VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA PROGRESIVNÍCH HW ŘEŠENÍ PRO ZPRACOVÁNÍ REAL-TIME MEDÍÍ

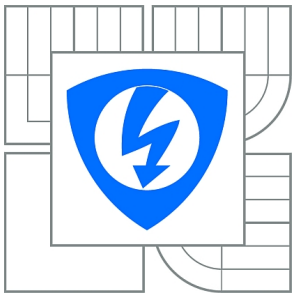
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JAN REŽNÝ
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA PROGRESIVNÍCH HW ŘEŠENÍ PRO ZPRACOVÁNÍ REAL-TIME MEDIÍ

ANALYSIS OF PROGRESSIVE HARDWARE FOR REAL-TIME MEDIA PROCESSING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

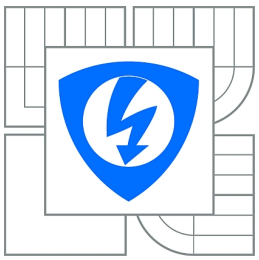
AUTOR PRÁCE — Bc. JAN REŽNÝ

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR SYSEL, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Jan Režný	***ID:***	119589
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Analýza progresivních HW řešení pro zpracování real-time medií

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s nejnovějšími trendy v oblasti mikroprocesorů vhodných pro zpracování real-time médií. Nastudujte parametry progresivních řešení vhodných pro zpracování zvuku, videa, převodu mezi kodeky, šifrování. Zhodnoťte jednotlivé typy z hlediska: ceny, spotřeby energie, podpory pro integraci do škálovatelného řešení, podpory pro paralelní výpočty. Zaměřte se na způsoby měření výkonnosti při zpracování zvukových signálů a proveďte porovnání platforem ARM (PandaBoard, Raspberry, BeagleBone) a řešení Parallella Computer. Pro měření využijte kódování a dekódování více paralelních datových toků v čase, např. dostupnou implementaci audiokodeku OPUS v jazyce C.

DOPORUČENÁ LITERATURA:

[1] Adapteva. Parallela Reference Manual. 2014. Dostupné na URL http://www.parallella.org/docs/parallella_manual.pdf [cite 16.10.2014]
[2] ARM Holdings. The ARM Architecture Reference Manual.
[3] IETF. Definition of the Opus Audio Codec. RFC6716. 2012. Dostupné na URL http://tools.ietf.org/pdf/rfc6716.pdf [cite 16.10.2014]

Termín zadání:	9.2.2015	***Termín odevzdání:***	26.5.2015

Vedoucí práce: Ing. Petr Sysel, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práce se zaměřuje na výběr vhodného HW řešení pro paralelní zpracování několika zdrojů zvukového signálu. Porovnává několik různých platforem založených na architekturách ARM, x86 a Epiphany, porovnává jejich výkon při sériovém a paralelním zpracování dat, jejich spotřebu energie a cenu.

KLÍČOVÁ SLOVA

opus, audio, kodek, ARM, x86, Epiphany, Parallella, paralelní zpracování dat

ABSTRACT

Diploma thesis focuses on the selection of suitable HW solution for parallell processing of multiple audio sources. Compares several different platforms based on architectures ARM, x86 and Epiphany, compares their performance in serial and parallel data processing, their energy consumption and price.

KEYWORDS

opus, audio, codec, ARM, x86, Epiphany, Parallella, parallel data processing

REŽNÝ, Jan *Analýza progresivních HW řešení pro zpracování real-time medií*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 46 s. Vedoucí práce byl Ing. Petr Sysel, Ph.D

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Analýza progresivních HW řešení pro zpracování real-time medií“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petru Syslovi Ph.D za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Tato práce se bude zabývat analýzou vhodných HW řešení pro paralelní zpracování několika zdrojů zvukového signálu. Paralelní zpracování dat je výhodné v případě vhodného hardwaru který dokáže efektivně využívat více procesů najednou.

V této práci budou popsány vybrané platformy založené na architekturách ARM, x86 a řešení Parallella. Parallella je jednodeskový minipočítač kombinující procesor architektury ARM a koprocesor architektury Epiphany. Tento koprocesor je sestaven z 16 jader optimalizovaných pro výpočty s plovoucí desetinnou čárkou, je tedy pro paralelní výpočty ideální.

Dále bude popsán vývoj analyzačního software, který bude měřit dobu zpracování zvukového signálu při kódování do formátu Opus a dobu zpracování při jeho dekódování.

Naměřená data budou zpracována a bude zvoleno nejvhodnější zařízení pro paralelní zpracování několika zdrojů signálu s nejlepším poměrem výkon/spotřeba.

1 SROVNÁNÍ DOSTUPNÉHO HW

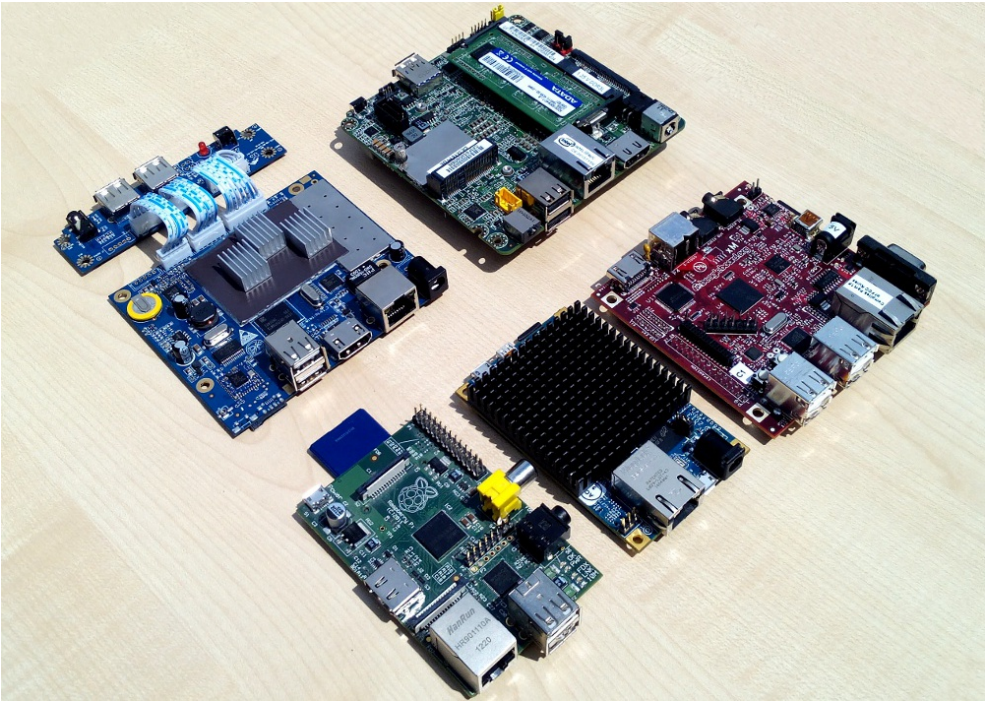
Pro analýzu bylo vybráno několik platforem založených na architekturách ARM včetně desky Parallella kombinující hlavní CPU architektury ARM a koprocesor architektury Epiphany. Vzhledem k rychlému vývoji vedeném firmou Intel v oblasti nízkoodběrových procesorů založených na platformě x86 byly dodatečně vybrány ještě dvě řešení založené na procesorech Intel Atom, průměrně vybavený notebook s mobilním CPU Intel Core i5 a stolní počítač s CPU Intel Core i7.

V následující tabulce je stručný přehled zvolených platforem pro analýzu s jejich parametry. Spotřeba energie je zde uvedena tak jak byla naměřena pomocí laboratorního zdroje při provádění analýzy, pouze u notebooku a stolního počítače byla spotřeba odhadnuta z výrobcem udávaného TDP[1] použitého procesoru. Tento odhad byl zvolen z důvodu přítomnosti jiných elektronických obvodů a periferií ovlivňujících spotřebu, ale neovlivňujících výkon celku při provádění analýzy (LCD displej, pevný disk, grafická karta). Jednodeskové platformy, u nichž byla spotřeba energie měřená, mají podobné obvody často také přítomny, avšak na celkové spotřebě se významně neprojevují.

Cena uvedená v tabulce níže je doporučená koncová cena udávaná výrobcem v době uvedení dané platformy na trh, v případě notebooku a stolního PC se jedná o cenu CPU.

platforma	**CPU**	**jádra**	**frekvence**	**spotřeba**	**cena**
Raspberry PI mod.B	ARM1176JZF-S	1	700 MHz	2W	35 $
Beagle Board xM	TI DM3730	1	1 GHz	3W	150 $
Parallella - ARM	ARM Cortex-A9	2	1 GHz	5W	99 $
Parallella - Epiphany	Epiphany-III	16	600 MHz	5W	99 $
Intel NUC	Atom E3815	1	1.46 GHz	5W	200 $
Pipo X7	Atom Z3736F	4	1.33 GHz	6W	100 $
Notebook	Core i5 4200U	2 (4)	1.6 GHz	~20W	280 $
Stolní PC	Core i7 3770K	4 (8)	3.5 GHz	~80W	320 $

[1]Thermal Design Power, nejvyšší tepelný výkon součástky, u nemechanických součástek je tato hodnota téměř rovná spotřebě energie



Obr. 1.1: Analyzované jednodeskové počítače

1.1 Architektura ARM

Procesory založené na architektuře ARM nabízejí nízkou spotřebu elektrické energie při vysokém výpočetním výkonu. Jednoduchý návrh a omezená instrukční sada se podepisují vysokou měrou na vysoké účinnosti při nízké výrobní ceně těchto procesorů. V dnešní době jsou tyto procesory velmi oblíbené u přenosných zařízení (smartphony, tablety, chytrá „nositelná elektronika“) právě z důvodů nízké spotřeby energie při provozu z baterií, a u levných minipočítačů díky jejich příznivé ceně.

Jak bylo výše uvedeno, procesory založené na architektuře ARM jsou procesory s omezenou instrukční sadou - RISC[2]. Oproti procesorům CISC[3] je zde například nutné některé operace rozepisovat do více instrukcí, což zvyšuje složitost jejich programování v případě používání strojového kódu, ale v současné době díky využívání vyšších programovacích jazyků není tato vlastnost výrazným problémem.

