MicArrayBoard_KiCAD.zip` ..... Návrh desky MicArrayBoard jakožto projekt v programu KiCAD zabalený do archivu

`mereni.zip` ..... Archiv se soubory vztahujícími se k měření mikrofonů

- `drzak_DPS/` ..... Složka s návrhem 3D modelu držáku desky
  - `drzak_DPS.png`
  - `drzak_DPS_podlozka.stl`
  - `drzak_DPS.py`
  - `drzak_DPS.stl`
- `recording-tools/` ..... Skripty pro kruhové mikrofonní pole
  - `krokac.py`
  - `LICENSE`
  - `mic_test.py`
  - `reference.py`

`DIP.zip` ..... Pracovní prostor zabalený do archivu

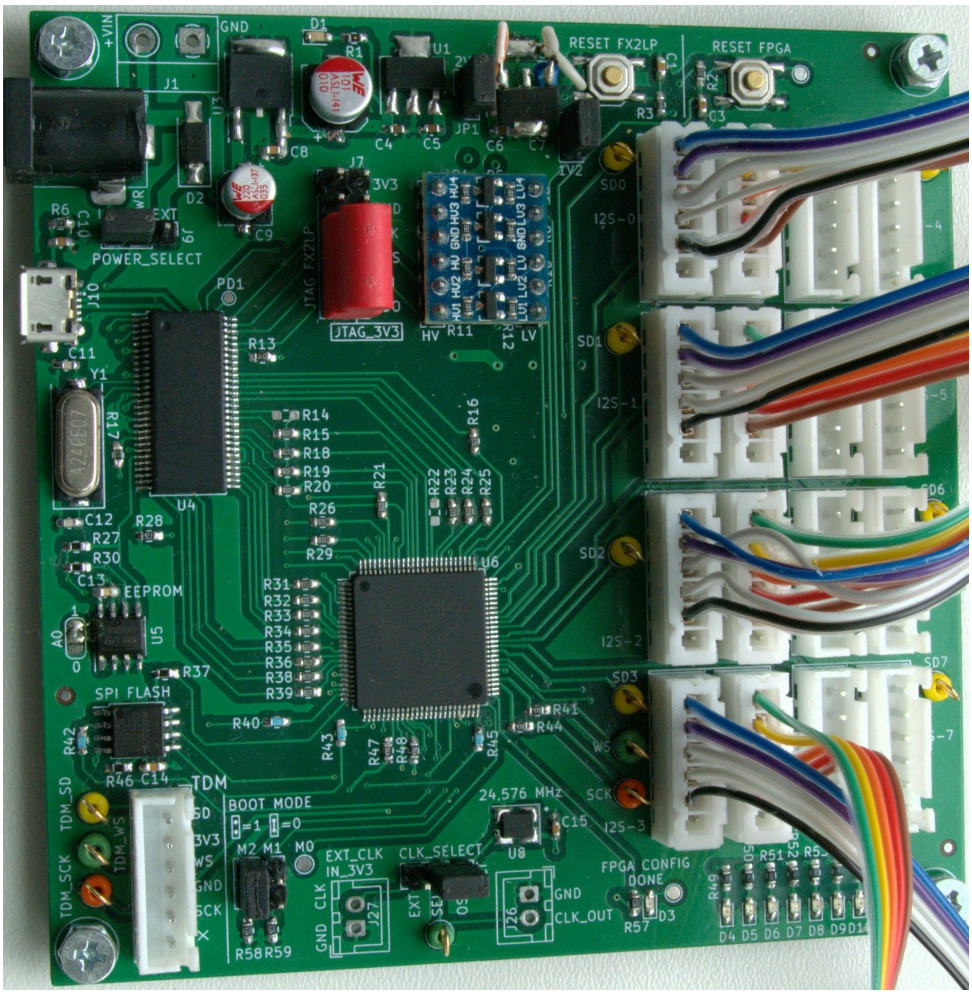
- `Makefile`
- `fx2lib/`
- `FX2LP_audio_ISO/`
  - `descriptor.a51`
  - `FX2LP_audio.c`
  - `.gitignore`
  - `LICENSE`
  - `Makefile`
  - `.git/`
- `MicArrayBoard_I2S_to_USB/`
  - `MicArrayBoard_I2S_to_USB.xise`
  - `MicArrayBoard.ucf`
  - `vhdl/`
    - `hlavni.vhd`
    - `hodiny.vhd`
    - `i2s_multi.vhd`
    - `konstanty.vhd`
    - `usb_fifo.vhd`

DIP.zip ..... Pracovní prostor zabalený do archivu (pokračování)

