VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

OVLÁDÁNÍ VESTAVĚNÉHO SYSTÉMU PŘES INTERNET

CONTROL OF EMBEDDED SYSTEM THROUGH INTERNET

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAKUB HORÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. DOBAI ROLAND, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Horák Jakub**
Obor: Informační technologie
Téma: **Ovládání vestavěného systému přes Internet**
Control of Embedded System Through Internet
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s problematikou ovládaní vestavěných systémů přes Internet.
2. Navrhněte a implementujte systém pro ovládání vestavěného systému na bázi Xilinx Zynq přes Internet.
3. Ověřte funkčnost implementovaného systému pro jednoduché úlohy ovládání vybraných prvků, které jsou dostupné na vývojové desce.
4. Zhodnoťte dosažené výsledky a diskutujte možnosti dalšího vývoje projektu.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodu 1 zadání, demonstrace rozpracovanosti bodu 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Dobai Roland, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Cílem této této bakalářské práce je umožnit běžnému uživateli ovládat vestavěný systém na bázi Xilinx Zynq přes Internet. Vestavěný systém bude obsahovat audio kodek ADAU1761. V práci popíši co je nutné pro připojení daného zařízení na platformě Xilinx Zynq. Poté budou popsány možnosti ovládání zařízení přes Internet a následně popsán návrh a implementace aplikace, která umožní uživateli interagovat se zařízením.

Abstract

The goal of this bachelor thesis is to enable a regular user to control the embedded system on Xilinx Zynq, which will include audio codec ADAU1761. In the thesis I will describe what is required to connect the device to the Xilinx Zynq platform. After that, the device's control through Internet will be described, followed by the design and implementation of an application that will allow the user to interact with the device.

Klíčová slova

vestavěné systémy, Xilinx, FPGA, Zynq, Zedboard, ADAU1761, operační systém, Linux, Linaro-Ubuntu, cherrypy, ALSA, Python

Keywords

embedded systems, Xilinx, FPGA, Zynq, Zedboard, ADAU1761, operating system, Linux, Linaro-Ubuntu, cherrypy, ALSA, Python

Citace

HORÁK, Jakub. *Ovládání vestavěného systému přes Internet*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Roland Dobai.

Ovládání vestavěného systému přes Internet

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Roland Dobai, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Horák
17. května 2017

Poděkování

Rád bych poděkoval panu Ing. Roland Dobai, Ph.D., vedoucímu mé bakalářské práce, za důvěru, vedení a podnětné rady.

Obsah

Slovník

AD – Analog Devices – výrobce audio kodeku.

- **ALSA** – Advanced Linux Sound Architecture – aplikační rozhraní pro zvukové karty v Linuxu.
- **API** – Application Programming Interface – aplikační rozhraní.
- **ARM** – Advanced RISC Machine – architektura procesorů.
- **AXI** – Advanced eXtensible Interface – rozhraní pro komunikaci s periferiemi.
- **BRAM** – Block RAM – konfigurovatelný paměťový blok.
- **CPU** – central processing unit – centrální procesorová jednotka.
- **CSS** – Cascading Style Sheets – jazyk pro popis způsobu zobrazení HTML elementů.
- **DDR, DDR2, DDR3, LPDDR2** – double data rate – typy operační paměti.
- **FPGA** – Field Programmable Gate Array – integrovaný obvod, který může být přeprogramován.
- **FS** – File system – kořenový souborový systém.
- **FSBL** – First Stage Boot Loader – zavaděč prvního stupně.
- **GCC** – GNU Compiler Collection – sada překladačů vytvořených v rámci projektu GNU.
- **GUI** – Graphical User Interface – grafické uživatelské rozhraní.
- **GPL** – General Public License – všeobecná veřejná licence GNU.
- **HDMI** – High-Definition Multimedia Interface – všeobecná veřejná licence GNU.
- **HTML** – Hypertext MarkUp Language – značkovací jazyk.
- **HTTP** – Hypertext Transfer Protocol – aplikační protokol pro přenos hypertextových dokumentů a obrázků.
- **I^{2}C** – Inter-Integrated Circuit – programovatelná logika.
- **IDE** – Integrated Development Environment – vývojové prostředí.
- **IP core** – Intellectual Property core – blok popisující určitou funkcionalitu v FPGA.
- **LED** – Light-Emitting Diode – světlo vyzařující dioda.
- **OLED** – Organic light-emitting diode – displeje využívající technologii LED.
- **OS** – Operating system – operační systém.
- **PL** – Programmable logic – programovatelná logika.
- **PS** – Processing System – procesorový systém.

- **SSH** – Secure Shell – Aplikační protokol pro zabezpečnou komunikaci.
- **SoC** – System on Chip – systém na čipu.
- **SPI** – Serial Peripheral Interface – sériové periferní rozhraní.
- **TCP** – Transmission Control Protocol – transportní protokol používaný pro spolehlivé spojení.
- **TCP/IP** – Transmission Control Protocol/Internet Protocol – rodina standartních síťových protokolů.
- **U-Boot** – Das U-boot – zavaděč druhého stupně.
- **UART** – Universal Asynchronous Receiver and Transmitter – asynchronní sériové rozhraní.
- **URL** – Uniform Resource Locator – jednotná adresa zdroje.
- **XSDK** – Xilinx Software Development Kit – IDE pro vývoj softwaru.

Kapitola 1

Úvod

Vestavěný systém je jednoúčelový systém, který je předem navržen pro konkrétní úlohy. Narozdíl od osobních počítačů, které mohou sloužit pro obecné úlohy [1]. Jelikož je požadovaný účel předem znám, lze použít pouze ten hardware, který je nutný pro dané úlohy a tím snížit náklady na výrobu.

Mezi vestavěné systémy můžeme řadit také platformu Zynq, jejíž základní charakteristikou je kombinace dvoujádrového procesoru spolu s programovatelným hradlovým polem (*Field Programmable Gate Array*, dále jen FPGA). Díky flexibilitě a škálovatelnosti FPGA jsou tyto zařízení předurčena pro použití v široké škále projektů. Také umožňuje vývojářům zvolit, které součásti bude návrh obsahovat a tím se zaměřit jak na nízkonákladové, tak i vysoce výkonné aplikace.

Tato práce se bude zabývat návrhem a implementací aplikace pro ovládání zvukové karty přes Internet. Zvuková karta bude součástí vestavěného systému na bázi Xilinx Zynq.

Cílem této práce je vytvořit aplikaci pro ovládání zvukové karty. Aplikace by měla běžnému uživateli zpřístupňovat základní nastavení, umožňovat jejich změnu a následně provádět jednoduchou demonstraci této změny.

V první části Kapitoly 2 představím platformu Zynq, následovat bude popis aktuálního stavu použití zvukové karty na této platformě. V Kapitole 3 nastíním důležité body pro úspěšné spojení mikročipu Zynq s audio kodekem ADAU1761 a následné nasazení vestavěného Linuxu. Zbytek kapitoly se bude věnovat návrhu a možnosti implementace aplikace, která bude dané zařízení ovládat přes Internet. V Kapitole 4 popíši konkrétní implementaci aplikace, dle návrhu z předcházející kapitoly. Závěrečná Kapitola 5 bude pojednávat o výsledcích mé práce a o funkčnosti aplikace.

Kapitola 2

Vestavěný systém na platformě Xilinx Zynq

V první části této kapitoly bude představena platforma Zynq, kde budou popsány její části a nástroje používané pro vývoj projektů na této platformě. V druhé časti se zaměřím nad možnostmi ovládání vestavěného systému. Poslední část bude pojednávat o stávajícím řešení možnosti ovládání zvukových zařízení a možností jeho vylepšení.

2.1 Platforma Zynq

Konkrétně mikročip *Xilinx Zynq-7000 All programable SoC*, kde termínem SoC (*System on Chip*) je označována technologie integrace více typů obvodů na jednom čipu. V tomto případě se jedná o spojení procesorového systému (*Processing System*, dále jen PS) obsahujícího dvoujádrový procesor ARM Cortex-A9 a programovatelnou logiku (*Programmable logic*, dále jen PL). Náhled struktury mikročipu lze vidět na Obrázku 2.1, kde je zobrazeno rozdělení mikročipu na PS a PL.

