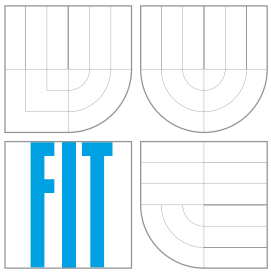
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

KOMUNIKACE USB3.0 ČIPU S FPGA

COMMUNICATION OF USB3.0 CHIP WITH FPGA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MATEJ ŠPEŤKO

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLASTIMIL KOŠAŘ

BRNO 2015

Abstrakt

SEC6NET je zkrácený název pro projekt Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace. Projekt je zaměřen na výzkum a vývoj prostředků monitorování síťového provozu a analýzu jeho záznamu. V rámci tohoto projektu jsou vyvíjená zařízení – sondy pro monitorování IPv6 sítí. Sondy využívají hardwarovou akceleraci pomocí FPGA. Moje práce spojuje dvě technologie: FPGA a USB. Cílem mojí práce je zabezpečení přenosu dat z FPGA čipu sondy do PC prostřednictvím mikrokontroléru Cypress EZ-USB FX3. V první části řeším přenos dat z FPGA do mikrokontroléru FX3. Druhá část popisuje úpravu firmware mikrokontroléru FX3 pro dosáhnutí maximální propustnosti. Poslední část řeší tvorbu PC aplikace pro operační systém Linux. Aplikace přijímá ze sondy zachycená data přes USB rozhraní a ukládá je na pevný disk ve formátu PCAP.

Abstract

SEC6NET is an abbreviation for project Modern tools for detection and mitigation of cyber criminality on the New Generation Internet. Project is focused on research and development of means for monitoring and analysing the network flow. Probes for monitoring IPv6 networks are developed within this project. Probes are using hardware acceleration based on FPGA platform. My thesis connects two technologies: FPGA and USB. The goal is to transfer data from the FPGA microchip to PC using Cypress EZ-USB FX3 microcontroller. The first part is focused on transferring data from FPGA to FX3 microcontroller. Second part describes the modification of FX3 firmware for getting maximum throughput. The last part focuses on implementing PC application for Linux operating system. The application receives data from the probe and saves them into hard drive using PCAP format.

Klíčová slova

FPGA, USB, USB 3.0, mikrokontrolér, mikrosonda, SEC6NET, firmware, PCAP, síťový provoz, Cypress EZ-USB FX3, GPIF II, paket, vlákno, synchronizace.

Keywords

FPGA, USB, USB 3.0, microcontroller, microprobe, SEC6NET, firmware, PCAP, network flow, Cypress EZ-USB FX3, GPIF II, packet, thread, synchronization.

Citace

Matej Špeťko: Komunikace USB3.0 čipu s FPGA, bakalářská práce, Brno, FIT VUT v Brně, 2015

Komunikace USB3.0 čipu s FPGA

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vlastimila Košaře.
Další informace mi poskytl Ing. Pavol Korček.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Matej Špeťko
31. července 2015

Poděkování

Ďakujem predovšetkým môjmu vedúcemu práce Ing. Vlastimilovi Košařovi za pripomienky, odborné rady, venovaný čas a pomoc pri spracovávaní tejto bakalárskej práce. Taktiež ďakujem Ing. Pavlovi Korčekovi za odbornú pomoc, konzultácie a poskytnuté informácie.

Obsah

1 Úvod **4**

2 USB **6**
2.1 Verzie a rýchlosti . . . 7
2.2 Topológia . . . 7
2.3 Protokol . . . 8
2.3.1 Rúry, endpointy . . . 8
2.3.2 Protokolová vrstva . . . 8
2.3.3 Typy prenosov – dátových tokov, paketov . . . 8
2.3.4 Fyzická a linková vrstva . . . 9
2.3.5 Detekcia novo pripojeného zariadenia – „*Enumeration*“ . . . 10
2.4 USB On-The-Go (OTG) . . . 10
2.5 USB 3.0 – rozdiely a vylepšenia . . . 10
2.5.1 Linková vrstva . . . 11
2.5.2 Fyzická vrstva . . . 11
2.6 USB 3.1 . . . 12
2.6.1 Konektor typu C . . . 12