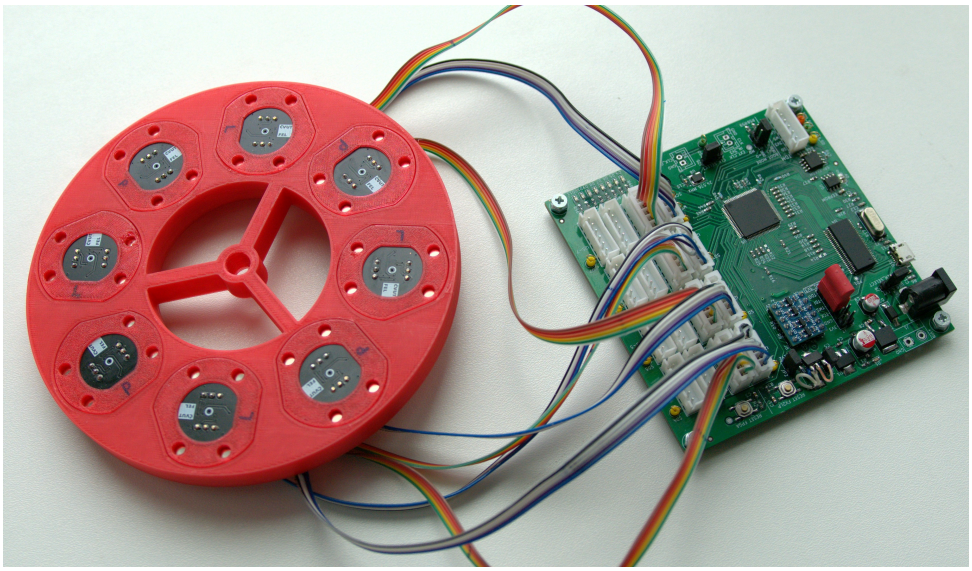
- micarrayboard-scripts/
  - fl.py
  - fx2erase.py
  - .gitignore
  - i2sconfig.sh
  - LICENSE
  - spiflash.py
  - .git/
- MicArrayBoard_SPI_durch/
  - MicArrayBoard_SPI_durch.xise
  - MicArrayBoard.ucf
  - SPI_durch.bit
  - SPI_durch.bit.svf
  - SPI_durch.vhd
- MicArrayBoard_TDM_to_USB/
  - hlavni_tb.wcfg
  - MicArrayBoard_TDM_to_USB.xise
  - MicArrayBoard.ucf
  - vhdl/
    - hlavni.vhd
    - hodiny.vhd
    - konstanty.vhd
    - tdm_multi.vhd
    - usb_fifo.vhd
    - sim/
      - hlavni_tb.vhd
      - prubehy.lua
      - prubehy.vhd
      - tdm_slave.vhd
- MicArrayBoard_TDM_to_USB_simul/
  - hlavni_tb.wcfg
  - MicArrayBoard_TDM_to_USB.xise
  - MicArrayBoard.ucf
  - vhdl/
    - hlavni.vhd
    - hodiny.vhd
    - konstanty.vhd
    - tdm_multi.vhd
    - usb_fifo.vhd
    - sim/
      - hlavni_tb.vhd
      - prubehy.lua
      - prubehy.vhd
      - tdm_slave.vhd

Příloha C

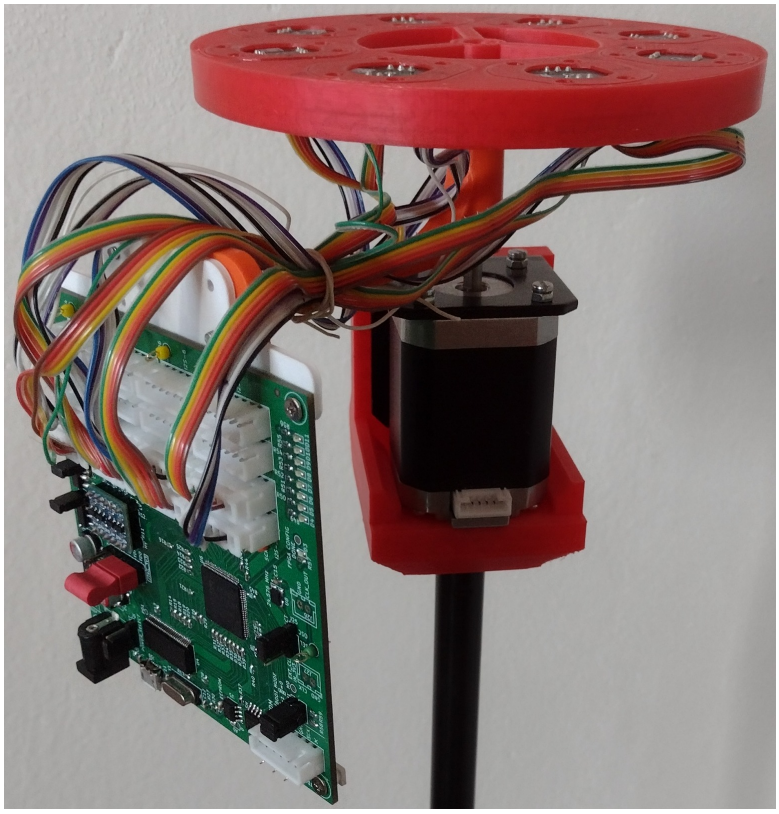
Fotografie



Obrázek C.1: Fotografie osazené desky z vrchní strany



Obrázek C.2: Fotografie desky spolu s kruhovým mikrofonním polem



Obrázek C.3: Fotografie desky MicArrayBoard uchycené na krokovém motoru

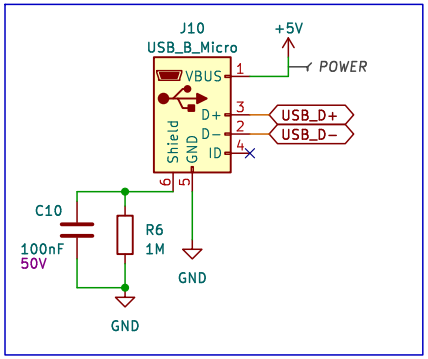
Příloha D

Schéma desky FPGA MicArrayBoard

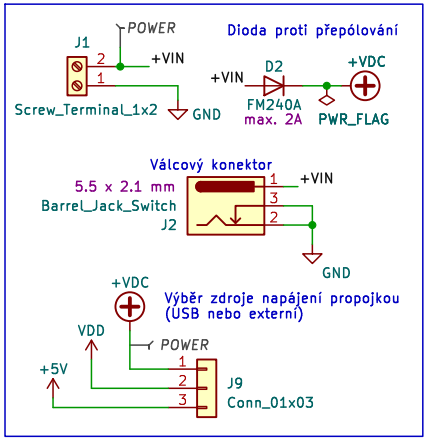
Členění schématu:

- List 1: Napájení
- List 2: USB FX2LP
- List 3: FPGA Spartan-3E
- List 4: Mikrofony

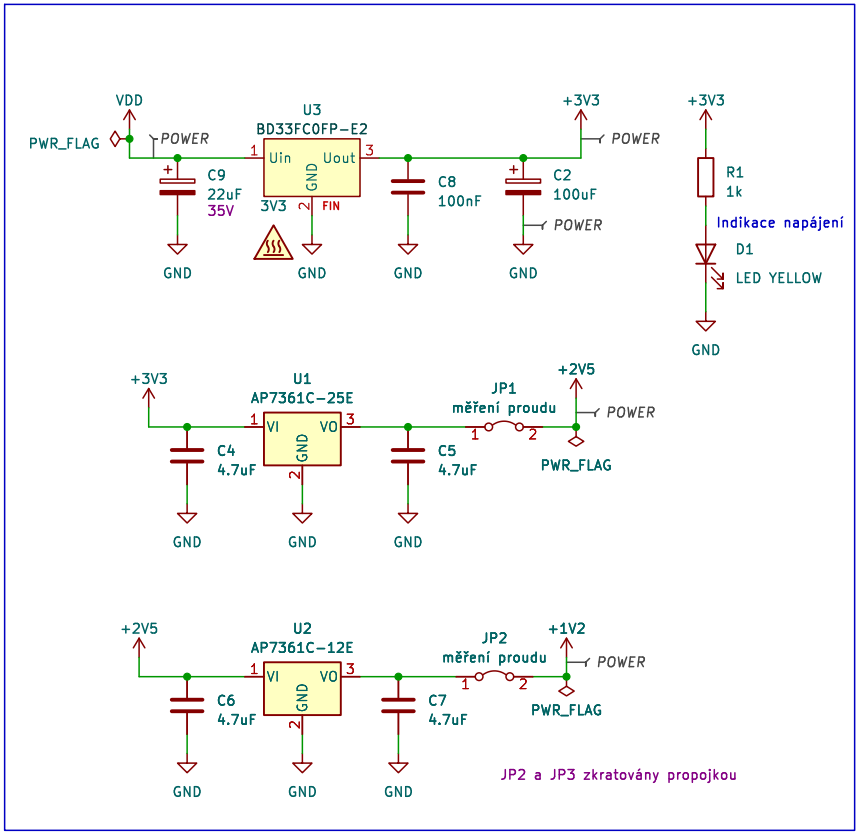
Konektor mikroUSB



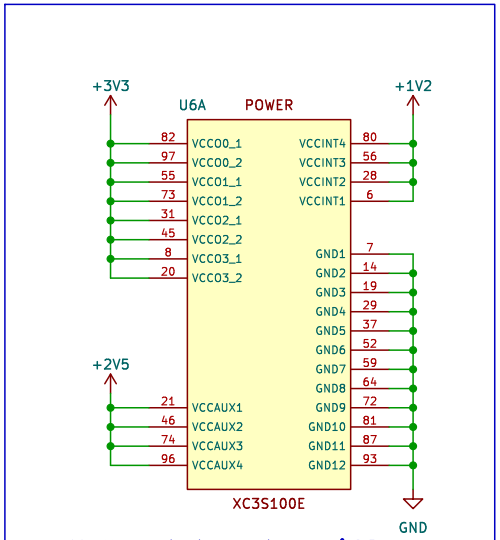
Externí napájení



Regulátory napětí

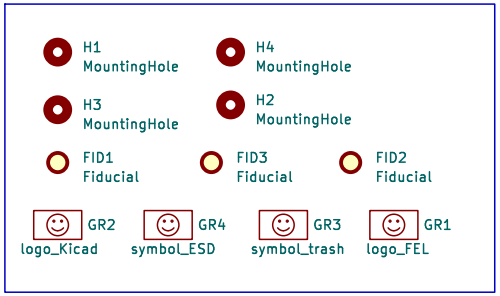


Napájení FPGA

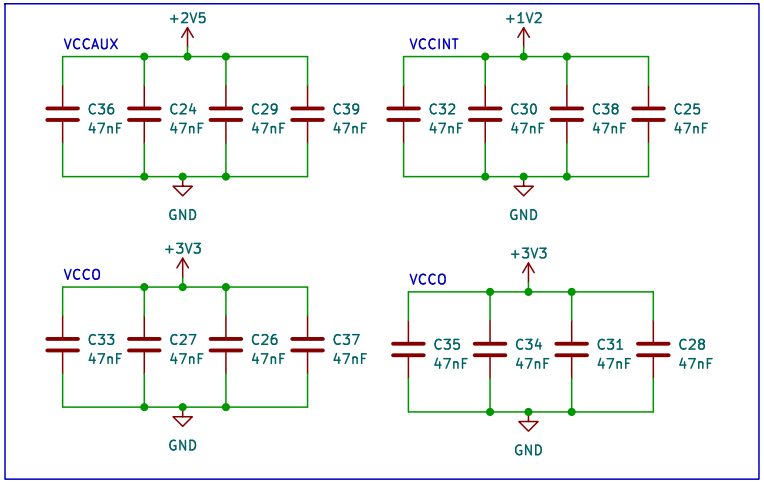


VCCAUX – napájení pomocných obvodů 2,5V
VCCINT – napájení vnitřního jádra 1,2V
VCCO – napájení výstupů v rámci banky, zvoleno 3,3V (pro všechny)