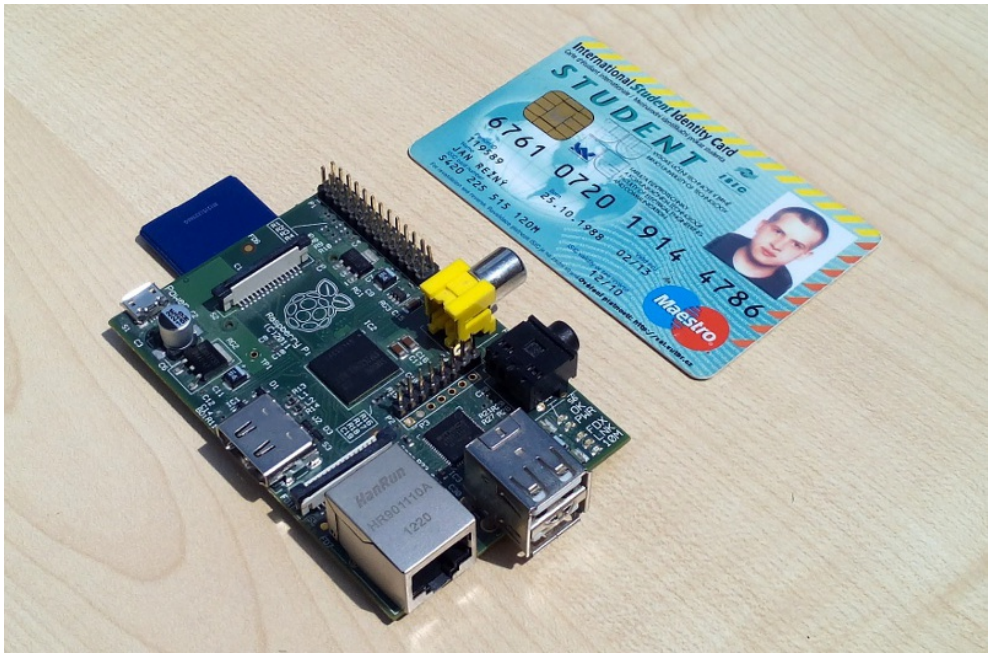
[2]Reduced Instruction Set Computing
[3]Complex Instruction Set Computing, procesory s komplexní instrukční sadou

1.1.1 Raspberry Pi

Raspberry Pi je série jednodeskových minipočítačů s deskou o velikosti platební karty. Tuto platformu vyvíjí britská nadace Raspberry Pi foundation původně za účelem podpory rozšíření výuky informatiky ve školách.

Pro analýzu byla využita verze Raspberry Pi „model B“, lišící se od základní verze větší operační pamětí RAM (512MB). Základem této desky je čip Broadcom BCM2835 využitý i v základní verzi, integrující jednojádrový CPU ARM1176JZF-S taktovaný na frekvenci 700MHz a grafický čip Broadcom VideoCore. Jako úložiště dat je zde využita SD karta ze které počítač také zavádí operační systém. V tomto řešení byl zvolen OS Raspbian, operační systém na Linuxovém jádře vycházející z OS Debian 7.

Tento minipočítač je celosvětově známý a velmi rozšířený, za velmi příznivou cenu nabízí dostatečný výkon pro nejrůznější aplikace. Kromě standardních konektorů HDMI, RJ-45, USB, 3,5mm jack disponuje také několika rozšiřujícími konektory určenými pro připojení kamery nebo jiných periferií (GPIO[4]) a konektorem Cinch pro analogový obraz. Napájení je řešeno pomocí konektoru MicroUSB.



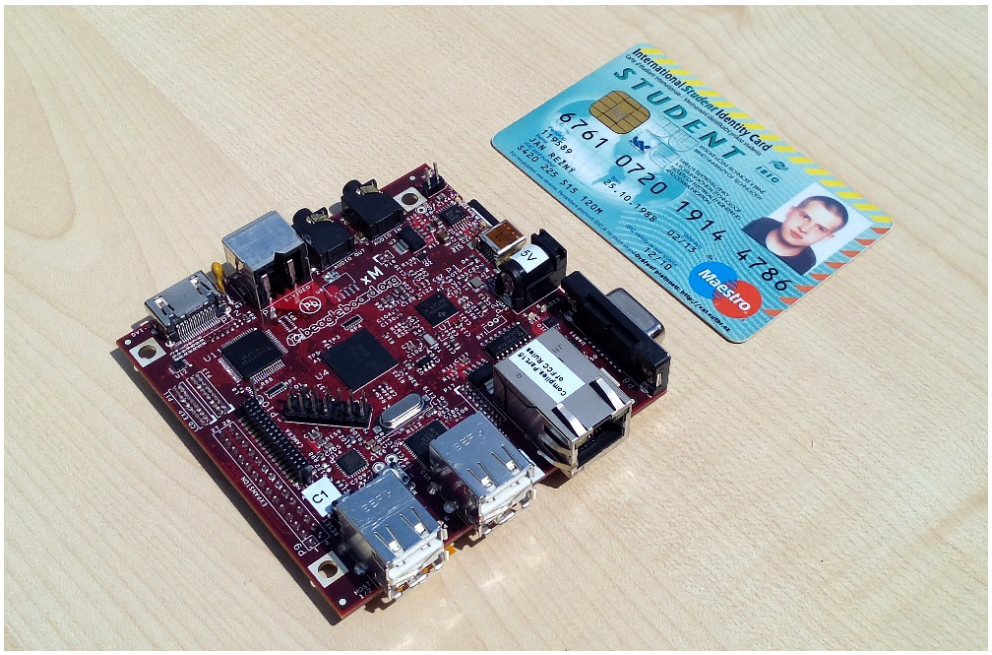
Obr. 1.2: Raspberry Pi

[4]General Purpose Input/Output

1.1.2 Beagle Board

Beagle Board je další ze sérií jednodeskových počítačů určených pro výuku a výzkum. Předností desek BeagleBoard je přítomnost DSP[5] procesoru pro zpracování multimédií, v této analýze však využití tohoto procesoru není plánováno. Ze série Beagle Board byla pro tuto analýzu použita verze Beagle Board xM. Oproti původní verzi nabízí rychlejší CPU a větší RAM paměť (CPU Texas Instruments DM3730 taktovaný na 1GHz a 512MB RAM). Interní paměť zde není přítomna, operační systém (zde Angstrom OS) se tedy zavádí z paměťové karty MicroSD.

Deska Beagle Board xM má rozměry 8,3x8,3cm, kromě bežných konektorů osazovaných na podobných minipočítačích (USB, RJ-45, HDMI) jsou zde přítomny konektory RS232 (COM) a S-video, další periferie lze připojit pomocí GPIO pinů. Napájení je zde zajištěno buď kulatým 5V konektorem, nebo konektorem MiniUSB.



Obr. 1.3: BeagleBoard xM

[5]Digitální signálový procesor, mikroprocesor optimalizovaný pro zpracování digitálních signálů

1.1.3 Parallella - ARM

Parallella je jednodeskový minipočítač vyvinutý za účelem zpřístupnění multijádrového systému co největšímu okruhu vývojářů. Tento minipočítač se stal celosvětově známým svou úspěšnou kampaní na crowdfundingovém serveru Kickstarter[6], kde dokázal za jeden měsíc získat od širokého spektra přispěvatelů téměř 900 000 USD na jeho vývoj a výrobu.

Jako základ tohoto minipočítače byl zvolen čip Zynq Z7020 kombinující dvoujádrový CPU ARM Cortex-A9 a FPGA[7] pole. Kromě tohoto čipu je deska osazena koprocesorem Epiphany-III, díky kterému je tato deska velice zajímavá pro paralelní zpracování dat. Více informací o architektuře Epiphany je uvedeno v následující kapitole. Deska je osazena 1GB RAM, pro potřeby komunikace mezi koprocesorem Epiphany a hlavním CPU je vyhrazeno 32MB sdílené paměti.

Parallella disponuje kromě konektorů RJ-45, MicroHDMI a MicroUSB také rozšiřujícími konektory GPIO dovolující připojení dalších desek nebo příslušenství. Napájení je možno řešit pomocí napájecího 5V konektoru nebo konektoru MicroUSB, jako úložiště dat je zde využito opět paměťové karty standardu MicroSD. Operačním systémem je zde Linuxová distribuce OS Ubuntu.

[6]Crowdfunding - způsob financování velkým počtem přispěvatelů po malých obnosech

[7]Field Programmable Gate Array - programovatelné hradlové pole. Speciální různě složité obvody s možností programově měnit konfiguraci propojení

1.2 Architektura Epiphany

Epiphany je skalární vícejádrová architektura vyvíjená společností Adapteva. Architektura dovoluje využít síť až 4096 jader - uzlů s RISC mikroprocesory sdílející jeden 32 bitový adresový prostor. Každý mikroprocesor může pracovat na frekvenci až 1 GHz, využívá svou vlastní instrukční sadu optimalizovanou pro vysoký výkon při výpočtech s plovoucí desetinnou čárkou. K dispozici je velmi dobře zdokumentované Epiphany SDK[8] dovolující programovat v jazyce C s využitím nástroje kompatibilního s GCC[9].

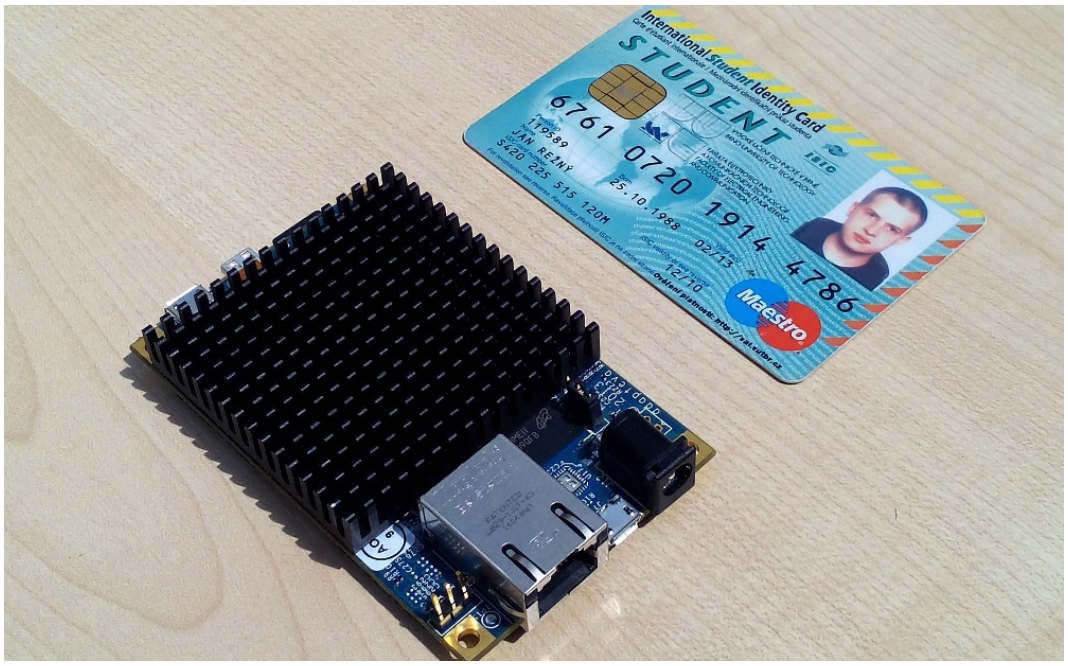
Každý uzel - jádro v síti architektury Epiphany obsahuje již výše zmíněný mikroprocesor, rychlou vnitřní paměť RAM (32KB), obvody DMA[10] a obvody zajišťující rychlou komunikaci s okolními jádry.

V současné době firma Adapteva vyrábí dva koprocesory založené na architektuře Epiphany, a to Epiphany-III (E16G301) a Epiphany-IV (E6G401). Hlavní rozdíl mezi koprocesory je v počtu jader (Epiphany-III má 16 jader, IV má jader 64), jejich maximální taktovací frekvenci (III - 1GHz, IV - 800MHz) a v technologii výroby (III - 65nm, IV - 28nm). Oba dva koprocesory mají výrobcem definovanou maximální spotřebu 2W.