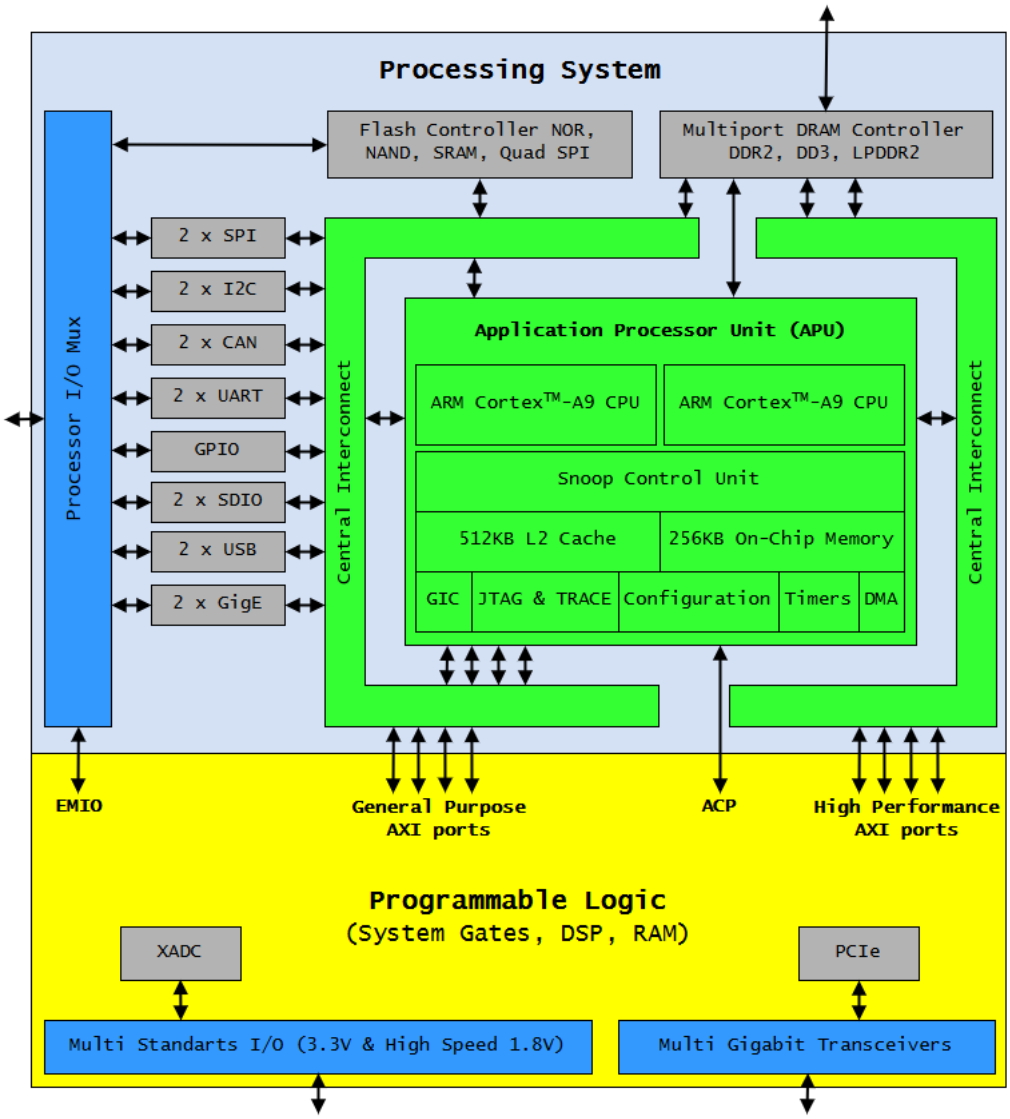
2.1.1 Procesorový systém

Na Obrázku 2.1 PS zabírá dominantní část. Mezi jeho nejdůležitější části patří jádro procesoru (*Application Processor Unit*) a propojovací můstek (*Central Interconnect*). Můstek slouží pro propojení a alokaci adresových prostorů jednotlivých rozhraní.

Mezi další důležité periférii patří SD/SDIO, která zajišťuje komunikaci s SD kartou. SD karta může být použita jako napěťově nezávislé, externí paměťové uložiště nebo z ní může systém nahrávat instrukce i konfiguraci PL. Posledně zmíněné možnosti se využívá při použití operačního systému.

Jako operační paměť procesoru lze použít interní BRAM (*Block Random-access memory*) paměť nebo externí paměť. Při užití interní paměti můžeme pomocí jednotky paměťového rozhraní obsahující DDR (*Double Data Rate*) kontroler, zvolit jeden ze standartů DDR2, DDR3 nebo LPDDR2.

Další možností připojení externích zařízení umožňuje PS pomocí jednotky pro synchronní komunikaci SPI (*Serial Peripheral Interface*), sběrnicí I^2C (*Inter-Integrated Circuit*) nebo rozhraním pro asynchronní komunikaci UART (*Universal Asynchronous Receiver and Transmitter*).



Obrázek 2.1: Blokové schéma systému Zynq-7000

2.1.2 Programovatelná logika

Firma Xilinx ve svých materiálech označuje FPGA jako PL, což je integrovaný obvod, který může být přeprogramován po jeho výrobě. Není tedy vyroben pro konkrétní účel a vývojář jej může naprogramovat podle požadované funkce. FPGA je složena z relativně malých bloků programovatelných logických obvodů. Tyhle bloky obsahují typicky několik registrů a několik desítek nizkoúrovňových, konfigurovatelných logických prvků, které jsou uspořádány do matice a propojeny pomocí programovatelných spojení [5].

Komunikace mezi PL a PS probíhá pomocí sběrnice AXI (*Advanced eXtensible Interface*). Se základní verzí sběrnice s velikostí 32 b lze teoreticky při frekvenci 150 MHz dosáhnout přenosové rychlosti pro zápis i čtení 600 MB/s oběma směry. V případě potřeby vyšší přenosové rychlosti můžeme použít vysokorychlostní verzi sběrnice s velikostí 64 b, která má dvojnásobnou přenosovou rychlost oproti základní, tedy zápis i čtení 1 200 MB/s oběma směry.

2.1.3 Vývojové nástroje

Vivado Design Suite (dále jen Vivado) je komplexní vývojové prostředí (dále jen IDE), které je vyvinuto firmou Xilinx pro její platformy. Mezi hlavní funkce patří vytváření blokového návrhu, který se skládá z IP jader (*Intellectual Property Cores*, dále jen IP). IP představuje již vytvořenou hardwarovou komponentu popsanou jazykem Verilog nebo VHDL [2], pokud pro požadovanou funkcionalitu Vivado neobsahuje IP, umožnuje jej vytvořit či naimportovat od třetích stran.

Mezi další funkce patří analýza, implementace, simulace a konečné vygenerování konfiguračního souboru *bitstream* na základě blokového návrhu. Soubor *bitstream* slouží pro naprogramování PL.

Pro vývoj softwaru se na plaftormě používá IDE *Xilinx Software Development Kit* (dále jen XSDK), které je založené na oblíbeném IDE Eclipse. Kromě vývoje softwaru umožňuje vytváření zavaděčů prvního stupně (*First Stage Boot Loader*, dále jen FSBL) a zaváděcích obrazů.

2.1.4 Linux na platfromě Zynq

Linux (oficiálně GNU/Linux) je operační systém (dále jen OS) založený na unixovém jádře. První verzi linuxového jádra naprogramoval Linus Torvalds v roce 1991. Mezi hlavní výhody patří volně dostupné zdrojové kódy Linuxu a všech nástrojů pro jeho tvorbu. Nejdůležitější Linuxové komponenty včetně jádra jsou zpřístupněny pod licencí GPL (*General Public License*). Linux také poskytuje širokou škálu síťových protokolů a nástrojů. Pro Linux byl napsán velký počet volně přístupných aplikací, z kterých si můžeme vybrat. Pokud by však žádná neodpovídala našim požadavkům, můžeme využít početné linuxové komunity, kde s největší pravěpodobností někdo podobný problém řešil nebo řeší [10].

Linux je oficiálně podporovaným OS pro platformu Zynq, na repozitáři firmy Xilinx je k dispozici adresář Xilinx-Linux[1], který vychází z oficiálního repozitáře Linuxu[2]. Oproti tomuto repozitáři však obsahuje specifické soubory jako například upravéné konfigurace jádra, stromy zařízení a ovladače pro platformy Zynq.

2.1.5 Nástroje pro nasazení vestavěného Linuxu

Vytváření a konfigurace vlastního vestaveného Linuxu lze provést ručně pomocí nástrojů, které poskytuje společnost Xilinx na svém repozitáři[3].