Mechanické + grafické prvky



Blokovací kondenzátory FPGA

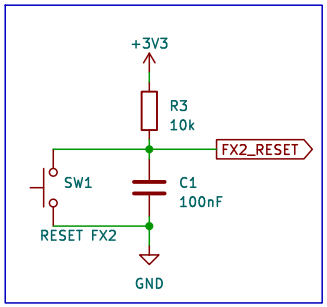


Sheet: /Napájení/
File: napajeni.kicad_sch

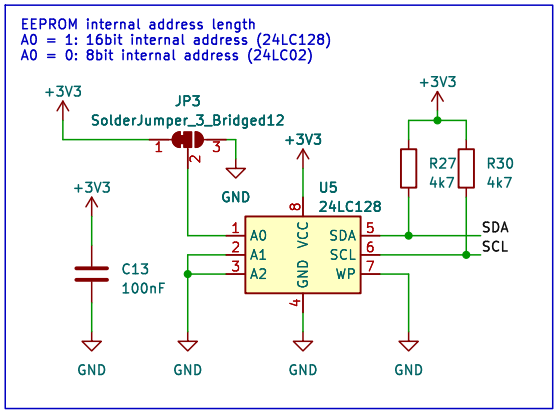
Title: Napájení

Size: A4	Date: 2023-04-23	**Rev: 1**
KiCad E.D.A. kicad 7.0.8-7.0.8~ubuntu22.04.1		Id: 2/5