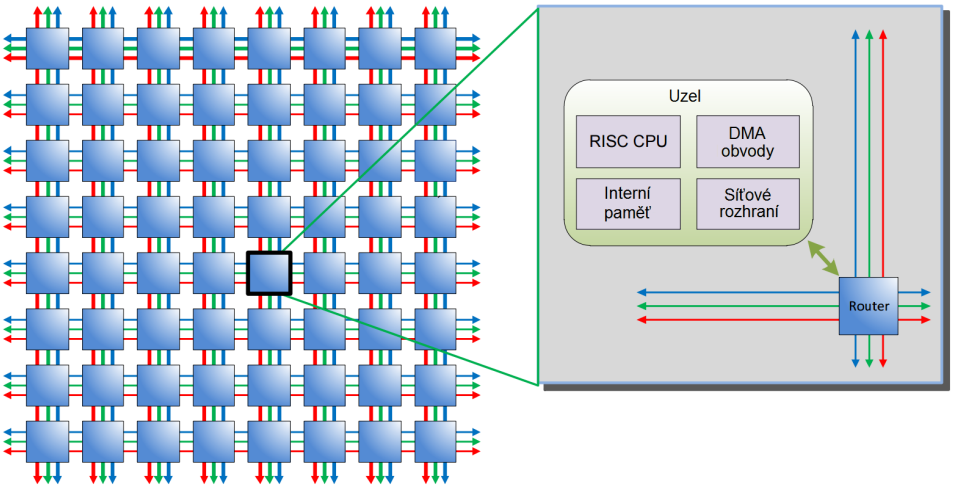
[8] Software Development Kit - sada nástrojů pro vývoj softwaru

[9] GNU Compiler Collection - sada překladačů programovacího jazyka C

[10] Direct Memory Access - způsob přímého přístupu do operační paměti kdy data nezpracovává procesor



Obr. 1.4: Parallella



Obr. 1.5: Architektura Epiphany, zdroj - Adapteva[4]

1.2.1 Parallella - Epiphany

Samotná deska je popsána v předchozí kapitole, kromě hlavního čipu Zynq se zde nachází již výše zmíněný koprocesor Epiphany-III. Přítomnost tohoto koprocesoru je hlavní předností této desky, deska představuje snadno dostupnou platformu pro vývoj, testování i produkci řešení využívajících ve velké míře paralelní výpočty, které byly do nedávné doby možny zajistit pouze superpočítači nebo výpočetními sítěmi.

Taktovací frekvence koprocesoru Epiphany-III je dle specifikací výrobce až 1GHz, na desce Parallella běží koprocesor na 600 MHz.

Jako velkou výhodu tohoto minipočítače výrobce uvádí velmi nízkou spotřebu koprocesoru Epiphany (25mW na jádro, celkově max. 2W) a zároveň vysoký výkon těchto jader ve výpočtech s plovoucí desetinnou čárkou.

1.3 Architektura x86

Jako doplněk k platformám založených na architektuře ARM byly vybrány také některé desky obsahující procesory architektury x86. Tato architektura má základ v 80.letech 20.století, instrukční sada x86 navazuje na mikroprocesory Intel 8086, 8088 a jejich varianty. Tato instrukční sada dala název celé architektuře a platformě označované jako IBM PC kompatibilní. CPU využívající instrukční sadu x86 se řadí mezi CISC, i když v současné době se rozdíl mezi nimi stírá (některé moderní CISC procesory jsou interně RISC, a běžné CISC instrukce jsou pak řešeny několika RISC instrukcemi).

Firma Intel intenzivně vyvíjí řady procesorů architektury x86, v současné době však má velkou konkurenci v podobě procesorů ARM kde soupeří jak v oblastech výkonu, tak i spotřeby a ceny. Řada procesorů Intel Atom tak nabízí zajímavý poměr ceny, spotřeby a výkonu jak pro mobilní zařízení tak pro levné jednodeskové počítače.

1.3.1 Intel NUC (Atom E3815)

Řada minipočítačů Intel NUC (Next Unit of Computing) je představená jako referenční model nastávající generace osobních počítačů PC. První varianty Intel NUC byly osazeny CPU Intel Celeron, dražší varianty jsou osazeny CPU Intel Core i3 nebo i5. Nejlevnější varianta Intel NUC je osazena jednojádrovým procesorem Intel Atom E3815 taktovaným na frekvenci 1,46 GHz. Součástí CPU je i grafický čip Intel HD Graphics, TDP procesoru je pouze 5W, je tedy možné jej provozovat pouze s pasivním chladičem.

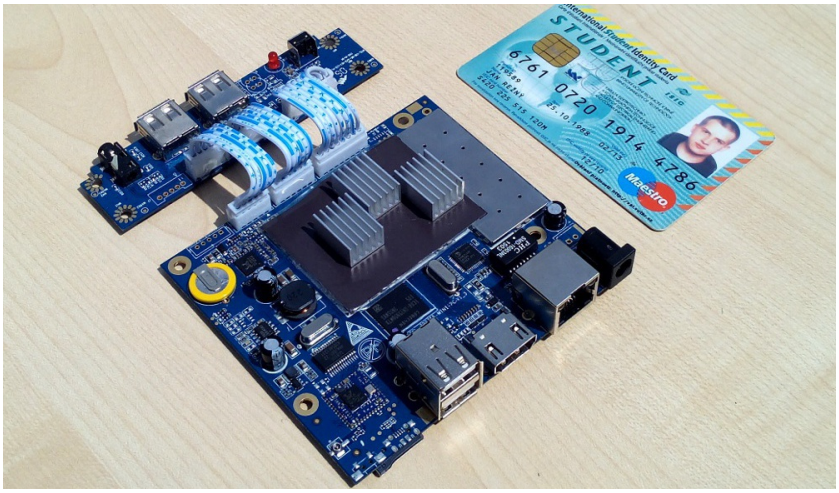
Deska počítače Intel NUC má rozměry 10x10cm, nabízí bohaté možnosti připojení nejrůznejších zařízení - na desce jsou přítomny konektory HDMI, RJ-45, SATA, MiniPCIe, USB 3.0, jsou vyvedeny piny umožňující připojení sériového portu, VGA monitoru nebo jiných zařízení prostřednictvím GPIO. Napájení desky je zajištěno 12V napájecím kulatým konektorem, nebo průmyslovým 4 pinovým napájecím konektorem Molex.
Na této desce není integrovaná paměť RAM, místo ní je zde SO-DIMM slot. Paměť RAM je tedy uživatelsky vyměnitelná a je možné tuto desku osadit přiměřenou velikostí RAM podle využití, pro tuto analýzu byla deska osazena 4GB DDR3L modulem. Přítomnost SATA konektoru předpokládá připojení pevného disku, operační systém však lze nainstalovat i na interní 4GB eMMC[11] modul.

Obr. 1.6: Intel NUC

[11]Embedded MMC - typ paměti určené pro ukládání dat

1.3.2 Pipo X7 (Atom Z3736F)

Pipo je čínská společnost zaměřující se na výrobu tabletů a „TV boxů“ založených na ARM procesorech Rockchip a x86 procesorech Intel Atom, tento konkrétní model je založen na procesoru Intel Atom Z3736F.
Pipo X7 je zařízení určené primárně pro koncové uživatele, je zde přítomno např. rozhraní Bluetooth nebo WLAN 802.11 b/g/n, což nebývá u jednodeskových počítačů obvyklé. Naopak zde chybí GPIO piny, což není vzhledem k původnímu zaměření tohoto zařízení překvapující.
Základní deska minipočítače Pipo X7 je velká přibližně 8,5x9,5cm, obsahuje konektory USB, HDMI, RJ-45 a micro USB slot. Napájení je zajištěno 12V kulatým konektorem. Kromě základní desky je přítomna i oddělitelná rozšiřující deska obsahující další USB konektory a konektor 3,5mm Jack.
Procesor Intel Atom Z3736F je založen na stejné mikroarchitektuře Silvermont jako předchozí E3815, je však čtyřjádrový a taktován na frekvenci 1.33 GHz. SDP[12] tohoto procesoru je udáváno 2.2W, procesor dokonce nemá ani výraznější pasivní chladič, jako chladič zde slouží stínící plech okolo CPU a RAM. Z preventivních důvodů proto byly na desku nalepeny malé hliníkové chladiče. Deska je osazena 2GB DDR3L RAM přímo na desce. Součástí základní desky je 32GB eMMC modul sloužící jako SSD[13], na kterém je od výrobce předinstalován operační systém Windows 8.1 with Bing.



Obr. 1.7: Pipo X7

[12]Scenario Design Power - oproti TDP se jedná o typickou spotřebu při běžném zatížení která však může být překročena
[13]Solid State Disc, pevný disk bez pohyblivých částí

1.3.3 Notebook (Core i5 4200U)

Jako doplněk k platformám založeným na procesorech Intel Atom byly pro analýzu vybrány také notebook střední třídy obsahující CPU Intel Core i5 4200U a stolní počítač popsaný níže.

Procesory Intel Core řady U jsou všeobecně známy pro svůj vysoký výkon a malou spotřebu, model 4200U je dvoujádrový CPU taktovaný na frekvenci 1,6GHz podporující technologii Intel HyperThreading. Součástí analýzy bude také sledování vlivu technologie HyperThreading na výkon při zpracování zvuku.

Technologie Intel HyperThreading zajišťuje vytvoření z jednoho fyzického procesoru dva virtuální, softwaru se tedy jeden fyzický CPU jeví jako dva logické CPU. HyperThreading umožňuje lépe využít hardware procesoru tím, že při zpracování určitých instrukcí dokáže zpracovat jiné další instrukce patřící jinému procesu. Tato technologie se jeví jako velmi efektivní hlavně při početních operacích.

Testovaný notebook je model Elitebook 820 značky HP, osazen 8GB DDR3L RAM. Konfigurace podobně výkonných notebooků se může značně lišit, proto je tento notebook v analýze uveden pouze jako CPU Core i5.

1.3.4 Stolní PC (Core i7 3770K)

Stolní PC bylo vybráno pro analýzu jako doplněk k notebooku a ostatním platformám architektury x86, pro porovnání možností dnešních technologií. Testovaný stolní PC je osazen procesorem nejvyšší řady dostupné pro běžná PC - Core i7 3770K taktovaný na frekvenci 3,5GHz. Oproti ostatním platformám je tento CPU projektován primárně na vysoký výkon, jeho spotřeba je tedy relativně vysoká. Předmětem testování tohoto procesoru je zjistit, zda CPU s takto vysokou spotřebou bude také při analýze adekvátně výkonný.

Procesor Core i7 3770K je čtyřjádrový s technologií HyperThreading, operačnímu systému se tedy jeví jako 8 logických procesorů. Stolní PC má základní desku Asus P8Z77-M a je osazen 16GB DDR3, zbývající parametry sestavy s při této analýze na celkovém výkonu neprojeví. Stolní PC je podobně jako notebook v analýze uváděn pouze jako CPU Core i7.

2 KODEK OPUS

Pro analýzu HW řešení bylo využito relativně nového ztrátového kodeku Opus navrženého pro interaktivní real-time využití na internetu. Formát Opus je vyvíjen od roku 2007, první verze tohoto kodeku byla uvolněna v září 2012. Jedná se o otevřený formát, kombinující formáty SILK (kódování např. řeči) a CELT (např. kódování hudby). Mezi těmito kompresními metodami dokáže plynule přecházet i při souvislém kódování jednoho zvukového proudu, a přitom zachovává nízkou latenci a vysokou kvalitu zvuku.

Formát Opus podporuje konstantní i variabilní datový tok (CBR[1],VBR[2]) 6-510kbit/s, velikosti rámců 2.5 až 60ms, vzorkovací frekvence od 8 do 48 kHz. Podporuje až 255 zvukových kanálů.