Druhou možností je použít nástroje pro konfiguraci, vytvoření a nasazení vestavěného Linuxového systému. Tyto nástroje umožňují upravit zavaděč, jádro Linuxu nebo linuxové aplikace. Přidávat nové jádra, ovladače zařízení, knihovny a aplikace. Narozdíl od ruční konfigurace jsou tyto operace zautomatizované a zahrnují již některé často používané balíčky aplikací. Výhodou jejich použití je urychlení vývoje vestavěného systému.

Yocto

Projekt Yocto je otevřený software poskytující šablony, nástroje a metody pro vytváření vlastních vestavěných systémů založených na Linuxu. Referenčním systémem projektu je Poky, který je složen z kolekce nástrojů a metadat [6]. Poky je nezávislý na platformě

[1] https://github.com/Xilinx/linux-xlnx
[2] http://kernel.org
[3] https://github.com/Xilinx/

a poskytuje křížový překladač, využívá BitBake nástroj, OpenEmbedded-Core a výchozí nastavení metadat.

PetaLinux

Zdarma dostupný nástroj společnosti Xilinx, který slouží pro konfiguraci, vytvoření a nasazení vestavěného linuxového systému na platformy Zynq a MicroBlaze. Jedná se o nádstavbu nad nástroji dostupnými v repozitáři zmíněném na začátku odstavce 2.1.5. PetaLinux navíc nabízí tyhle nástroje:

- Aplikace, ovladače zařízení & generátor knihoven, šablony.
- Agenty pro ladění.
- Zautomatizované nástroje.
- Podporu pro *Xilinx System Debugger*

Narozdíl od projektu Yocto nevytváří PetaLinux linuxovou distribuci, umožnuje integrovat nástroje společnosti Xilinx jako například možnost nalinkovat projekt z programu Vivado a následné vygenerování DTB souboru. Oproti projektu Yocto je však hůře přizpůsobitelný a méně flexibilní.

2.1.6 Ovládání vestavěného Linuxu přes Internet

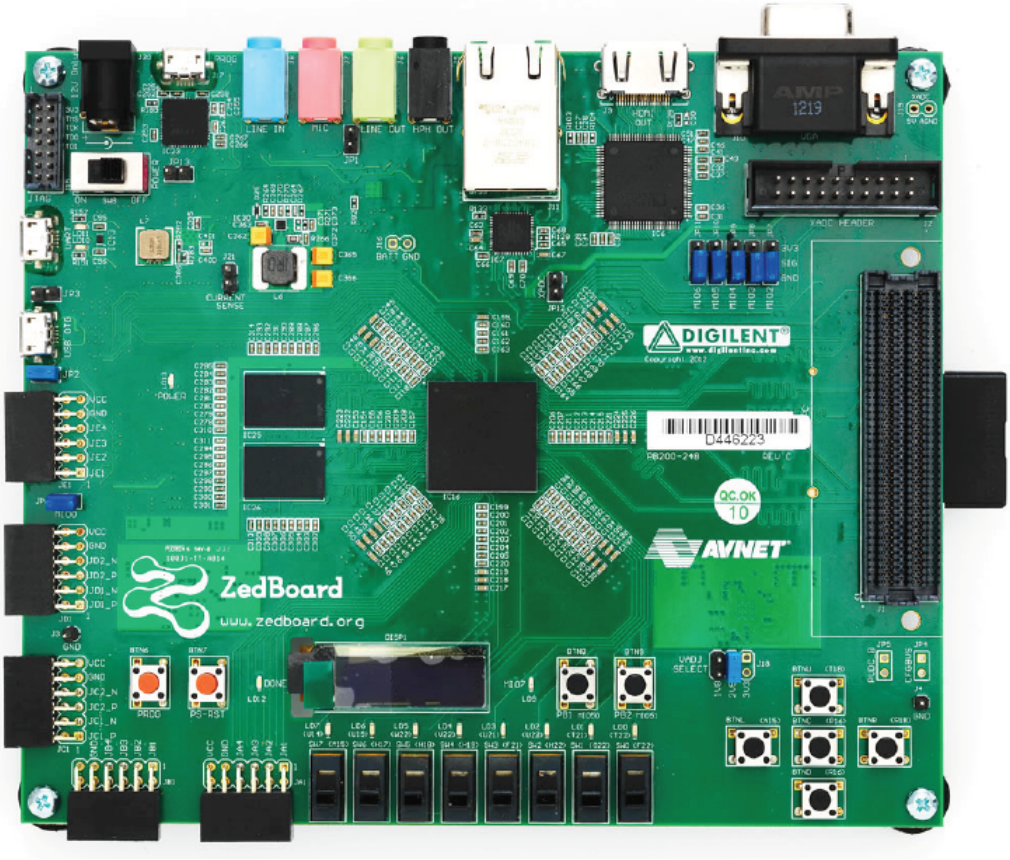
K ovládání systému přes Internet je nutné vytvořit spojení mezi stanicí a klientem, k tomuto účelu obsahuje Linux rodinu standartních síťových protokolů TCP/IP (*Transmission Control Protocol/Internet Protocol*), která se dnes používá k realizaci většiny síťových komunikací. Nejčastěji používanými protokoly pro interakci s vestavěným systémem jsou:

- TCP – Transportní protokol používaný pro spolehlivé spojení mezi dvěmi stanicemi.
- SSH (*Secure Shell*) – Aplikační protokol, který slouží pro zabezpečnou komunikaci. Lze využít pro zpřístupnění příkazového řádku vzdáleného počítače.
- HTTP (*Hypertext Transfer Protocol*) – Aplikační protokol, jehož primárním úkolem je přenos hypertextových dokumentů a obrázků.

2.2 Referenční zařízení

Referenčním zařízením byla zvolena vývojová deska ZedBoard, jež je znázorněna na Obrázku 2.2. ZedBoard disponuje mikročipem Xilinx Zynq®-7000 All Programmable SoC, obsahujícího nízkonákladovou a nízkopříkonovou FPGA řady Artix-7 [8]. Kromě něj je osazena vestavěnými komponentami jako jsou například:

- LED (*Light-Emitting Diode*)
- OLED (*Organic light-emitting diode*) displej
- Tlačítka
- Přepínače



Obrázek 2.2: Referenční zařízení ZedBoard

Pokud je potřeba připojit externí zařízení, lze jej připojit například pomocí konektorů:

- HDMI (*High-Definition Multimedia Interface*)
- VGA (*Video Graphics Array*)
- Audio vstup/výstup
- RJ-45 (*Registered Jack*)

Velký počet rozhraní předurčuje vývojovou desku ZedBoard pro využití v široké škále různých projektů.

2.2.1 Zvuková zařízení

Jako výběr prvku pro ovládání na této platformě jsem si zvolil zvukové zařízení. Pro práci se zvukem obsahuje Zedboard HDMI kodek ADV7511 a audio kodek ADAU1761. Oba kodeky jsou vyrobeny firmou Analog Devices (dále jen AD), která k nim vytvořila ovladače pro OS Linux. Prvně zmíněný kodek nabízí jen omezené možnosti pro práci se zvukem a jeho primární účel je grafický výstup.

ADAU1761 je 24 b audio kodek s integrovaným digitálním zpracováním zvuku, které poskytuje 48 KHz stereo nahrávání i přehrávání. Stereo audio ADC a DAC podporuje vzorkovací frekvenci od 8 KHz do 96 KHz.

Přenos audio dat probíhá přes sériovou sběrnici I^2S (*Inter-IC Sound*). Nastavení kodeku probíhá pomocí zápisu do konfiguračních registrů dostupných přes sériovou sběrnici I^2C.

2.2.2 Stávající řešení

Pro samostatné použití audio kodeku ADAU1761 není dostupný žádný blokový návrh, avšak lze použít část blokového návrhu firmy AD zabývající se prací s HDMI kodekem ADV7511. AD také nabízí na svém repozitáři zdrojové kódy Linuxu, který obsahuje všechny potřebné soubory pro nasazení vestavěného Linuxu s jejich zařízeními. Následnou obsluhu audio kodeku přes internet lze provést přípojením ke stanici pomocí protokolu SSH, který zpřístupní příkazový řádek stanice. Dostupné programy na stanici jsou:

- **Alsamixer** - Grafický program pro správu zvukové karty.
- **aplay** - Přehrávač zvuku.
- **arecord** - Nahrávání zvuku.