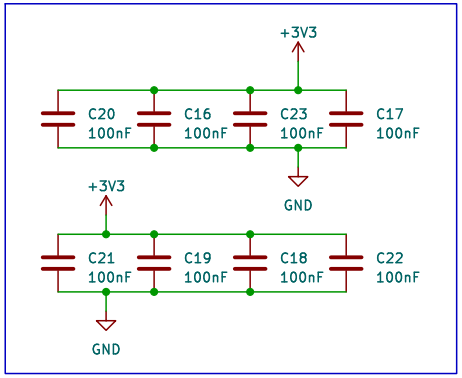
RESET FX2LP



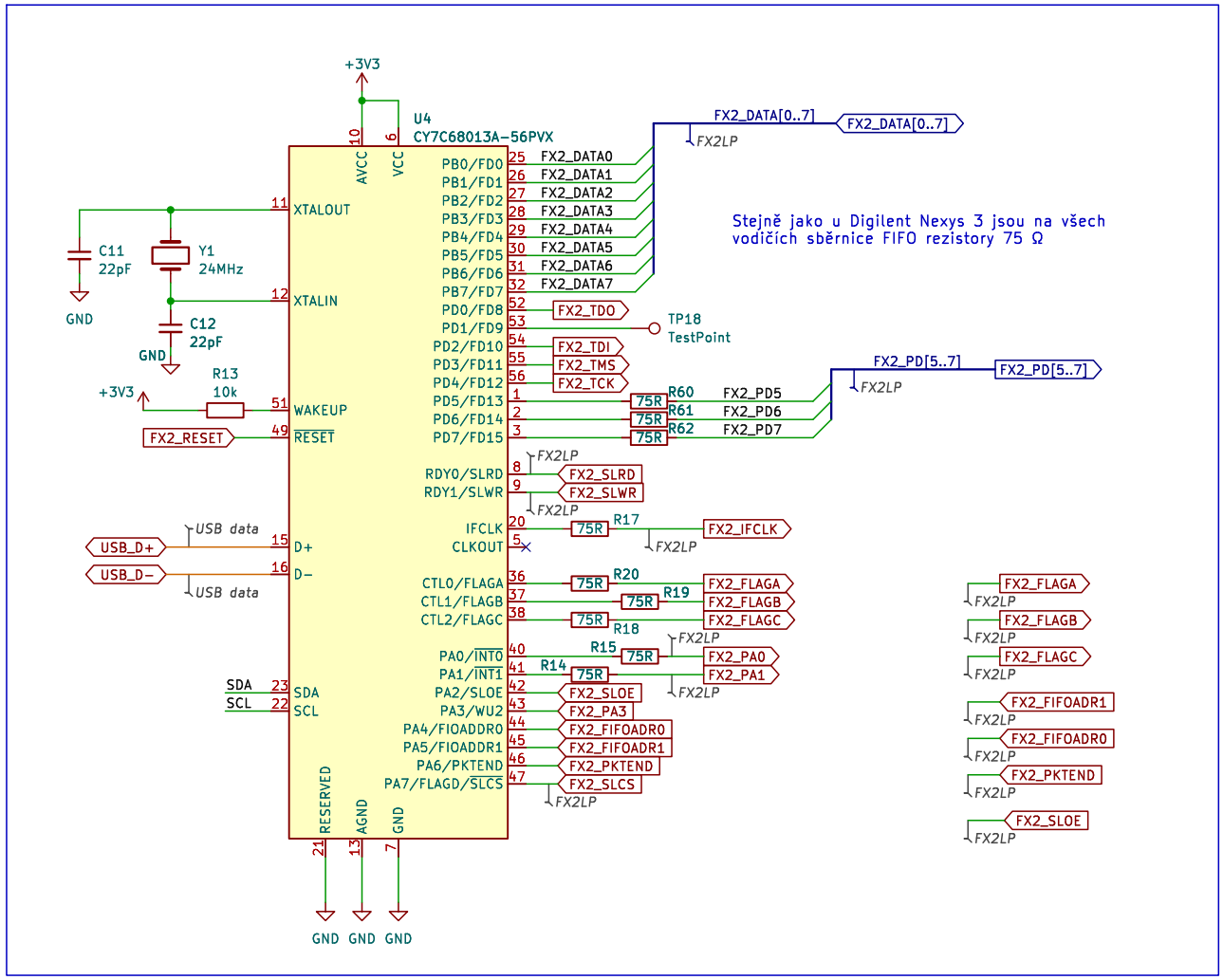
I2C EEPROM



Blokovací kondenzátory FX2LP

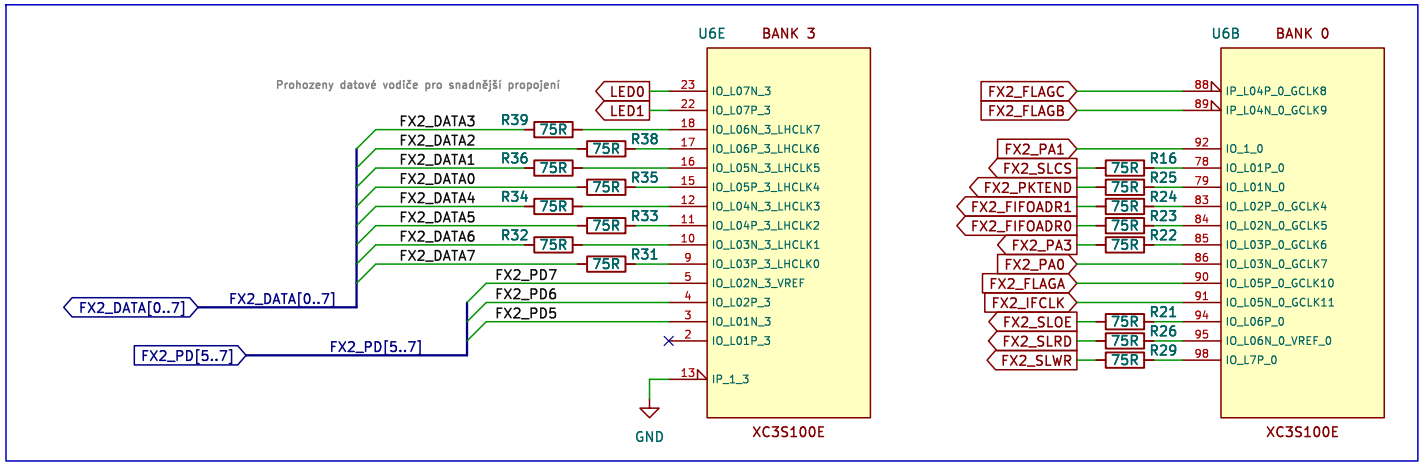


USB mikrořadič FX2LP

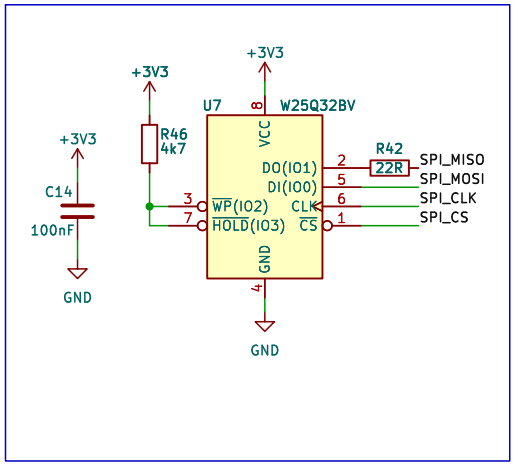


Sheet: /USB_FX2LP/		
File: USB_FX2LP.kicad_sch		
Title: USB FX2LP		
Size: A4	Date: 2023-04-23	**Rev: 1**
KiCad E.D.A. kicad 7.0.8-7.0.8~ubuntu22.04.1		Id: 3/5