Výše zmíněná nízká latence při zpracování signálu (standardně 26.5ms) je vhodná pro použití např. v IP telefonii nebo videokonferencích. I při nízkých datových tocích je díky kombinaci SILK a CELT zachována vysoká kvalita záznamu oproti MP3 nebo jiným ztrátovým formátům při poslechových testech.

Díky těmto velmi dobrým vlastnostem a otevřenosti se podpora formátu Opus v softwarech pro koncové uživatele rychle rozšiřuje.

[1] Constant Bitrate, konstatní datový tok
[2] Variable Bitrate, proměnný datový tok

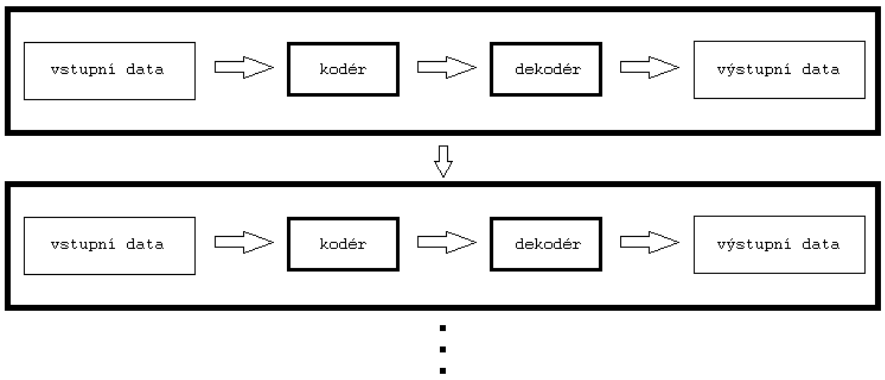
3 VYVÍJENÝ ANALYZAČNÍ SOFTWARE

Analyzační software pro porovnání HW řešení byl vyvíjen v jazyce C s využitím knihovny libopus, princip funkčnosti je naznačen v blokovém schématu v další kapitole.

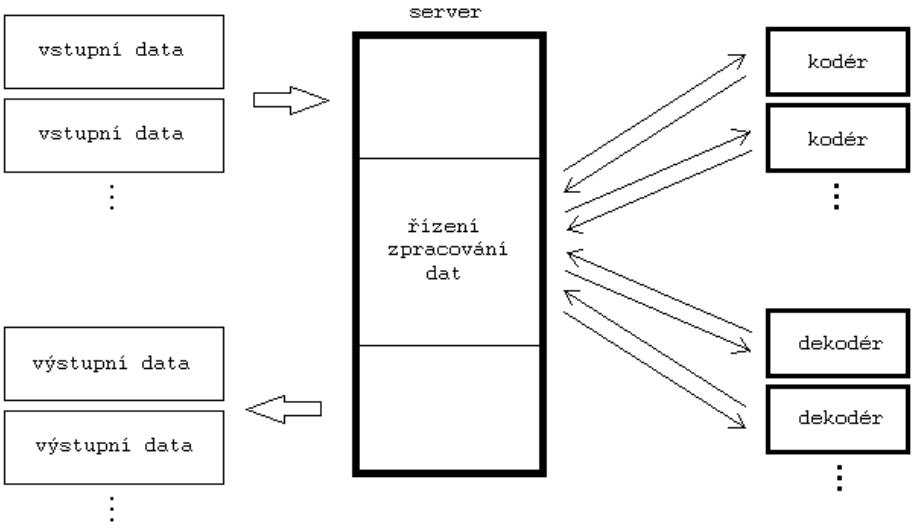
Při vyvíjení analyzačního softwaru bylo nejprve otestováno samotné využití knihovny pomocí dodávaných příkladů zdrojových kódů, a tyto byly poté využity jako základ pro analyzační aplikaci pro sériové zpracování (*opus_simple*). Tato aplikace byla navržena tak, aby postupně kódovala a dekódovala vstupní zvukový proud načtený ze souboru a výsledek uložila do výstupního souboru. Pro eliminaci zpoždění vlivem čtení a zápisu na disk je využito cachování vstupních i výstupních dat do RAM.

Podobný princip využívá i analyzační aplikace pro paralelní zpracování (*opus_server*) s tím rozdílem, že samotné kódování a dekódování je zajištěno dalšími dvěma aplikacemi (*opus_encode*, *opus_decode*). Aplikace *opus_server* zajišťuje čtení, zápis dat a řízení zpracování, *opus_encode* a *opus_decode* jsou spuštěny jako další procesy, kódování a dekódování jednotlivých zvukových proudů tedy probíhá odděleně.

3.1 Blokové schéma



Obr. 3.1: Blokové schéma - sériové zpracování



Obr. 3.2: Blokové schéma - paralelní zpracování

3.1.1 Pseudokód řízení paralelního zpracování

```
int main()
{
x=argument_count();
file[0-x]=load_file(argument[0-x]);
output_buffer[0-x]=allocate_memory();
start_encoders(x)
start_decoders(x)
chunk[0-x]=0;
while (1)
 {
   data=receive_packet();
   switch (data.type)
   {
     case (encoded_data)
        {
          send_to_decoder(data);
          break;
        }
     case (decoded_data)
        {
          save_to_buffer(file[data.file_id],data.value,chunk[data.file_id]);
          chunk[i]++;
          load_from_file(file[data.file_id],data.value,++chunk[data.file_id]);
          send_to_encoder(data);
          break;
        }
     case (encoder_ready)
        {
          load_from_file(file[data.file_id],data.value,++chunk[data.file_id]);
          send_to_encoder(data);
          break;
        }
     case (invalid_data)
        {
          break;
        }
   }
 }
return 1;
}
```

3.2 Popis problémů a jejich řešení

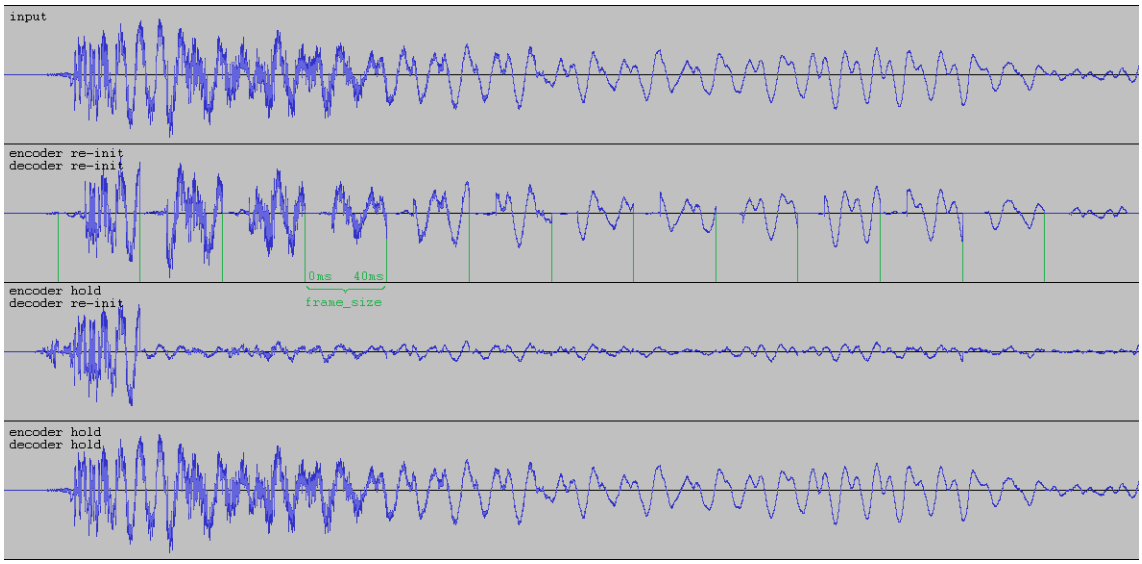
Při vývoji aplikace pro paralelní zpracování bylo zapotřebí zvolit metodu komunikace mezi jednotlivými procesy zajišťujícími čtení/zápis/řízení zpracování, kódování dat a dekódování dat. Jeden z prvních návrhů bylo využití pipe / emulace obousměrné pipe. Ukázalo se, že tento způsob není pro toto využití vůbec vhodný, řešení emulace dvousměrné pipe by bylo velmi náročné a neefektivní, bylo proto nutné zvolit odlišný způsob meziprocesní komunikace. Jako jednoduché a efektivní řešení bylo využití socketové komunikace, která je obsluhována linuxovým jádrem, a přenášení UDP[1] pakety.

První verze aplikace *opus_server* byla řešena tak, že v případě nutnosti kódování/dekódování dat došlo ke spuštění procesu *opus_encode* nebo *opus_decode*, vytvoření socketu, přenesení dat UDP paketem, zpracování dat, přenesení zpracování dat UDP paketem zpátky, uzavřením socketu a ukončením kódovacího/dekódovacího procesu. Takto se dalo efektivně využít zrovna volné jádro a zároveň možnost sledování komunikace např. pomocí programu Wireshark. Další výhodou mohlo být využití i jiného zařízení (UDP pakety by mohly inicializovat spuštění kodéru/dekodéru i na jiném zařízení a zajistit i následný přenos dat). Při testování tohoto způsobu aplikace však docházelo k tomu, že při velkém počtu kódovaných/dekódovaných částí narůstal počet využitých UDP portů a recyklace (znovupoužití) těchto portů nebylo často možné. Původ tohoto problému byl zjištěn právě v linuxovém jádře, kdy při vysoké rychlosti došlo k prodlevě a nedošlo tedy k okamžitému uzavření socketu a uvolnění portu. Zároveň se tento způsob komunikace ukázal jako neefektivní z důvodu velké prodlevy mezi spuštěním kódovací/dekódovací aplikace a získáním zpracovaných dat.

Aplikaci bylo tedy nutné přepracovat, další verze byla vyvinuta tak, že byl spuštěn předem stanovený počet kodérů/dekodérů a tyto procesy měly předem stanovený port na kterém naslouchaly. Řídící aplikace *opus_server* tedy zajišťovala předávání dat k zakódování/dekódování příslušným procesům. Komunikace se tímto značně zjednodušila a zrychlila, avšak při dalším praktickém testování bylo zjištěno, že zpracované zvukové proudy jsou určitým způsobem poškozeny, výstupní průběh signálu neodpovídal vstupnímu. Poslechovým testem zněla nahrávka přerušovaně, odpovídající výstupní signál tomu odpovídal. Experimentálním způsobem bylo po delší době zjištěno, že instance kodéru a dekodéru si pro jednotlivé zvukové proudy zane-

[1] User Datagram Protocol, protokol pro negarantovaný ale rychlý a jednoduchý přenos dat po IP síti

chává v paměti jistý kontext, proto je nutné jednu instanci kodéru/dekodéru využít pro zpracování jednoho zvukového proudu. V praxi toto tedy znamenalo upravit řízení zpracování tak, aby jeden spuštěný kodér a jeden dekodér byly využity pro jeden navazující signál. Průběhy vstupního a výstupního signálu jsou znázorněny na obr.3.3.