Tyto programy využívají linuxovou jadernou komponentu *Advanced Linux Sound Architecture* (dále jen ALSA), která poskytuje aplikační rozhraní (dále jen API) pro zvukové karty.

2.2.3 Cíl práce

Cílem této práce je nasadit vestavěný Linux, který bude umožňovat vzdálenou správu audio kodeku přes internet. Po nasazení se bude práce zabývat návrhem a implementací aplikace, která bude poskytovat přehledné grafické rozhraní pro uživatele, jenž bude mít možnost vidět stav jednotlivých nastavení a dokáže je měnit v dovoleném rozsahu. Kromě toho bude moct přehrávat tóny a používat mikrofón.

2.2.4 Porovnání se stávajícím řešením

Nynější řešení pomocí příkazové řádky není uživatelsky přívětivé a nutí si pamatovat příkazy pro jednotlivé ovládání zařízení. Toto řešení není vhodné pro běžné uživatele, kteří se s příkazovým řádkem ještě nikdy nesetkali, z hlediska bezpečnosti není žádoucí, umožnit uživatelům přístup do systému. Výsledné řešení by oproti tomu mělo odstínit uživateli všechny součásti systému, které s audio kodekem nesouvisí a umožnit jej kontrolovat i uživatelům se základní znalostí počítačů.

Kapitola 3

Návrh vestavěného systému na platformě Zynq

Na počátku kapitoly se nejdříve zaměřím na hardwarovou část vestavěného systému, kde popíši jak připojit audio kodek ADAU1761 k PS, pak se budu zabývat potřebnými kroky pro úspěšné spuštění a nasazení vestavěného Linuxu s tímto hardwarem. Následně se přesunu k samotnému návrhu možností pro vývoj aplikace, která umožní uživateli ovládat daný vestavěný systém přes Internet.

3.1 Blokový návrh

Jak již bylo řečeno v Odstavci 2.2.2, blokový návrh bude vycházet z referenčního návrhu ADV7511 dostupného na repozitáři[1] firmy AD, který obsahuje i chybějící IP jádro sběrnice I^2S ve Vivadu.

Pro importování blokového návrhu je potřeba stáhnout Vivado verze 15.4.2, zvolená verze není nejnovější, avšak je nutné ji použít z důvodu kompatibility s hlavní větví `master` repozitáře.

Na Obrázku 3.1 je zobrazena část blokového návrhu, která umožňuje konfigurovat audiokodek a přenášet data mezi ním a PS. Blokový návrh se skládá z několika důležitých IP jader, která popíši níže.

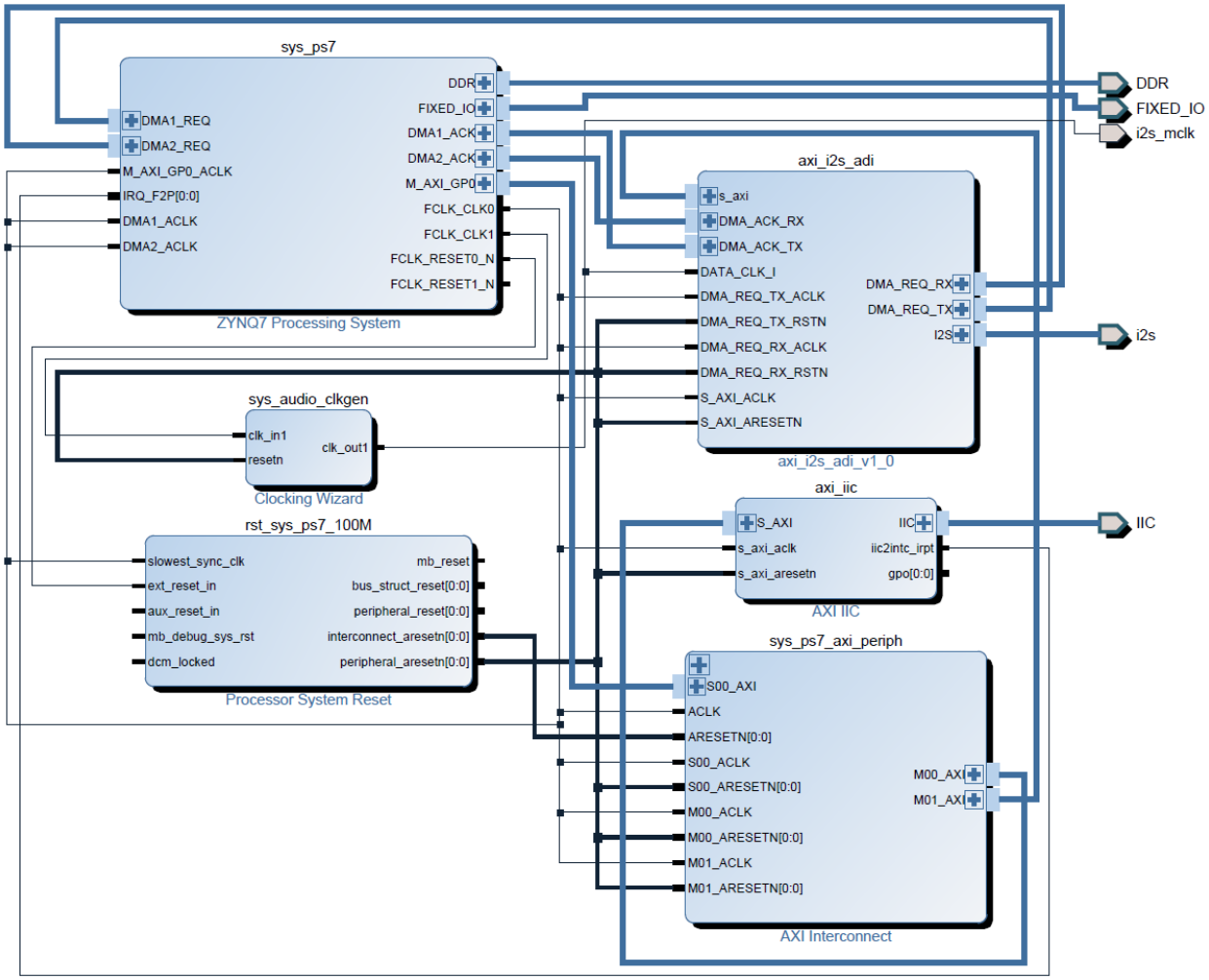
Procesor

Hlavní součástí celého blokové návrhu je procesor, který je označen jako IP jádro s názvem `sys_ps`, představuje PS a je nutný pro použití OS. Kromě důležitých komponent pro OS jako je například operační paměť, je PS nakonfigurována pro připojení čtečky SD karty, z které bude načítán OS [9]. Dále je povolen ethernet, který bude sloužit pro připojení k Internetu.

Sběrnice AXI

Pro propojení PS s PL slouží sběrnice AXI, která je označena názvem `sys_ps7_axi_periph`.

[1] https://github.com/analogdevicesinc/hdl/



Obrázek 3.1: Blokový návrh ve Vivado

Sběrnice I^{2}C

Realizace zápisu do konfiguračních registrů audio kodeku je provedena pomocí IP jádra `axi_iic`. Externí rozhraní `IIC` je namapováno na fyzické piny audio kodeku na desce pomocí souboru `zed_sys_constr_.xdc`.

Sběrnice I^{2}S

Sběrnici I^{2}S představuje IP jádro `axi_i2s_adi_` s přímým přístupem do paměti PS, které slouží pro přenos zvukových dat. Stejně jako předešlá sběrnice má vyvedeno externí rozhraní `i2s`, které je taktéž namapováno ve stejném souboru. Obě sběrnice jsou namapovány dle Tabulky 3.1.

Hodinový signál

Pro generování hodinového signálu pro audiokodek i sběrnici I^{2}S je použito IP jádro s názvem `sys_audio_clkgen`, které je nastaveno na frekvenci 12,288 MHz. Tato hodnota nastaví vzorkování do celočíselného módu.

Blokový návrh na obrázku 3.1 jsem pouze vyňal z referenčního návrhu zmíněného na začátku odstavce, jelikož pro ovládání audio kodeku nejsou potřeba všechny části a snímek návrhu by se nevešel na stránku. Výsledný soubor `bitstream`, který bude sloužit k naprogramování PL ve vestavěném systému v této práci, je vygenerován z referenčního návrhu.