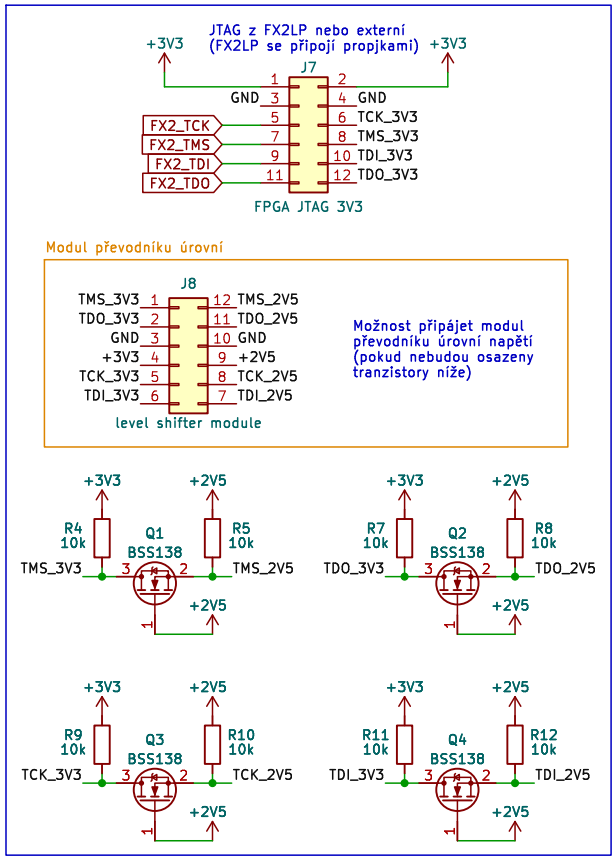
Propojení s FX2LP



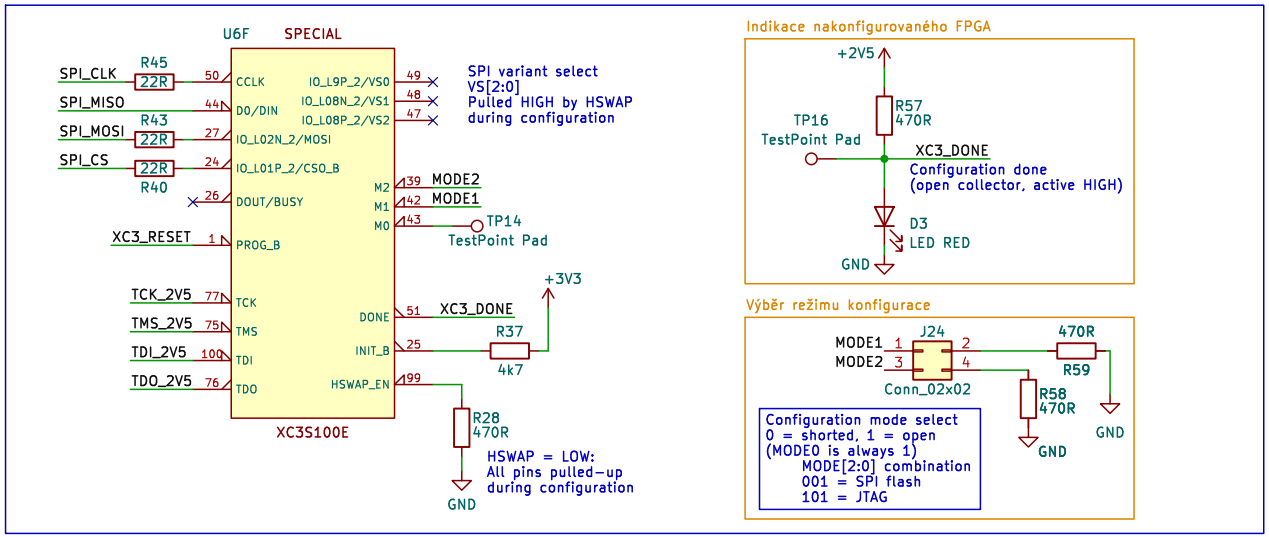
SPI FLASH pro FPGA



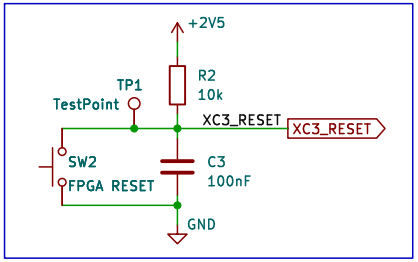
JTAG + převod úrovní napětí



FPGA – rozhraní konfigurace



Reset FPGA

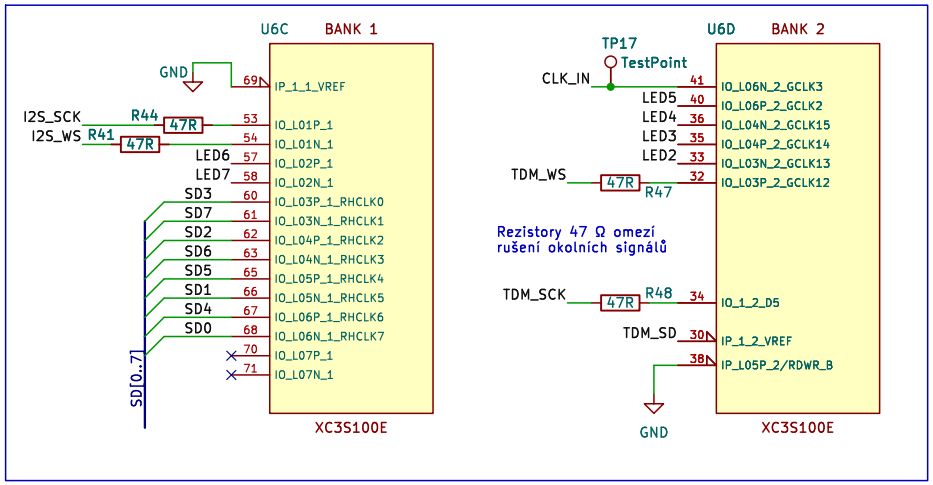


Sheet: /FPGA/
File: FPGA.kicad_sch

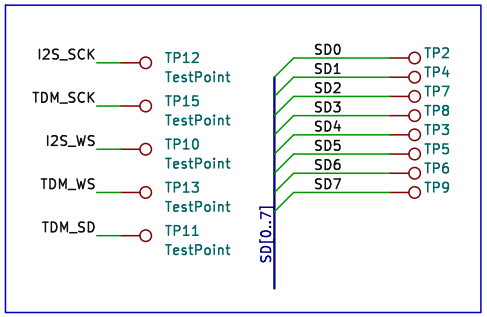
Title: FPGA Spartan-3E

Size: A4	Date: 2023-04-23	**Rev: 1**
KiCad E.D.A. kicad 7.0.8-7.0.8~ubuntu22.04.1		Id: 4/5

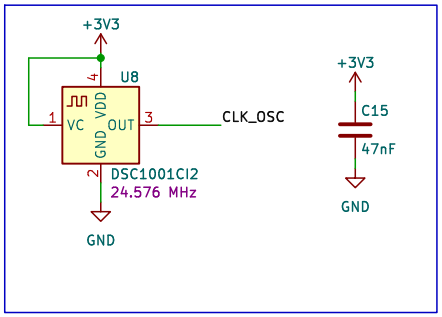
Mikrofonní rozhraní FPGA



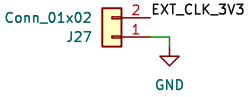
Testovací body



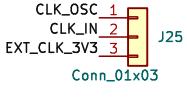
Oscilátor 24,576MHz



Vstup externího zdroje kmitočtu (s úrovní 3,3V)