Obr. 3.3: Vstupní a výstupní signál analyzačního softwaru

Při praktickém testování docházelo na některých testovacích sestavách k dalšímu nepříjemnému jevu - po přenesení cca 2048 paketů se náhle komunikace zastavila, data byly sice od kodérů / dekodérů odeslány řídící aplikaci, ta je však nepřijme. Tento jev se např. na Raspberry Pi neprojevil, avšak na zařízeních s OS Ubuntu se jednalo o nepřekonatelný problém. Tento problém však nastal jen při zpracování více než 30 sekund záznamu, proto bylo při testování využito pouze 5 sekundových ukázek, a to maximálně 6 najednou.

3.3 Vyvinutá aplikace

Vyvinuté analyzační aplikace pro sériové i paralelní zpracování mají stejné vstupní parametry, identický je i jejich výstup v příkazové řádce. Aplikace se spouštějí s parametry *input1 output1 input2 output2 .....* Aplikace pro sériové zpracování *opus_simple* nejprve všechny vstupní soubory uloží do paměti RAM a postupně jeden po druhém po kouscích kóduje, dekóduje a uloží výstup. Aplikace pro paralelní zpracování *opus_server* všechny vstupní soubory uloží do RAM, spustí procesy kodérů a dekodérů *opus_encode* a *opus_decode* (pro každý soubor jeden kodér a jeden dekodér) a vyčká na potvrzovací zprávy nově spuštěných procesů. Poté začne postupně číst všechny soubory a po jednotlivých rámcích je posílat na kodéry. Kodér přijme rámec, zakóduje, pošle zpět a čeká na další rámec, obdobně pak dekodér. Řídící aplikace přijímá data od kodérů a postupně je posílá dekodérům, přijatá dekódovaná data uloží a načte další rámec který pošle kodéru. Takto se celý proces opakuje pro všechny soubory, dokud se nezpracují celé soubory.

Parametry kódování a dekódování nelze u zkompilovaných aplikací měnit, tyto parametry jsou zapsány ve zdrojových kódech aplikací *opus_simple* a *opus_server*. Pro účely analýzy v této diplomové práci tento způsob není zapotřebí měnit.

3.4 Modifikace pro koprocesor Epiphany

Jednotlivá jádra koprocesoru Epiphany jsou značně omezená, nešlo tedy zkompilovat kód přímo pro Epiphany bez provedení změn.
Kompilace samotné knihovny libopus bylo provedeno pomocí změn v konfiguraci makefile, byla provedena změna kompilátoru z gcc na e-gcc (upravený pro epiphany) a kompilace knihovny libopus jako statické.

Při kompilaci aplikace ovšem došlo k problému že nalinkovaná aplikace s knihovnou libopus se nevejde do interní paměti jádra v koprocesoru Epiphany, bylo tedy nutné patřičně upravit LDF soubor používaný při linkování zkompilované aplikace. Soubory LDF (Linker Description File) používané při linkování popisují v jaké paměti bude uložen kód aplikace a knihovny. Standardně dodávané LDF soubory počítaly se scénáři *internal* (nejrychlejší, avšak do vnitřní paměti se celá knihovna nevejde), *fast* (kód, uživatelská data a statické knihovny ve vniřní paměti, zbytek v externí), a *legacy* (interní paměť není využívaná, využívá se externí která je ovšem pomalá). Poslední scénář se zdál být vhodný, avšak objevily se problémy při využívání proměnných, které se alokovaly v externí paměti. Adresy těchto proměnných z neznámého důvodu nebyly správné, docházelo k problémům při přístupu k těmto proměnným, bylo proto nutné vytvořit vlastní LDF soubor. Jako výchozí LDF byl vybrán poslední scénář *legacy* který byl upraven tak, aby vytvářené proměnné byly alokovány ve vnitřní paměti jednotlivých jader.

Další problém nastal při pokusu o kompilaci řídící aplikace paralelního zpracování. Ukázalo se totiž, že Epiphany SDK nepodporuje žádnou IP funkcionalitu, tudíž bylo nutné změnit způsob meziprocesní komunikace. Tento problém však měl za následek nutnost přepsat analyzační software na komunikaci pomocí sdílené paměti.

Jádra Epiphany mají přístup ke dvěma typům paměti - interní a externí, k oběma je možný přímý přístup (vytvořením ukazatele v C). Interní paměť je velmi rychlá (cca 500MB/s, při využití DMA dosahuje rychlost zápisu až 1900MB/s), externí znatelně pomalejší (zápis okolo 150MB/s, čtení pouze 4MB/s), je tedy vhodné využívat pro výpočty interní paměť. Rychlost přístupu ARM procesoru Parallelly k interní paměti jednotlivých jader je pomalejší (okolo 15MB/s), je to však rychlejší než využívat pro komunikaci pouze externí paměť.

Při návrhu verze softwaru pro koprocesor Epiphany byla zvolena externí RAM pro signalizaci a interní RAM jednotlivých jader pro předávání dat pro zpracování. Při

návrhu aplikace bylo postupováno podle Epiphany SDK s využitím funkcí pro přístup k externí paměti i pro přístup k interním pamětem jader, při testování se však tuto komunikaci nepodařilo zprovoznit. Došlo zde k problému kdy jednotlivá jádra nejsou schopny přečíst obsah externí RAM určené k signalizaci, nepodařilo se tedy bohužel tento analyzační software na koprocesoru Epiphany spustit.

Možné příčiny tohoto problému jsou mnohonásobný přístup do paměti který se jeví ze strany ARM procesoru jako bezproblémový (zapsaná signalizační data jdou přečíst a jsou správná), ovšem jádra zde tyto zapsaná data nepřečtou. Další možná varianta je problém na straně Epiphany SDK, který se ovšem jeví jako nepravděpodobný.

Vzhledem k principu přenosu dat pomocí RAM nebylo možné zde nijak jednoduše spustit ani sériový analyzační software, nejsou tudíž k dispozici žádná data využitelná pro porovnání rychlosti zpracování zvuku pomocí knihovny libopus na jádrech Epiphany a rychlosti zpracování na ostatních platformách.

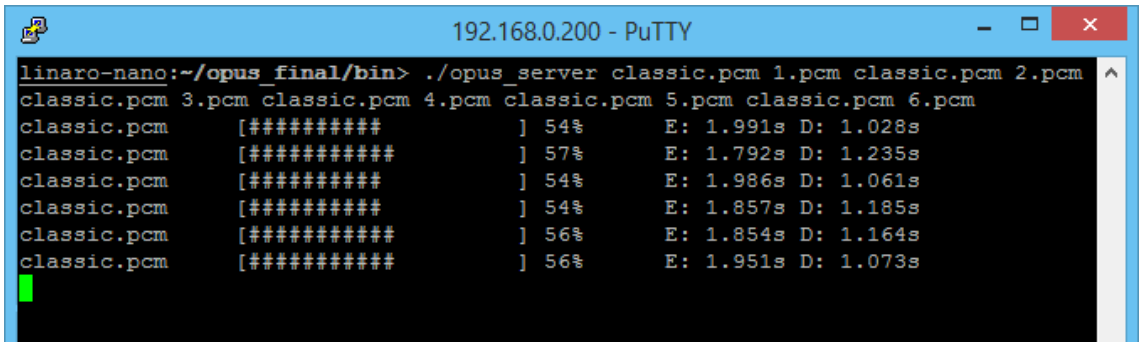
4 VÝKONOVÁ ANALÝZA PLATFOREM

4.1 Metodika testování

Při každém měření byl spuštěn analyzační software nejprve sériový a poté paralelní s různým počtem souborů ke zpracování. Aplikace sama měří dobu trvání zpracování každého souboru, zvlášť kódování, zvlášť dekódování a celkovou dobu zpracování souboru. Pro paralelní zpracování jednoho souboru se spustí zvlášť proces kódování a dekódování, kdy je ale pro každý soubor je v danou dobu aktivní pouze kodér nebo dekodér.

U sériového zpracování je relevantní údaj posledního souboru - udává celkový čas zpracování všech souborů. Při paralelním zpracování je relevantní údaj posledního zpracovaného souboru. Do doby zpracování je započítána i doba přenosu dat a doba zpracování paketů operačním systémem což do měření zanáší drobnou nepřesnost, která se ovšem projevuje až u velmi rychlých zařízení. Tato nepřesnost by se dala eliminovat odečtením doby přenosu - tato by se změřila testováním přenášení prázdných dat které by se nezpracovávaly, pouze by se měřila doba odeslání a přijetí paketu. Proto aby byla chyba měření co nejmenší, byla zvolena co největší délka rámce pro zpracování (frame_size=2880, při souboru vzorkovaném 48kHz se jedná o délku rámce 60ms).

Pro zpřesnění hodnot a eliminaci dočasných výkonových výkyvů byly procesy zpracování souborů opakovaně spuštěny celkem 10x a výsledné hodnoty zprůměrovány. Tyto hodnoty byly zapsány do tabulky a vybrané závislosti byly zhodnoceny a zaneseny do grafů.



Obr. 4.1: Příklad zobrazení průběhu testování

4.2 Naměřené hodnoty

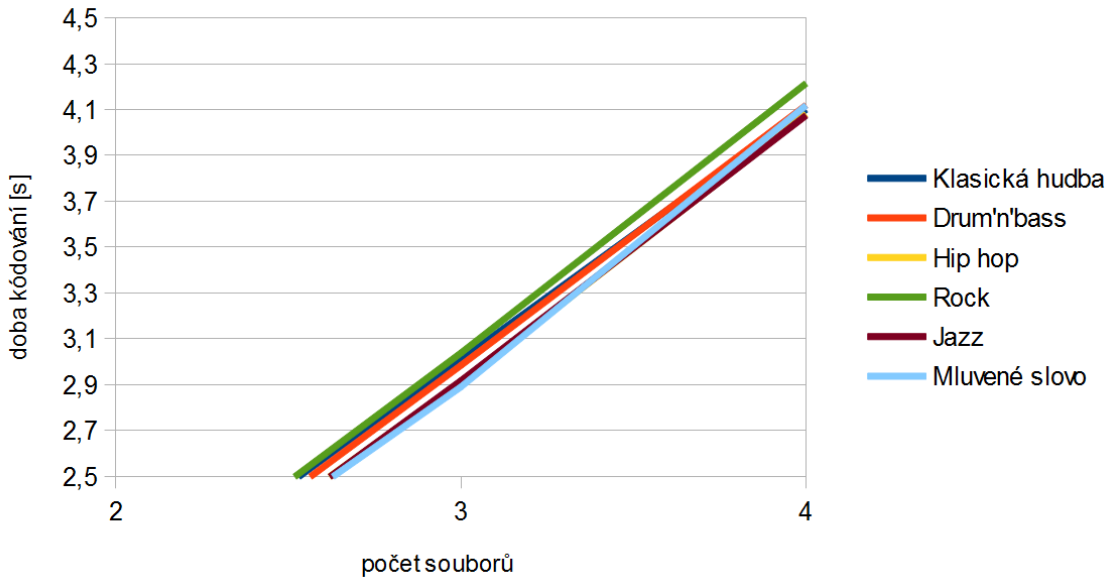
4.2.1 Závislost na vstupních datech

Před samotnou analýzou jednotlivých platforem bylo nutné zjistit závislost rychlosti zpracování na vstupních datech. Pro účely této analýzy bylo připraveno 5 ukázek hudebních stylů (zdroj - Free Music Archive[6]) a ukázka mluveného slova, každá o délce 30 sekund. Tyto ukázky byly zpracovány na testovaném notebooku (Core i5), naměřené hodnoty zapsány do tabulky 4.1 a zpracovány do grafu 4.2.