Název signálu	Popis	Zynq pin
i2s_mclk	Master Clock Input	AB2
i2s_bclk	Digital Audio Bit Clock Input/Output	AA6
i2s_lrclk	Digital Audio Left-Right Clock Input/Output	Y6
i2s_sdata_out	Digital Audio Serial-Data DAC Input	Y8
i2s_sdata_in	Digital Audio Serial-Data ADC Output	AA7
iic_sda	I2C Serial Data interface	AB5
iic_scl	I2C Serial Data interface	AB4

Tabulka 3.1: Tabulka propojení pinů audiokodeku ADAU1761 se signály blokového návrhu.

3.2 Vestavěný Linux

Jako OS jsem zvolil vestavěný Linux (dále jen Linux) z důvodu absence ovladačů pro IP jádro sběrnice I^2S a audio kodek ADAU1761 u jiných OS. Aby bylo možné spustit Linux na platfomě Zynq, je vyžadováno vytvoření následujících souborů [3]:

1. **Zaváděcí obraz** (`BOOT.BIN`) – Jedná se o složený soubor, obsahující různé soubory. Soubor nesmí mít jiný název než je uvedeno a typicky obsahuje následující soubory:
   (a) **FSBL** – Soubor po nahrání inicializuje procesorové prostředky. Tento soubor je jako jediný povinný, avšak byla by provedena pouze inicializace.
   (b) **Bitstream** – Popisuje PL. Je nepovinný a při jeho absenci není použita PL.
   (c) **U-Boot** – Oficiálně *Das U-Boot* je zavaděč systémů používaný především ve vestavěných systémech spolu s OS Linux.
2. **Objektový kód Stromu zařízení** – Soubor obsahující detailní informace o všech zařízeních, které jsou v systému používány.
3. **Linuxové jádro** – Provede inicializaci hardwaru systému a připojí kořenový souborový systém.
4. **Kořenový souborový systém** – Obsahuje samotný OS.

Jako první se budu zabývat kořenovým souborovým systémem, jelikož má největší vliv na úpravy ostatních souborů. Následně popíši jednotnou část jádro a strom zařízení, které lze vytvořit v jednom kroku. Při vytváření souboru U-Boot je nutné upravit specifikaci načítání. V poslední části vytvořím zaváděcí obraz. Po posledním kroku bude možné spustit vestavěný Linux na vývojové desce ZedBoard.

Kořenový souborový systém

Stejně jako v běžném OS Linux se i zde kořenový souborový systém (dále jen FS) skládá z různých adresářů a souborů. Jediný rozdíl je v tom, že většinou obsahuje pouze takové adresáře a složky, které jsou pro daný účel potřebné. Není nutné vytvářet nový FS ručně, ale můžeme využít možnosti stáhnout čistý FS třetích stran a upravit jej dle vlastního uvážení. Využitím této možnosti také eliminujeme možnost naší chyby při jeho vytváření.

Na Internetu lze nalézt velké množství čistých FS. Například minimální FS pro architektury ARM nabízí firma Xilinx na svých stránkách[2]. Já jsem použil R ze stránek[3] společnosti Linaro, která se zabývá vestavěnými Linuxovými systémy na architektuře ARM ve spolupráci s Ubuntu. Díky tomu již bude vestavěný Linux obsahovat základní knihovny a bude mít přístup k softwarovým archivům distribuce Ubuntu, nadruhou stranu bude mít FS větší velikost než prvně zmíněná možnost. Pro uložení FS na SD kartu můžeme zvolit dva způsoby:

- Vložení do komprimovaného souboru `uramdisk.image.gz`. Jeho předností je menší velikost. Tato přednost se však může stát nevýhodou při větších velikostech FS, kdy je nutné při zavádění systému nejdříve soubor dekomprimovat.
- Nakopírování do vytvořeného oddílu formátu `ext4`. Tento způsob je výhodný pro větší FS, avšak klade větší paměťové nároky na SD kartu.

Já použiji druhý způsob, jelikož již od počátku bude mít FS větší velikost a v budoucnu by se mohl ještě zvětšit z důvodu nahrávání zvuku, tím by se mohla stát časová náročnost dekomprimace při zavádění systému prvního způsobu neúnosná.

U-Boot

Z repozitáře[4] firmy xilinx jsem stáhl nástroj pro vytváření souboru U-Boot, pro nastavení jsem použil konfigurační soubor `zynq_zed_config`, po jeho aplikování je nutné upravit sekci definice zavádění v souboru `zynq-common.h`, úprava je zobrazena na ukázce Blokového kódu 3.1. Tato úprava je nezbytná, jelikož ve výchozím nastavení U-Boot hledá při zavádění FS komprimovaný soubor `uramdisk.image.gz`.

```
"sdboot=if mmcinfo; then " \
    "run uenvboot; " \
    "echo Copying Linux from SD to RAM... && " \
    "load mmc 0 ${kernel_load_address} ${kernel_image} " \
    "load mmc 0 ${devicetree_load_address} ${devicetree_image} " \
    "bootm ${kernel_load_address} - ${devicetree_load_address};\0 " \
    "fi\0" \
```

Blokový kód 3.1: Změna nastavení sekce sdboot pro načítání Linuxu z oddílu ext4.

[2] http://www.wiki.xilinx.com/Build+and+Modify+a+FS

[3] https://releases.linaro.org/archive/15.05/ubuntu/vivid-images/developer/linaro-vivid-developer-20150522-704.tar.gz

[4] https://github.com/Xilinx/u-boot-xlnx

Jádro a strom zařízení

Vytvoření těchto souborů můžeme provést pomocí programu PetaLinux nebo Yocto, avšak pokud nechceme konfigurovat od začátku, musíme do nich naimportovat zdrojové soubory Linuxu z repozitáře[5] firmy AD, který obsahuje potřebné ovladače a konfigurační soubory pro náš projekt. Obsahuje také samotné nástroje pro jejich tvorbu, jelikož nejsou následující operace náročné, použiji místo nutnosti stažení a instalace prvně zmíněných nástrojů tyto skripty. Pro vytvoření souboru jádra `uImage` jsem použil konfigurační soubor `zynq_xcomm_adv7511_defconfig`, při jeho vytvoření se zároveň vytvoří objektový kód stromu zařízení `devicetree.dtb`, který bude použit pro popis hardwaru při zavádění Linuxu.

Zaváděcí obraz

V této části jsem použil nástroj XSDK stejné verze jako Vivado. XSDK umožňuje vytvořit soubor FSBL z exportovaného projektu ve Vivado a následně i zaváděcí obraz `BOOT.BIN`. Po vytvoření zaváděcího souboru jsem narazil na problém, kdy jej ZedBoard nemohl nalézt a nijak nereagoval. Po překontrolování správného přiřazení ovladačů a pořádí umístění souborů, problém stále přetrvával. Nakonec se mi podařilo soubor načíst na ZedBoard, avšak při vytvoření v nižší verzi XSDK, konkrétně 13.4.

3.3 Implementační prostředí

V tomto odstavci představím technologie, které je možné pro požadovanou aplikaci použít. Od zvolení vhodné programovacího jazyka se již další části budou odvíjet od jeho knihoven a možností jeho napojení na danou technologii.

3.3.1 Programovací jazyk

Než se pustím do návrhu aplikace, je nutné zvolit, který programovací jazyk zvolit pro vytvoření aplikace. Ve vestavěných systémech probíhá vývoj většiny aplikací v jazyce C. Důvodem je zejména velký důraz na výkon a omezené prostředky vestavěných systémů [7]. V případě platformy Zynq však máme dostatečné prostředky pro využití některého z vysokoúrovňových jazyků, které nám mohou umožnit rychlejší vývoj aplikace.

Z vysokoúrovňových jazyků se nabízí jazyk C++, který vychází z jazyka C, avšak vývoj není tak rychlý jako například v některém z interpretovaných jazyků Python nebo Ruby. Já jsem se rozhodl použít Python, jelikož jsem se s ním již během studia setkal a oproti jazyku Ruby za ním stojí širší základna uživatelů, díky čemuž má Python více knihoven a rozšíření.

Python je dostupný ve dvou hlavních verzích, a to 2.7 a 3.6. V projektu bude použita verze 2.7, neboť v době psaní této práce byla označena za finální, tedy i stabilní. Disponuje také obsáhlejší dokumentací a větší kompatibilitou se staršími knihovnami[6]. Do vestavěného systému není nutné instalovat tuto verzi Python interpretu, jelikož je součástí většiny distribucí systému Linux.