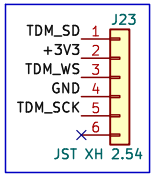


Výběr vstupního zdroje hodin propojkou

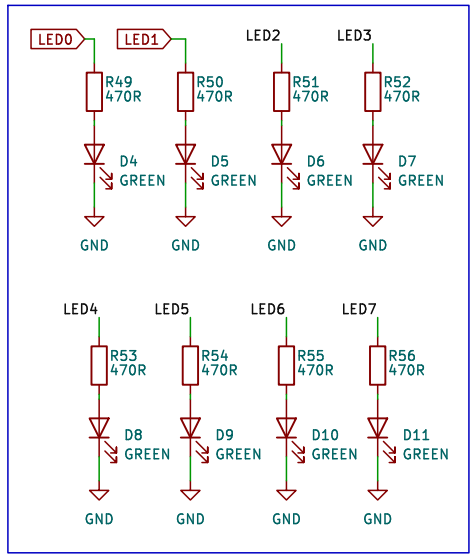


Výstup hodin

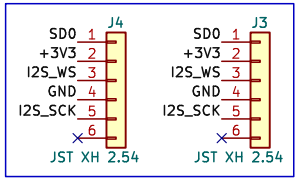
TDM



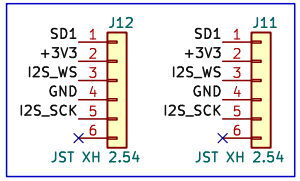
Signalizační LED



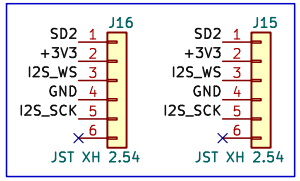
I2S-0



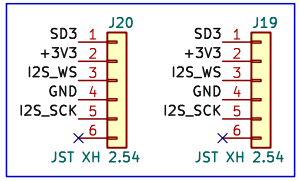
I2S-1



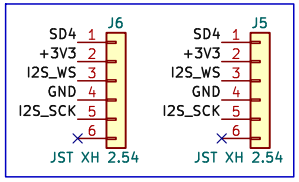
I2S-2



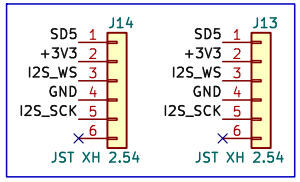
I2S-3



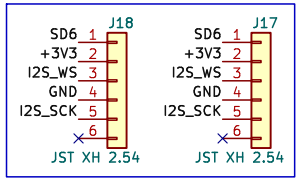
I2S-4



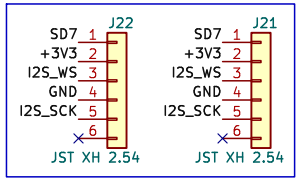
I2S-5



I2S-6



I2S-7



Každý I2S kanál má 1 JST konektor

Sheet: /Mikrofony/		
File: mikrofony.kicad_sch		
Title: Mikrofony		
Size: A4	Date: 2023-04-23	**Rev: 1**
KiCad E.D.A. kicad 7.0.8-7.0.8~ubuntu22.04.1		Id: 5/5

Příloha E

Rozpiska součástek pro desku FPGA MicArrayBoard

Tabulka E.1: Rozpiska součástek pro desku FPGA MicArrayBoard

Symbol součástky	Typ	Hodnota	Popis	Počet	Pouzdro
C1, C3, C8, C10, C13, C14, C16, C17, C18, C19, C20, C21, C22, C23	kondenzátor	100 nF ±10 %	Kondenzátor SMD keramický; 50 V; X7R	14	SMD 0603
C2	kondenzátor	100 µF	Kondenzátor SMD elektrolytický; 10 V	1	Průměr: 6,3 mm
C4, C5, C6, C7	kondenzátor	4,7 µF −20/+80 %	Kondenzátor SMD keramický; 10 V; Y5V	4	SMD 0603
C9	kondenzátor	22 µF	Kondenzátor SMD elektrolytický; 35 V	1	Průměr: 5 mm
C11, C12	kondenzátor	22 pF ±1 %	Kondenzátor SMD keramický; 50 V; C0G (NP0)	2	SMD 0603
C15, C24, C25, C26, C27, C28, C29, C30, C31, C32, C33, C34, C35, C36, C37, C38, C39	kondenzátor	47 nF ±10 %	Kondenzátor SMD keramický; 50 V; X7R	17	SMD 0603
D1	LED	-	Svítivá dioda – žlutá	1	SMD 0603
[α]D2	FM240A	2 A, 40 V	Schottkyho dioda, úbytek 500 mV při 1 A	1	SMA (DO-214AC)
	SS210	2 A, 100 V	Schottkyho dioda, úbytek 650 mV při 1 A	1	SMB (DO-214AA)
D3	LED	-	Svítivá dioda – červená	1	SMD 0603
D4, D5, D6, D7, D8, D9, D10, D11	LED	-	Svítivá dioda – zelená	8	SMD 0603
J1	svorkovnice šroubovací	2 póly	Svorkovnice šroubovací; rozteč 5,08 mm; 2 póly	1	THT
J2	CON-SOCJ-2155	5,5×2,1 mm	Souosý válcový konektor (Jack); vnitřní průměr 2,1 mm; vnější průměr 5,5 mm; samice	1	THT
J3, J4, J5, J6, J11, J12, J13, J14, J15, J16, J17, J18, J19, J20, J21, J22, J23	B6B-XH-A	2,5 mm	Konektor JST XH; rozteč 2,5 mm; 6 kontaktů	17	THT
[β]J7	kolíková lišta	2×6	Kolíková lišta vertikální; rozteč 2,54 mm; 2×6 kolíků	1	THT
[γ]J8	modul	-	Modul převodníku napěťových úrovní; 4 kanály	1	THT
[δ]J9, J25	kolíková lišta	1×3	Kolíková lišta vertikální; rozteč 2,54 mm; 1×3 kolíky	2	THT
J10	ZX62D-B-5PA8(30)	-	Zásuvka USB B micro vodorovná; SMD; ukotvení THT	1	THT
[ϵ]J24	kolíková lišta	2×2	Kolíková lišta vertikální; rozteč 2,54 mm; 2×2 kolíky	1	THT
J26, J27	B2B-XH-A	2,5 mm	Konektor JST XH; rozteč 2,5 mm; 2 kontakty	2	THT
JP1, JP2	kolíková lišta	1×2	Kolíková lišta vertikální; rozteč 2,54 mm; 1×2 kolíky	2	THT
JP3	pájitelná propojka	-	Pájitelná propojka o dvou polohách – zapájet do polohy „1“	1	SMD
[γ]Q1, Q2, Q3, Q4	BSS138-TP	N-kanál	MOSFET s kanálem typu N; U_{DS} 50 V; I_D 220 mA; U_{GSth} 800 mV	4	SOT-23-3
R1	rezistor	1 kΩ	SMD rezistor; tolerance 1 %	1	SMD 0603