Z naměřených hodnot a grafu zpracování různých hudebních stylů a mluveného slova bylo zjištěno, že jak při kódování tak dekódování mají odlišná vstupní data minimální vliv na dobu jejich zpracování. Nejdéle trvalo zpracování rockové hudby a nejkratší dobu trvalo zpracování ukázky jazzu. Rozdíly však byly zanedbatelné, bylo tedy rozhodnuto použít pro porovnání jednotlivých platforem ukázku klasické hudby, jejíž doba zpracování se blížila průměru doby zpracování všech hudebních stylů.

Tab. 4.1: Závislost doby kódování na vstupních datech

	1 soubor	**2 soubory**	**3 soubory**	**4 soubory**
Klasická hudba	0,899 s	1,923 s	3,011 s	4,100 s
Drum'n'bass	0,913 s	1,873 s	2,985 s	4,119 s
Hip hop	0,874 s	1,830 s	2,911 s	4,080 s
Rock	0,894 s	1,919 s	3,039 s	4,214 s
Jazz	0,865 s	1,816 s	2,918 s	4,074 s
Mluvené slovo	0,863 s	1,836 s	2,891 s	4,118 s



Obr. 4.2: Závislost doby kódování na vstupních datech

4.3 Porovnání platforem

Při analýze výkonu jednotlivých platforem bylo provedeno několik měření na každé platformě, měřila se doba zpracování 5 sekundové ukázky klasické hudby. Při sériovém zpracování byla měřena doba zpracování jednoho až čtyřech souborů po sobě, při paralelním zpracování byla měřena doba zpracování jednoho až šesti souborů najednou.

4.3.1 Rychlost kódování - všechny platformy

Při porovnávání všech platforem najednou byly zvoleny pouze naměřené hodnoty při kódování.

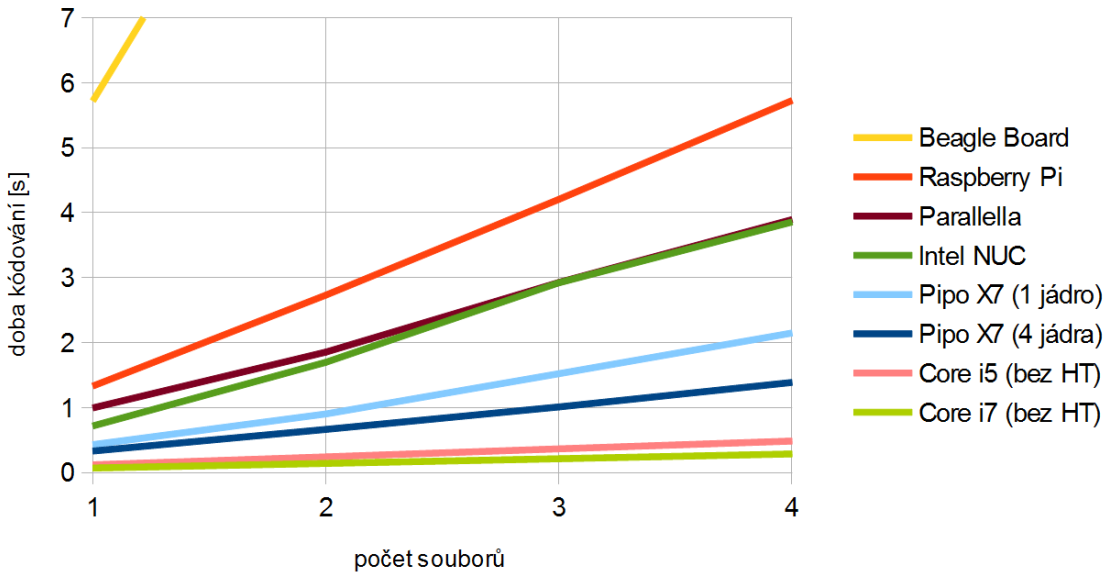
Sériové zpracování

Z grafu naměřených hodnot4.3 při sériovém kódování můžeme vidět na první pohled patrný nižší výkon platforem založených na architektuře ARM, zajímavé je především porovnání výsledku ARM procesoru na desce Parallella a x86 procesorů na Intel NUC a Pipo X7. Výkon Parallelly a NUC je velmi podobný, Parallella má ale procesor taktovaný na 1GHz, procesor Intel má frekvenci 1,46 GHz. Na zařízení Pipo X7 je zase zajímavé že i přes nižší frekvenci než Intel NUC je při aktivovaném

pouze 1 jádru rychlejší. Nepatrně vyšší výkon Pipo X7 při aktivovaných 4 jádrech je zřejmě způsoben rozdělením ostatních běžících procesů na jiná jádra než na kterém zrovna probíhalo kódování.
Velmi nízký výkon Beagle Board je překvapivý, je tak nízký že v některých grafech není vůbec uveden aby změna měřítka nezkreslovala zobrazení výkonu ostatních zařízení.

Tab. 4.2: Závislost doby kódování na platformě - sériové zpracování

	1 soubor	**2 soubory**	**3 soubory**	**4 soubory**
Parallella	0,995 s	1,854 s	2,926 s	3,894 s
Raspberry Pi	1,332 s	2,732 s	4,206 s	5,727 s
Beagle Board	5,722 s	11,622 s	18,039 s	24,613 s
Intel NUC	0,718 s	1,697 s	2,922 s	3,857 s
Pipo X7 (1 jádro)	0,430 s	0,902 s	1,522 s	2,148 s
Pipo X7 (4 jádra)	0,331 s	0,665 s	1,011 s	1,387 s
Core i5 (bez HT)	0,119 s	0,241 s	0,365 s	0,485 s
Core i7 (bez HT)	0,070 s	0,141 s	0,213 s	0,287 s



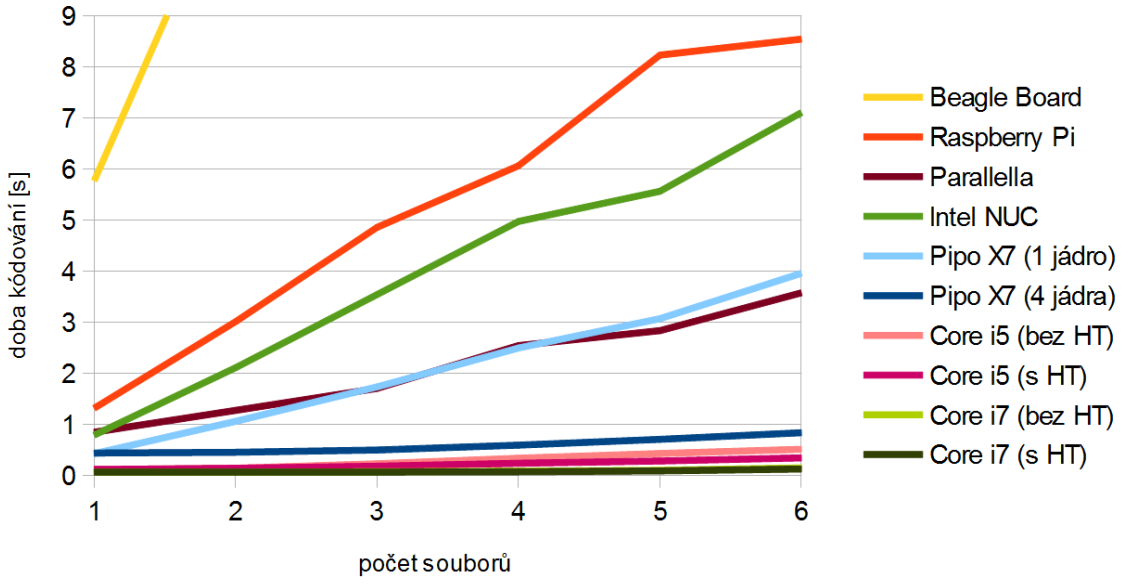
Obr. 4.3: Závislost doby kódování na platformě - sériové zpracování

Paralelní zpracování

Při paralelním kódování je znatelný nárust výkonu u vícejádrových procesorů. Zejména podobné výkony Parallelly a Intel NUC při sériovém zpracování se zde již neobjevují, Parallella je dle dle naměřených hodnot4.3 očekávání zhruba 2x rychlejší než podobně výkonný Intel NUC pouze s jedním procesorovým jádrem.

Tab. 4.3: Závislost doby kódování na platformě - paralelní zpracování

	1 soub.	**2 soub.**	**3 soub.**	**4 soub.**	**5 soub.**	**6 soub.**
Parallella	0,853 s	1,280 s	1,710 s	2,547 s	2,839 s	3,579 s
Raspberry Pi	1,323 s	3,016 s	4,859 s	6,065 s	8,228 s	8,541 s
Beagle Board	5,772 s	12,054 s	18,684 s	23,764 s	29,824 s	36,893 s
Intel NUC	0,797 s	2,115 s	3,544 s	4,975 s	5,565 s	7,103 s
Pipo X7 (1 j.)	0,433 s	1,065 s	1,739 s	2,500 s	3,073 s	3,961 s
Pipo X7 (4 j.)	0,445 s	0,460 s	0,505 s	0,599 s	0,713 s	0,843 s
Core i5 (bez HT)	0,120 s	0,147 s	0,235 s	0,344 s	0,436 s	0,519 s
Core i5 (s HT)	0,124 s	0,148 s	0,187 s	0,245 s	0,290 s	0,351 s
Core i7 (bez HT)	0,072 s	0,074 s	0,080 s	0,090 s	0,107 s	0,155 s
Core i7 (s HT)	0,073 s	0,073 s	0,074 s	0,080 s	0,095 s	0,132 s



Obr. 4.4: Závislost doby kódování na platformě - paralelní zpracování

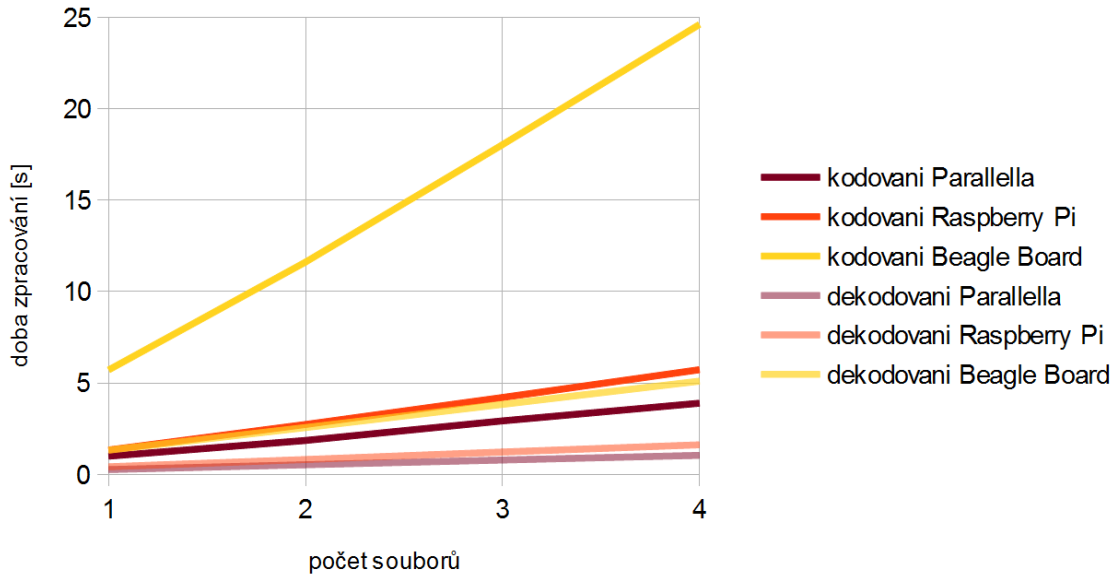
4.3.2 Srovnání platforem architektury ARM

Sériové zpracování

Při sériovém zpracování je z grafu4.5 evidentní očekávaný lineární průběh doby zpracování, zajímavý je ovšem rozdíl ve výkonu kódování a dekódování na desce Beagle Board, velmi nízký výkon ve srovnání s ostatními platformami při kódování zde zachraňuje alespoň uspokojivý výkon při dekódování.