[5] https://github.com/analogdevicesinc/linux
[6] https://wiki.python.org/moin/Python2orPython3

3.3.2 Grafické uživatelské rozhraní

Jednou z možností pro vývoj grafického uživatelského rozhraní (dále jen GUI) je použít některou z grafických knihoven jazyka Python. Mezi nejpoužívanější se řadí knihovny PyGTK a PyQT. Tyto knihovny vytvářejí nádstavbu nad grafickým subsystémem OS. Pro komunikaci mezi klientem a serverem můžeme například použít modul QtNetwork pro knihovnu PyQt, modul používá pro komunikaci rodinu protokolů TCP/IP. Avšak nevýhodou těchto knihoven je nutnost uživatelskou aplikaci nahrát do každého zařízení, které má být určeno pro ovládání. Další nepříjemností je absence těchto knihoven mezi standartními knihovnami jazyka Python, čož by nutilo klienta k doinstalování těchto knihoven.

Abych nenutil klienta, nahrávat či doinstalovávat žádné součásti aplikace a odstínil jej co nejvíce od použité technologie, rozhodl jsem se použít možnost vytvoření GUI pomocí webového grafické rozhraní. V tomto případě musí mít klient pouze nainstalovaný internetový prohlížeč, jenž je v dnešní době součástí základních programu naprosté většiny zařízení.

Rozdělení aplikace tedy bude na klientskou část, která poběží ve webovém prohlížeči a serverovou část aplikace běžící na ZedBoard. Komunikace mezi klientem a serverem bude probíhat asynchronně pomocí zasílání zpráv, kdy klient při interakci s GUI zašle zprávu serveru a server mu následně odpoví dle požadavku klienta.

Klientská část aplikace bude využívat běžnou kombinaci technologií pro vývoj webového rozhraní, která se skládá z HTML, CSS a JavaScriptu. Stručná charakteristika těchto technologií je:

- **HTML** *Hypertext MarkUp Language* – Je značkovací jazyk sloužící pro tvorbu webových stránek. Obsah stránek se skládá z elementů, které jsou definované pomocí tzv. tagů v souboru s příponou `*.html`. V projektu bude použita jeho poslední verze označována jako HTML 5.
- **CSS** *Cascading Style Sheets* – Neboli kaskádové styly, je označení pro jazyk definující vzhled a umístění elementů definovaných v HTML.
- **JavaScript** – Objektově orientovaný skriptovací jazyk, v klientské části aplikace jej budu používat pro ovládání interaktivních prvků GUI. K tomuto účelu použiji jeho knihovnu jQuery.

3.3.3 Webový server

Pro komunikaci mezi klientem a serverem, musíme použít na serverové části webový server, což je program používající protokol HTTP pro výměnu hypertextových dokumentů ve formátu HTML. Dnešní webové servery poskytují širokou škálu služeb. Tyto servery však kladou vyšší nároky na hardware, což není z hlediska vestavěných systémů žádoucí. Proto je vhodné použít tzv. odlehčené webové servery, které obsahují naprosté minimum služeb, případně dovolují přidat či odebrat požadovanou službu. Mezi odhlečené webove servery patří:

- **Nginx** používá škálovatelnou, událostmi řízenou (asynchronní) architekturu. Díky čemuž je velice škálovatelný a díky lepší práci s pamětí se hodí pro zařízení s omezenými prostředky. Pokud jsou zapotřebí služby, které minimální instalace neobsahuje, lze provést jejich instalaci pomocí volitelných modulů a doplňků.
- **Lighttpd** nabízí bezpečnost, rychlost a flexibilitu. Je navržen a optimalizován pro vysoce výkonná prostředí. V porovnání s jinými webovými servery nabízí efektivní

správu zatížení CPU a paměti. Nabízí celou řadu pokročilých funkcí jako například SCGI (*Simple Common Gateway Interface*), FastCGI a možnost autorizace.

- **Monkey HTTP Deamon** zaměřený na trh s vestavěnými systémy. Poskytuje funkce jako jsou virtuální hostitele, podpora pluginů, rozhraní C API a HTTP/1.1. Stejně jako Nginx se jedná o asynchronní (událostně řízený) webový server. Nenabízí však tak rozsáhlé množství volitelných doplňků jako předešlé příklady. Velkou předností tohoto webového serveru je však jeho velikost, Monkey má přibližně 100 KB při instalaci a kolem 250 KB za běhu.
- **CherryPy** nejedná se pouze o webový server, ale je to objektově orientovaný webový framework jazyka Python, který obsahuje svůj vlastní vestavěný server. V případě potřeby použití jiného serveru, lze aplikaci napsanou v tomto frameworku nasadit na jiné servery podporující rozhraní WSGI (*Web Server Gateway Interface*) [4]. Kromě knihovny Python nepotřebuje CherryPy žádné další knihovny pro svůj běh a jeho velikost je pouze 650 KB.

Z výše uvedeného seznamu jsem v projektu použil framework CherryPy (verze 3), jelikož obsahuje vestavěný server, který je ihned po instalaci frameworku připravený k použití, nabízí CherryPy oproti ostatním možnostem pustit se rovnou do vývoje webové aplikace a díky snadnému zpřístupnění metod jazyka Python umožňuje možnost rychlejšího vývoje.

3.4 Ovládání zvukové karty

Pro ovládání zvukové karty na Linuxu slouží ALSA (*Advanced Linux Sound Architecture*). Jedná se knihovnu a linuxovou jadernou komponentu, která poskytuje API pro ovladače zvukových karet. ALSA nahradila zastaralé zvukové rozhraní *(*Open Sound System) a stala se součástí vetšiny linuxových jader od verze 2.5. Jelikož je napsána v jazyce C, musíme pro její použití v Pythonu vytvořit adaptér nebo použít již existující knihovnu, která nabízí pro tento účel API. Pro Python existují následující knihovny pro tento účel:

- **pyalsa** umožňuje pouze možnost přehrávání zvukových dat v neblokovém režimu.
- **alsaaudio** nabízí separátní třídy pro možnost nastavení směšovače i přehrávání zvukových dat.

Knihovna pyalsa, stejně jako mnoho dalších Python knihoven pro práci s ALSA již není dále vyvíjena. Proto jsem se rozhodl využít pro imlementaci knihovnu alsaaudio, která by měla podporovat většinu funkcí, které budu v aplikaci využívat.

3.5 Uživatelské rozhraní

V této části specifikuji požadavky, které může uživatel na aplikaci klást a na jejich základě navrhnout výsledný modul, který tyto požadavky uspokojí. Hlavním účelem uživatelského rozhraní bude umožnit uživateli nastavovat vlastnosti audio kodeku a vyzkoušet zda mu zvolené nastavení vyhovuje. Jednotlivá nastavení by měla aplikace seskupit, tak aby uživatel nemusel jednotlivá nastavení hledat mezi ostatními.

Pro navigaci jsem se rozhodl použít jednoduché menu, hlavní stránka bude obsahovat směšovač audio kodeku a prvek pro jednoduchou demonstraci daného směšovače. Výsledný návrh uživatelského rozhraní je zobrazen na Obrázku 3.2 a jednotlivé bloky slouží pro:

Obrázek 3.2: Návrh rozložení modulů.

- `Menu` bude obsahovat položky, které budou umožňovat uživateli navigaci.
- `Směšovač` bude informovat uživatele o jednotlivých nastaveních audio kodeku pro jednotlivá ovládání a umožňovat jejich změnu.
- `Ovládání` bude obsahovat prvek pro demonstraci prováděných změn ve směšovači.

Kapitola 4

Imlementace aplikace

Tato kapitola popisuje jednotlivé části implementace aplikace dle návrhu řešení, který byl popsán v předešlé kapitole.

4.1 Serverová část

Tato část řeší obsluhu zvukové karty jako je její nastavení, přehrávání a záznam zvuku. Kromě toho přijímá požadavky z klientské části a poskytuje ji požadované data nebo provede požadovanou operaci. Z důvodu velkého počtu činností jsem tuto část rozdělil do dvou podčástí.

4.1.1 Ovládání zvukové karty

Ovládání zajišťuje třída `PlayerController`, která používá třídu `Player` pro přehrávání tónů a obsahuje metody pro nastavení zvukové karty. Třída bude sloužit jako rozhraní pro práci se zvukovou kartou.