Tabulka pokračuje na následující stránce.

Tabulka E.1: Rozpiska součástek pro desku FPGA MicArrayBoard (pokračování)

Symbol součástky	Typ	Hodnota	Popis	Počet	Pouzdro
[γ]R2, R3, R4, R5, R7, R8, R9, R10, R11, R12, R13	rezistor	10 kΩ	SMD rezistor; tolerance 1 %	11	SMD 0603
R6	rezistor	1 MΩ	SMD rezistor; tolerance 1 %	1	SMD 0603
R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R29, R31, R32, R33, R34, R35, R36, R38, R39, R60, R61, R62	rezistor	75 Ω	SMD rezistor; tolerance 1 %	25	SMD 0603
R27, R30, R37, R46	rezistor	4,7 kΩ	SMD rezistor; tolerance 1 %	4	SMD 0603
R28, R49, R50, R51, R52, R53, R54, R55, R56, R57, R58, R59	rezistor	470 Ω	SMD rezistor; tolerance 1 %	12	SMD 0603
R40, R42, R43, R45	rezistor	22 Ω	SMD rezistor; tolerance 1 %	4	SMD 0603
R41, R44, R47, R48	rezistor	47 Ω	SMD rezistor; tolerance 1 %	4	SMD 0603
SW1, SW2	tlačítko	-	Tlačítko SMD kovové; 4 vývody; 5,2×5,2×2 mm	2	SMD
TP2, TP3, TP4, TP5, TP6, TP7, TP8, TP9, TP10, TP11, TP12, TP13, TP15, TP17	testovací bod	-	Testovací bod; drátová smyčka na skleněné průchodce	14	THT
U1	AP7361C-25E-13	2,5 V ±1 %	Regulátor napětí 2,5 V; LDO, lineární; 1 A; úbytek napětí 0,44 V	1	SOT223
[α]U2	AP7361-12E-13	1,2 V ±1 %	Regulátor napětí 1,2 V; LDO, lineární; 1 A; úbytek napětí 0,71 V	1	SOT223
	AP2114HA-1.2TRG1	1,2 V ±1,5 %	Regulátor napětí 1,2 V; LDO, lineární; 1 A; úbytek napětí 1,2 V	1	SOT223
U3	BD33FC0FP-E2	3,3 V ±1 %	Regulátor napětí 3,3 V; LDO, lineární; 1 A; úbytek napětí 0,4 V	1	DPAK
U4	CY7C68013A-56PVXCT	-	USB mikrořadič FX2LP	1	SSOP-56
U5	M24128-BRMN6TP	16 kB	Paměť EEPROM; I^2C rozhraní; 16 k×8 bit	1	SO8
U6	XC3S100E-4VQG100C	-	FPGA; 100 k systémových hradel; 2160 ekvivalentních logických buněk	1	VQFP-100
U7	W25Q32BV	4 MB	Paměť Flash NOR, SPI rozhraní, 32 Mbit	1	SOIC-8 208 mil
U8	DSC1001CI2-024.5760	24,576 MHz ±25 ppm	Oscilátor; stabilita 25 ppm; 3,2×2,5 mm	1	CDFN-4
Y1	ABLS-24.000MHZ-D-R60-1-W-T	24 MHz ±25 ppm	Rezonátor krystalový SMD; 24 MHz; ±25 ppm; 18 pF	1	HC-49/US

[α] Lze zvolit součástku ze dvou alternativ.
[β] K této liště přísluší čtyři zkratovací propojky (jumpery).
[γ] Pokud je osazen modul J8, neosazují se tranzistory Q1, Q2, Q3, Q4 ani rezistory R4, R5, R7, R8, R9, R10, R11 a R12.
[δ] Ke každé této liště přísluší jedna zkratovací propojka (jumper). Případně lze, vzhledem k její funkci, tuto lištu nahradit dvoupolohovým přepínačem.
[ϵ] K této liště přísluší dvě zkratovací propojky.