Tab. 4.4: Závislost doby zpracování na platformě - sériové zpracování, platforma ARM

		1 soubor	**2 soubory**	**3 soubory**	**4 soubory**
kódování	**Parallella**	0,995 s	1,854 s	2,926 s	3,894 s
	Raspberry Pi	1,332 s	2,732 s	4,206 s	5,727 s
	Beagle Board	5,722 s	11,622 s	18,039 s	24,613 s
dekódování	**Parallella**	0,262 s	0,523 s	0,785 s	1,041 s
	Raspberry Pi	0,412 s	0,813 s	1,220 s	1,616 s
	Beagle Board	1,328 s	2,565 s	3,832 s	5,099 s



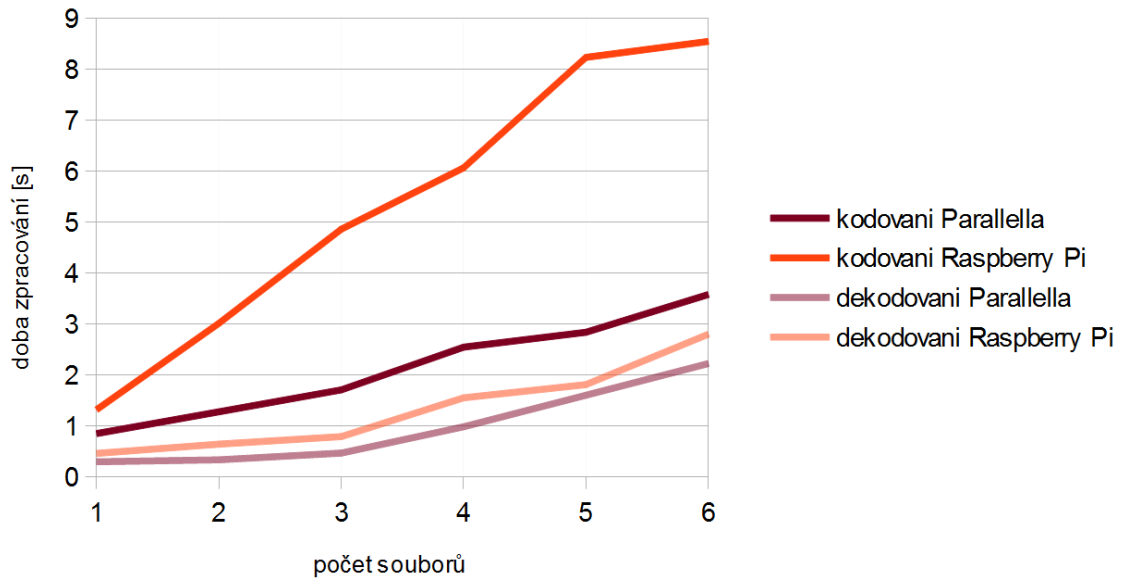
Obr. 4.5: Závislost doby zpracování na platformě - sériové zpracování, platforma ARM

Paralelní zpracování

Při paralelním zpracování není v grafu4.6 vyobrazen průběh časů zpracování na Beagle Board, je zde ale patrný předpokládaný výkonový náskok při kódování na desce Parallella s dvoujádrovým procesorem.
Zajímavostí je podobný výkon při dekódování, to zřejmě probíhá tak rychle, že se zde stírá rozdíl mezi jednojádrovým a dvoujádrovým procesorem.

Tab. 4.5: Závislost doby zpracování na platformě - paralelní zpracování, platforma ARM

		1 soub.	**2 soub.**	**3 soub.**	**4 soub.**	**5 soub.**	**6 soub.**
kód.	**Parallella**	0,853 s	1,280 s	1,710 s	2,547 s	2,839 s	3,579 s
	Raspberry Pi	1,323 s	3,016 s	4,859 s	6,065 s	8,228 s	8,541 s
dek.	**Parallella**	0,300 s	0,339 s	0,470 s	0,987 s	1,605 s	2,225 s
	Raspberry Pi	0,464 s	0,646 s	0,794 s	1,554 s	1,813 s	2,801 s



Obr. 4.6: Závislost doby zpracování na platformě - paralelní zpracování, platforma ARM

4.3.3 Srovnání platforem architektury x86

Srovnání platforem s procesory x86 bylo provedeno pouze při paralelním zpracování, byly porovnány zvlášť výkony procesorů Intel Atom a Intel Core.

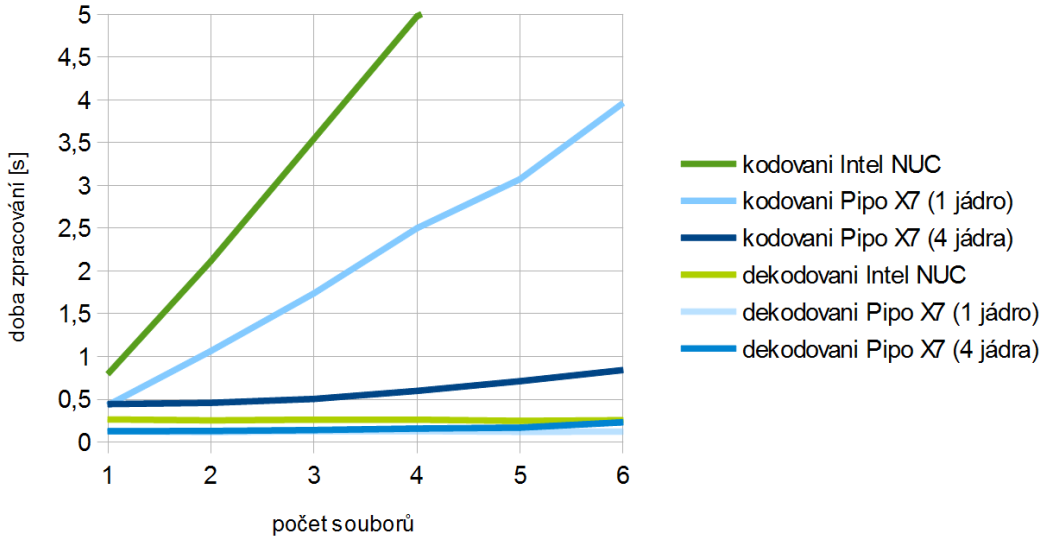
CPU Intel Atom

Při porovnání rychlostí kódování je v grafu4.7 velmi dobře vidět velmi vysoký výkon čtyřjádrového procesoru, kdy kódování probíhá až do 4 souborů najednou s velmi malým nárustem doby kódování, ta roste až při 5 a více souborech najednou. Při více souborech než 6 se dá předpokládat lineární průběh nárustu doby kódování.

Rychlost dekódování je zde velmi podobná bez ohledu na počet aktivních jader procesorů.

Tab. 4.6: Závislost doby zpracování na platformě - procesory Intel Atom

		1 soub.	**2 soub.**	**3 soub.**	**4 soub.**	**5 soub.**	**6 soub.**
	Intel NUC	0,797 s	2,115 s	3,544 s	4,975 s	5,565 s	7,103 s
kód.	**Pipo X7 (1 j.)**	0,433 s	1,065 s	1,739 s	2,500 s	3,073 s	3,961 s
	Pipo X7 (4 j.)	0,445 s	0,460 s	0,505 s	0,599 s	0,713 s	0,843 s
	Intel NUC	0,266 s	0,255 s	0,264 s	0,263 s	0,249 s	0,257 s
dek.	**Pipo X7 (1 j.)**	0,128 s	0,118 s	0,130 s	0,130 s	0,122 s	0,123 s
	Pipo X7 (4 j.)	0,128 s	0,131 s	0,142 s	0,159 s	0,170 s	0,233 s



Obr. 4.7: Závislost doby zpracování na platformě - procesory Intel Atom

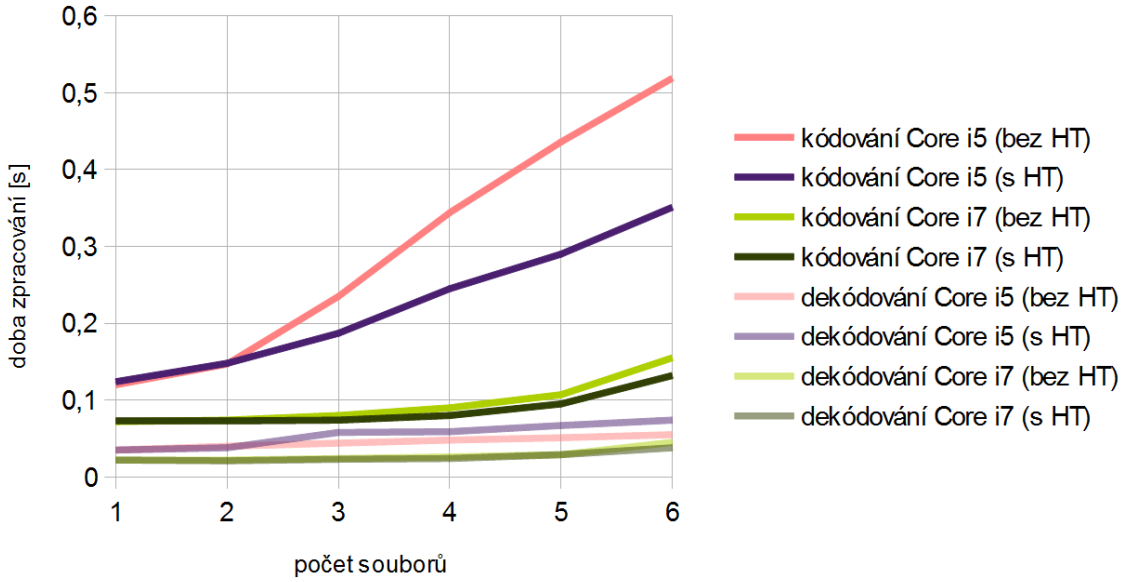
CPU Intel Core

Procesory Intel Core v použitých zařízeních jsou vícejádrové s podporou technologie Intel HyperThreading. Z grafu4.8 je velmi dobře patrný pozitivní vliv této technologie hlavně u času kódování na procesoru Core i5. Vliv u procesoru i7 není tak moc patrný, rozdíl by byl zřejmě větší při zpracování ještě většího počtu souborů.

Rychlost dekódování je zde velmi vysoká, avšak díky většímu rozlišení grafu i u dekódování lze vysledovat vliv počtu jader i technologie HyperThreading.