Přehrávání

Přehrávání tónu zajišťuje třída `Player`, která používá knihovnu alsaaudio a standartní knihovnu wave. Aby bylo možné nahrávku spustit, je nutné nejdříve zjistit informace o přehrávaném souboru, pro tento účel jsem vytvořil přepravku `SampleInfo`, přepravka používá knihovnu wave pro zjištění:

- Počtu kanálů
- Snímkovací frekvenci
- Velikost periody
- Šířky vzorku
- Formátu pulzně kódové modulace

Dle zjištěných informací je následně inicializován objekt metodou `alsaaudio.PCM()`, který umožňuje přehrát nahrávku.

Nastavení

Pro nastavení zvukové karty jsem původně zamýšlel použít knihovnu alsaaudio, která obsahuje třídu `Mixer` pro nastavení jednotlivých možností zvukové karty. Pomocí této třídy však nebylo možné provádět změny nastavení z důvodu její nekompability, kdy nebylo možné vytvořit objekt třídy `Mixer` pro daný název zvukové karty. Proto jsem k tomuto účelu použil nástroj `amixer`, který slouží pro nastavování zvukových karet podporujících ALSA. V třídě `PlayController` jsem implementoval metody umožňující měnit nastavení, mezi tyto metody patří například:

- `setHphVolume` – Nastaví úroveň hlasitosti pro port `Headphone out` na požadovanou hodnotu.
- `setHphState` – Zapne či vypně port `Headphone out`.
- `play` – Přehraje požadovaný tón.
- `record` – Povolí použití mikrofonu.
- `setCaptureVolume` – Nastaví zesílení záznamu mikrofonu na portu `MIC`.
- `setLeftChannelVolume` – Nastaví úroveň hlasitosti levého kanálu.
- `setLeftChannelState` – Zapne či vypne levý kanál.

4.1.2 Webový server

Z nabídky možností webových serverů v Odstavci 3.3 jsem vybral možnost použít framework CherryPy, který již má v sobě zabudovaný server. Důvodem této volby je jeho důraz na jednoduchost použití a konfiguraci. Cherrypy také poskytuje přehlednou a kvalitní dokumentaci[1]. Použiji jeho verzi 3, která je označena jako stabilní.

Ačkoliv jde jeho webový server použít ihned po instalaci, musel jsem udělat dvě drobné změny v jeho konfiguraci:

- Změnit rozhraní, na kterém bude nasazen (ve výchozím nastavení je nastaven na *localhost*)
- Nastavit statickou adresu adresáře pro statické soubory jako jsou HTML, CSS a JavaScript. Tato úprava je nutná kvůli trasování metod v Cherrypy, které mění aktuální kontext umístění při navigaci na webových stránkách. Bez této konfigurace není možné v HTML souborech načítat statické soubory.

Navigace

Pro navigaci na webové stránce jsem vytvořil dvě třídy, kdy každá reprezentuje jednotlivé podstránky nabízené v menu. Aby byla metoda v jazyce Python zpřístupněna webové stránce, je nutné před její definici vložit anotaci `@cherrypy.exposed`. Obě třídy implementují metodu `index()`, která slouží pro zpřístupnění obsahu popsaného v jazyce HTML pro danou podstránku.

[1] http://docs.cherrypy.org/en/latest/

Zpracování požadavků

Zpracování požadavků uživatele je implementováno pomocí zpřístupněných metod, kdy každý název metody odpovídá URL (*Uniform Resource Locator*) adrese, z které bude daný požadavek zaslán. Je očekáván požadavek typu `POST` spolu s jeho parametry, které mají být aplikovány. Po jejich zpracování jsou na základě jejich parametrů zavolány metody implementované v odstavci 4.1.

V ukázce Blokového kódu 4.1 je vidět jednoduchá implementace zpřístupněných metod v CherryPy, které jsou použity v aplikaci.

```
@cherrypy.expose
def index(self):
 return open(os.path.join(VIEW_DIR, u'public/player.html'))

@cherrypy.expose
def playTone(self, **data):
 player.play(data['classNames'])
```

Blokový kód 4.1: Ukázka zpřístupnění stránky přehrávání a zachycení požadavku na tón.

4.2 Klientská část

Uživateli je klientská část aplikace zpřístupněna pomocí webového rozhraní, kdy vizuální část uživatelského rozhraní je implementovaná v jazyce HTML, CSS a dynamické prvky pomocí JavaScriptové knihovny jQuery. Webová stránka pro nastavení zvukové karty a přehrávání tónů je zobrazena na Obrázku 4.1. Skládá se ze tří hlavních částí, první je menu (1), následuje směšovač(2) a klavír (6) pro přehrávání. Poslední dvě zmíněné části rozeberu níže.

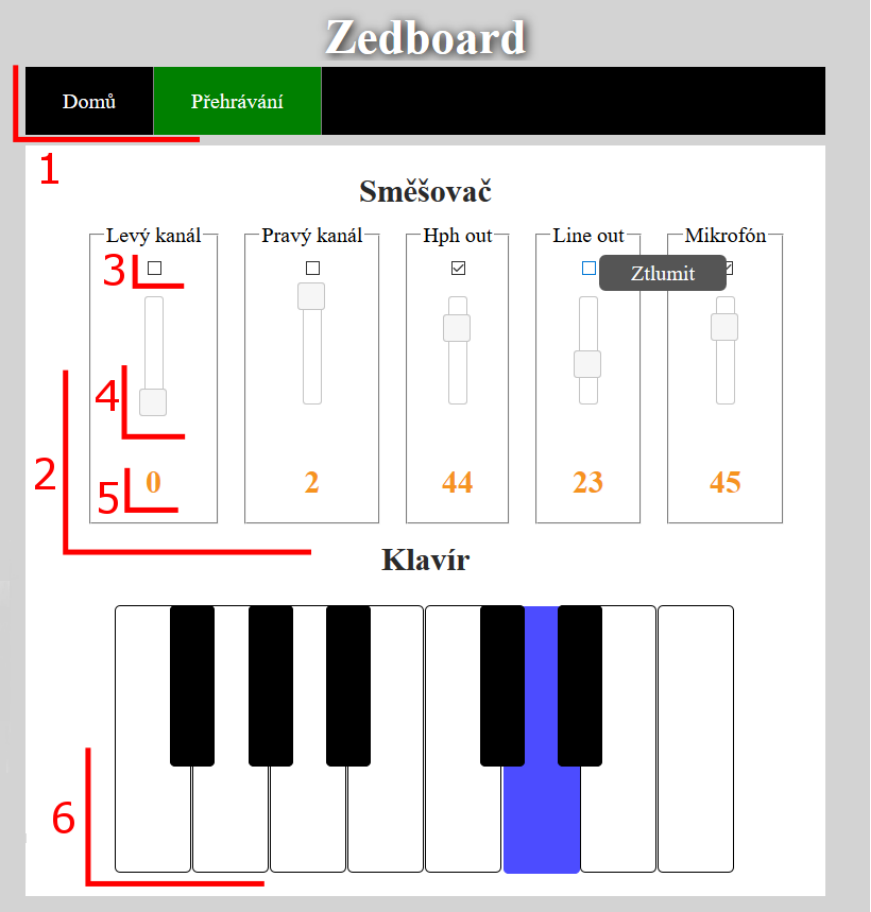
4.2.1 Směšovač

Slouží uživateli pro nastavení zvukové karty a je implementovaný jako seznam prvků. Jednotlivé skupiny seskupují spolu související nastavení a skládají se ze zatrhávácího tlačítka (3), které danou skupinu povolí či zakáže. Pro určení hodnoty daného nastavení slouží posuvník (4), pod kterým se nachází jeho aktuální hodnota (5).

Chování prvků je implementováno pomocí knihovny jQuery. Například posuvník umožňuje měnit hodnotu v dovoleném rozmezí pro dané nastavení. Hodnota ve vizuální části se mění v reálném čase, avšak požadavek na změnu se posílá jakmile uživatel ukončí manipulaci s pusuvníkem. Díky tomu nejsou serveru zasílány velké množství požadavků. V Blokovém kódu 4.2 lze vidět danou implementaci metod. Kdy událost `slide` mění pouze hodnotu proměnné, kdežto událost `stop` volá metodu `POST`, která pošle serveru požadavek.

```
slide: function( event, ui ) {
 $("#lChannelValue").val(ui.value);
},
stop: function( event, ui ) {
 $.post('/player/setLeftChannelVolume', {value : ui.value})
   .done(function (reply) {;} );
}
```

Blokový kód 4.2: Ukázka metod posuvníků.