Tab. 4.7: Závislost doby zpracování na platformě - procesory Intel Core

		1 soub.	2 soub.	3 soub.	4 soub.	5 soub.	6 soub.
kód.	**Core i5 (bez HT)**	0,120 s	0,147 s	0,235 s	0,344 s	0,436 s	0,519 s
	Core i5 (s HT)	0,124 s	0,148 s	0,187 s	0,245 s	0,290 s	0,351 s
	Core i7 (bez HT)	0,072 s	0,074 s	0,080 s	0,090 s	0,107 s	0,155 s
	Core i7 (s HT)	0,073 s	0,073 s	0,074 s	0,080 s	0,095 s	0,132 s
dek.	**Core i5 (bez HT)**	0,035 s	0,040 s	0,044 s	0,048 s	0,051 s	0,055 s
	Core i5 (s HT)	0,035 s	0,038 s	0,058 s	0,059 s	0,067 s	0,074 s
	Core i7 (bez HT)	0,022 s	0,022 s	0,024 s	0,026 s	0,029 s	0,045 s
	Core i7 (s HT)	0,022 s	0,021 s	0,023 s	0,024 s	0,029 s	0,038 s



Obr. 4.8: Závislost doby zpracování na platformě - procesory Intel Core

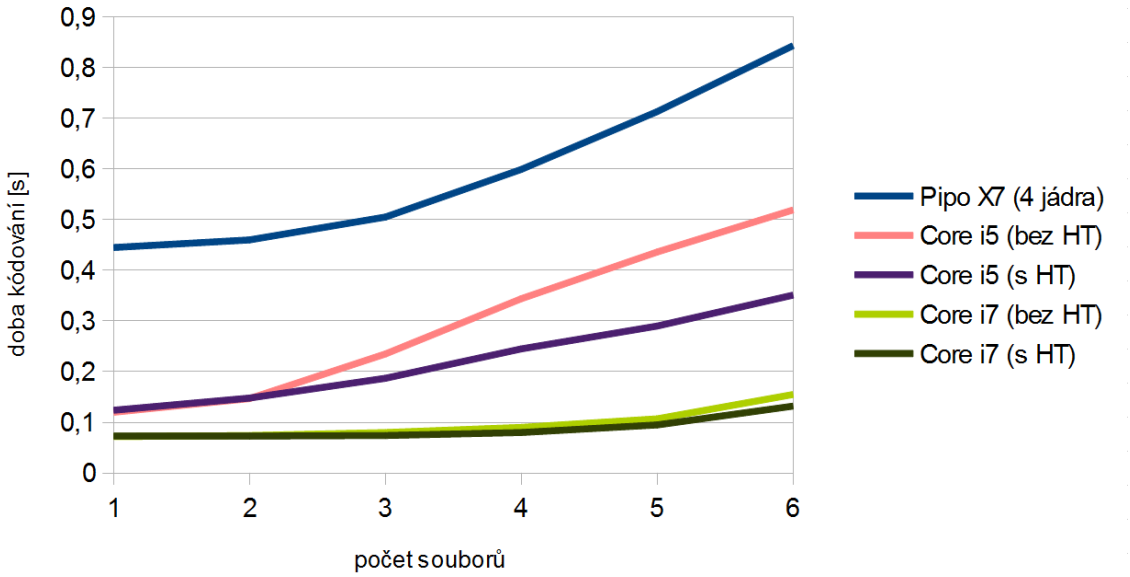
Srovnání vícejádrových procesorů Intel

Výkon čtyřjádrového procesoru Intel Atom Z3736F je při kódování obdivuhodný, časy zpracování testovacích hudebních ukázek se blíží mnohem dražším procesorům Intel Core.

Z detailnějšího grafu naměřených hodnot4.9 při srovnání výkonu kódování vícejádrových procesorů Intel je však patrná ještě jedna zajímavost - při zpracování jednoho souboru je zde procesor Core i5 téměř 4x rychlejší, při zpracování 4 souborů je rychlejší uz pouze cca 2,4x i při použití technologie HyperThreading. Je možné, že zde má vliv technologie Intel TurboBoost, která dokáže dočasně procesor přetaktovat i výrazně výše nad základní frekvenci, zejména při zátěži pouze jednoho jádra (u tohoto procesoru je to přetaktování jednoho jádra z 1,6GHz až na frekvenci 2,6 GHz)

Tab. 4.8: Závislost doby kódování na platformě - vícejádrové procesory Intel

	1	**2**	**3**	**4**	**5**	**6**
Pipo X7 (4 jádra)	0,445 s	0,460 s	0,505 s	0,599 s	0,713 s	0,843 s
Core i5 (bez HT)	0,120 s	0,147 s	0,235 s	0,344 s	0,436 s	0,519 s
Core i5 (s HT)	0,124 s	0,148 s	0,187 s	0,245 s	0,290 s	0,351 s
Core i7 (bez HT)	0,072 s	0,074 s	0,080 s	0,090 s	0,107 s	0,155 s
Core i7 (s HT)	0,073 s	0,073 s	0,074 s	0,080 s	0,095 s	0,132 s



Obr. 4.9: Závislost doby kódování na platformě - vícejádrové procesory Intel

4.3.4 Srovnání podle spotřeby

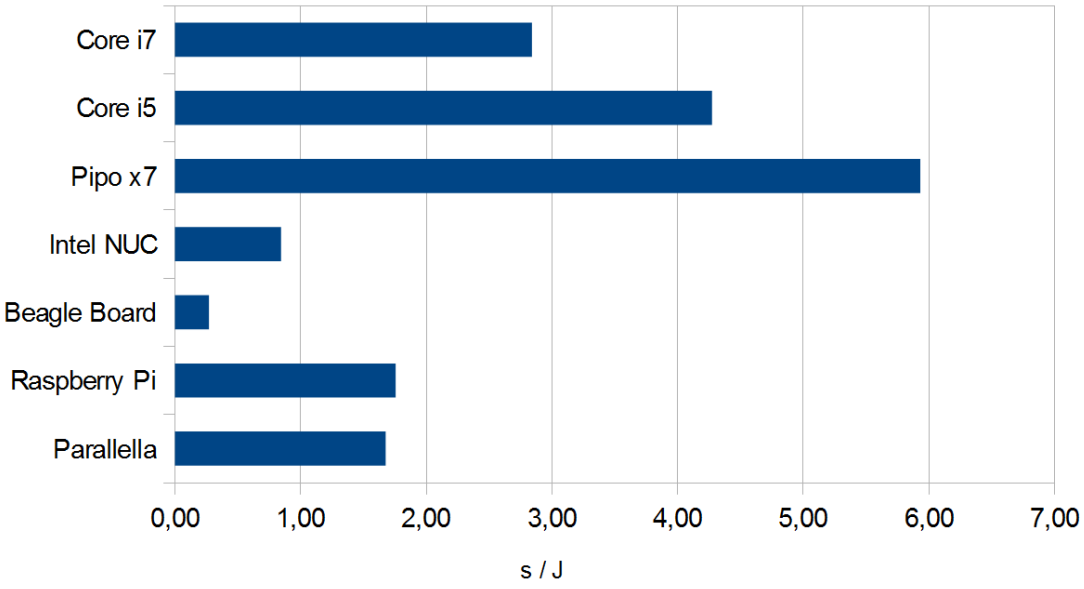
Na analyzovaných platformách bylo provedeno měření jejich příkonu pomocí laboratorního zdroje, naměřené hodnoty jsou uvedeny v tabulce 4.9.

Tab. 4.9: Naměřený příkon jednotlivých platforem

	napájecí napětí	**odebíraný proud**	**spočítaný příkon**
Parallella	5V	1A	**5W**
Raspberry Pi	5V	0,4A	**2W**
Beagle Board	5V	0,6A	**3W**
Intel NUC	12V	0,4A	**5W**
Pipo X7	12V	0,5A	**6W**

Tab. 4.10: Srovnání platforem podle spotřeby

	doba kód. 30s ukázky	**zakódováno za 1s**	**zakódováno za 1Ws**
Parallella	3,579 s	8,382 s	1,676 s
Raspberry Pi	8,541 s	3,512 s	1,756 s
Beagle Board	36,893 s	0,813 s	0,271 s
Intel NUC	7,103 s	4,224 s	0,845 s
Pipo x7	0,843 s	35,587 s	5,931 s
Core i5	0,351 s	85,470 s	4,274 s
Core i7	0,132 s	227,273 s	2,841 s



Obr. 4.10: Srovnání platforem podle spotřeby

Z nejlepších naměřených hodnot při kódování 30 sekund ukázky pro každou testovanou platformu bylo vypočítáno, kolik sekund zvuku dokáže za jednu sekundu zakódovat. Nejrychlejší je podle předpokladů stolní PC s procesorem Core i7, ovšem s vysokou spotřebou elektrické energie. Pokud je přihlédnuto ke spotřebě elektrické energie, nejlépe pak vychází zařízení Pipo X7 s procesorem Intel Atom Z3736F. Za 1 joule spotřebované energie (1 Watt/sekunda) dokáže zakódovat téměř 6 sekund zvukového záznamu, což je více než dvojnásobek nejrychlejšího procesoru Core i7.

5 ZÁVĚR

V rámci této diplomové práce byl nastudován kodek Opus a jeho implementace v jazyce C, vytvoření testovacího softwaru pro paralelní zpracování více datových toků v čase a otestování některých HW řešení které by byly pro zpracování zvukových signálů vhodné. Pro paralelní zpracování více zvukových signálů jsou vhodné vícejádrové procesory, čím větší počet jader, tím je výkonnost celku při zpracování větší. Významný přínos zde také přináší technologie Intel HyperThreading. V této diplomové práci bylo zjištěno, že nejnovější generace procesorů Intel Atom jsou výkonnější než procesory ARM při podobné spotřebě energie, jako nejefektivnější vyšel čtyřjádrový procesor Intel Atom Z3736F. Testované zařízení Pipo X7 s tímto procesorem má podobnou spotřebu jako testované jednodeskové zařízení s procesory ARM, podobnou cenu ale vyšší výkon. Bohužel se nepodařilo spustit analýzu na koprocesoru Epiphany, jeho výkon při paralelním zpracování dat by byl zřejmě vyšší než u ostatních testovaných zařízení.

6 OBSAH PŘILOŽENÉHO CD

readme.txt - obsah CD
opus_final.zip - zdrojové kódy + makefile pro x86
opus_arm.zip - zdrojové kódy + makefile pro ARM
opus_epiphany - zdrojové kódy + build.sh pro Parallellu
xrezny00.pdf - diplomová práce
xrezny00.ods - naměřené hodnoty - tabulky, grafy
xrezny00.xls - naměřené hodnoty - tabulky, grafy

LITERATURA

[1] *Epiphany SDK Reference* [online].
Dostupné z URL: `http://adapteva.com/docs/epiphany_sdk_ref.pdf`.

[2] *Epiphany Architecture Reference* [online].
Dostupné z URL: `http://adapteva.com/docs/epiphany_arch_ref.pdf`.

[3] *Memory on the parallella* [online].
Dostupné z URL: `https://github.com/coduin/epiphany-bsp/wiki/Memory-on-the-parallella`.

[4] *Adapteva - The Future of Parallel Computing* [online].
Dostupné z URL: `http://www.adapteva.com/introduction/`.

[5] *Opus audio codec: API and operations manual* [online].
Dostupné z URL: `http://www.opus-codec.org/docs/html_api-1.1.0/index.html`.

[6] *Free Music Archive* [online].
Dostupné z URL: `http://freemusicarchive.org/`.

[7] Brian W. Kernighan, Dennis M. Ritchie *Programovací jazyk C*
Brno: Computer Press, 2006. 288 s. ISBN: 80-251-0897-X