Obrázek 4.1: Vzhled uživatelského rohraní aplikace pro změnu nastavení zvukové karty a demonstaci této změny.

Pro vysvětlení daného prvku je po najetí kurzoru vypsán jeho účel, mixér umožňuje uživateli měnit následující nastavení.

- Hlasitost portu Line out a jeho ztlumení.
- Hlasitost portu Headphone a jeho ztlumení.
- Úroveň levého kanálu a jeho povolení/zakázání.
- Úroveň pravého kanálu a jeho povolení/zakázání.
- Povolení zaznamenávání zvuku a jeho streaming na výstupní porty.

4.2.2 Klavír

Klavír (6) je vytvořen pomocí dvou seznamů, jejichž vzhled je upraven pomocí CSS. Stisk klávesy je simulován mírnou změnou pozice položky a změnou její barvy. Po stisku dochází k zaslání požadavku na přehrání odpovídajícího tónu[2]. Účelem této sekce je demonstrovat změny nastavení, které uživatel provedl ve směšovači.

[2]Nahrávky tónů jsem použil volně dostupné nahrávky na adrese: https://www.freesound.org/people/pinkyfinger/packs/4409/

Kapitola 5

Závěr

Cílem této práce bylo seznámit se s platformou Xilinx Zynq a vytvořit vestavěný systém na její platformě, který bude obsahovat zvukové zařízení a umožnit běžnému uživateli jeho správu. Jako dostupným referenčním zařízením s touto platformou byla použita vývojová deska ZedBoard, která je osazena audio kodekem ADAU1761.

V práci byla nejdříve popsána samotná platforma Zynq, kde byly představeny jednotlivé části platformy. Následně jsem popsal stavající řešení, definoval jsem nevýhody stávajícího řešení a specifikoval požadavky na jeho vylepšení.

V Kapitole 3 jsem nejdříve popsal hardwarový návrh nutný pro správné propojení audio kodeku s PS. Když již byla hotova hardwarová část, popsal jsem nutné kroky a možnosti při nasazení vestavěného Linuxu, tak aby bylo možné, používat zvolený hardware. Po těchto částech již bylo možné spustit ZedBoard a vyzkoušet správné spuštění Linuxu spolu s otestováním funkčnosti audio kodeku.

Následně jsem se zabýval analýzou a návrhem aplikace, která by umožňovala vzdálenou správu daného vestavěného systému. Zde jsem popsal různé možnosti řešení a představil technologie, které dané řešení mohou usnadnit. Zhodnotil jsem výhody a možná úskalí navrhnutých řešení.

Zvolená řešení jsem nakonec implementoval v Kapitole 4, kde jsem popsal jednotlivé části implementace a zmínil také problémy, které se objevily při implementaci navrhnutých řešení. V této kapitole je také zobrazena výsledná podoba aplikace s popisem.

Lze říci, že se mi všechny zadané požadavky na vestavěný systém i aplikaci podařilo splnit. Experimentálním testováním jsem ověřil funkčnost jednotlivých prvků aplikace, při kterém se aplikace chovala podle předpokládaného scénaře. Hlavním přínosem této práce spatřuji v umožnění ovládání zvoleného zařízení běžnému uživateli, a to i bez znalosti linuxového prostředí či práce s příkazovým řádkem. Dalším přínosem je možnost použít jednotlivé části v projektech, které dané zařízení používají.

I přesto, že aplikace splňuje všechny specifikované požadavky, můžeme v ní najít vylepšení nebo další funkcionalitu, která by nabídla uživateli více možností práce se zvukovým zařízením. V budoucnu by mohlo být implementováno například vkládání a následné přehrávání libovolných zvukových dat nebo přehrávání nahraných dat přímo v prohlížeči. V neposlední řadě jsem při návrhu aplikace nepředpokládal ovládání zařízení více uživateli najednou, při tomto požadavku na systém by bylo nutné, periodicky ověřovat nové informace o změnách provedených jiným uživatelem, tak aby měli všichni přehled o aktuálním stavu zařízení.

Literatura

[1] Berger, A. S. *Embedded systems design.* San Francisco: CMP Books, 2002. ISBN 1–57820–073–3.

[2] Crockett, L. H., Elliot, R. A., Enderwitz, A. M. et al. *The Zynq book: tutorials for Zybo and ZedBoard.* Glasgow: Strathclyde Academic Media, 2015. ISBN 978–0–9929787–3–0.

[3] Crockett, L. H., Elliot, R. A., Enderwitz, A. M. et al. *The Zynq book: embedded processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 all programmable SoC.* Glasgow: Strathclyde Academic Media, 2014. ISBN 978–0–9929787–0–9.

[4] Hellegouarch, S. *CherryPy essentials: rapid Python web application development.* [b.m.]: Packt Publishing, 2007. ISBN 1904811841.

[5] Pellerin, D. a Thibault, S. *Practical FPGA Programming in C.* Upper Saddle River: Prentice Hall, 2005. ISBN 0–13–154318–0.

[6] Salvador, O. a Angolini, D. *Embedded Linux Development with Yocto Project.* [b.m.]: Packt Publishing, 2014. ISBN 1783282339.

[7] Soulier, P. a Depeng, L. Blueprint for an embedded systems programming language. In *IET Conference Proceedings* [online]. Stevenage: The Institution of Engineering, 2014. ISBN 978–1–84919–970–4.

[8] Xilinx. *Artix-7 FPGA Product Brief* [online]. 2016, Poslední změna 27. září 2016 [cit. 20. dubna 2017]. Dostupné na: `https://www.xilinx.com/support/documentation/product-briefs/artix7-product-brief.pd`

[9] Xilinx. *Zynq-7000 All Programmable SoC - Technical Reference Manual, UG585, v1.11* [online]. 2016, Poslední změna 27. září 2016 [cit. 25. dubna 2017]. Dostupné na: `https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf`.

[10] Yaghmour, K. *Building embedded Linux systems.* 2. vyd. Sebastopol: O'Reilly, 2008. ISBN 978–0–596–52968–0.

Přílohy

Příloha A

Obsah přiloženého DVD

- `bootFiles` – Adresář se soubory pro zavedení vestavěného systému na ZedBoard.
- `manual.pdf` – Manuál pro spuštění vestavěného systému (včetně webové aplikace).
- `vivado` – Adresář s Vivado projektem.
- `webapp` – Adresář s webovou aplikací.
- `rootfs` – Adresář s kořenovým souborovým systémem řešení.
- `tex` – Adresář s texty práce v LaTeXu.

Příloha B

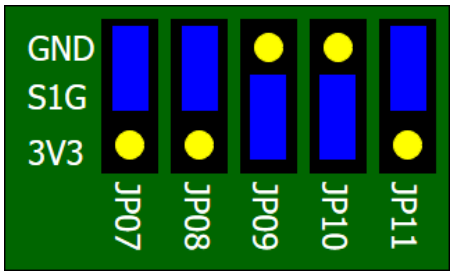
Manuál k spuštění projektu

Tento manuál popisuje spuštění vestavěného systému, webového serveru a následné připojení k webové aplikaci na vývojové desce ZedBoard.

Jako první je nutné připravit SD kartu, musí obsahovat následující dva oddíly s následujícími souborovými systémy:

- **FAT32** (min 50 MB) – Zde se umístí kořenový soubory z adresáře `bootFiles`, který se nechází na přiloženém na DVD.
- **ext4** (min 1 GB)– Zde se umístí kořenový souborový systém z adresáře `rootfs`, který se nechází na přiloženém na DVD.

Následně je nutné nastavit jumpery na pinech stejně jako je zobrazeno na Obrázku B.1. Tím se nastaví ZedBoard pro zavádění systému z SD karty.



Obrázek B.1: Nastavení pinů pro zavádění systému z SD karty.

Po vložení SD karty a následném zapnutí ZedBoard se začné načítat operační systém. Pro přípojení k systému lze použít UART nebo SSH. Pro přihlášení použijte přihlašovací jméno **root** s heslem **root**.

Webová aplikace se nachází v adresáři `/root/webapp/`. Spuštění aplikace se provede příkazem `python myapp.py`. Následně je možné pomocí webového prohlížeče ovládat zvukovou kartu. Webová stránka je dostupná na adrese přidělené pro rozhraní `ethernet` a portu 8080.