3 USB subsystém na μSonde **14**
3.1 Projekt μSondy . . . 14
3.2 Účel μSondy . . . 15
3.3 USB na μSonde . . . 15
3.4 Hardwarový popis μSondy . . . 16
3.4.1 Popis USB subsystému na μSonde – Cypress EZ-USB FX3 . . . 17
3.5 FX3 hardwarový popis . . . 18
3.5.1 Procesor . . . 18
3.5.2 DMA . . . 19
3.5.3 GPIF II . . . 20

4 Implementačná časť **21**
4.1 Činnosť μSondy a jej komponent . . . 21
4.2 FPGA komponenta . . . 22
4.3 FX3 firmware . . . 23
4.4 Aplikácia pre PC . . . 24
4.4.1 Knižnica CyUSB . . . 25
4.4.2 Návrh algoritmu . . . 25
4.4.3 Formáty INI3 a PCAP . . . 26
4.4.4 Kruhový buffer a synchronizácia vlákien . . . 28

4.4.5 Stavový automat 29

5 Testovanie **31**
5.1 Problémy 32

6 Možnosti rozšírenia práce **33**

7 Záver **34**

Seznam obrázků

2.1 USB topológia, znázornená vo forme pyramídy. [8] . . . 7
2.2 Koncové body, rúry a rozhrania. [15] . . . 8
2.3 USB pakety, prenos z hostiteľa do zariadenia. [10] . . . 9
2.4 Štruktúra USB 2.0 kábla. [8] . . . 9
2.5 USB 3.0: dual-bus architektúra. [12] . . . 11
2.6 Štruktúra USB 3.0 kábla. [12] . . . 12
2.7 USB typ C, konektor a kábel. [14] . . . 13

3.1 Architektúra systému pre zákonné odpočúvanie s použitím μSondy. [6] . . . 14
3.2 Ďalšia ukážka použitia μSondy. Použitie s TAPom je bezpečnejšie, pretože pri výpadku napájania alebo poruchy μSondy nebude odpočúvaná linka prerušená. [4] . . . 16
3.3 Predná strana μSondy [16] . . . 16
3.4 Schématické znázornenie μSondy. USB subsystém je vyznačený modrým rámikom. [16] . . . 17
3.5 USB detail subsystému μSondy: A – USB 3.0 kontrolér, B – konektor USB 3.0 typu B, C – reset jumper, D – reset tlačítko, E, F, G – pole PMOD, H – flash pamäť, I – I^2C rozhranie, J – UART, K – JTAG rozhranie. [16] . . . 17
3.6 FX3 – diagram logických blokov. [10] . . . 18
3.7 FX3 – hlavné prvky procesora. [10] . . . 19
3.8 Zbernicová sieť na mikrokontroléri FX3. [10] . . . 20

4.1 Schematické znázornenie častí implementovaných v mojej práci. [5] . . . 21
4.2 Komponenty implementované v FPGA μSondy. [2] . . . 22
4.3 Štrukrúra komponenty riadiaca prenosy z FPGA. . . . 22
4.4 Vývojový diagram stavového automatu komponenty. . . . 23
4.5 Vývojový diagram stavového automatu komponenty. . . . 24
4.6 Znázornenie činnosti vlákien v PC aplikácii. [5] . . . 26
4.7 Globálna hlavička, umiestnená na začiatku PCAP súboru. [1] . . . 27
4.8 Porovnanie hlavičiek paketov vo formáte INI3 a PCAP. [3][1] . . . 27
4.9 Pseudokód vlákien čitateľa a zapisovateľa. . . . 28

5.1 Testovanie FX3 mikrokontroléru na μSonde pomocou JTAG debuggera D-STREAM. . . . 31
5.2 Prepojka USB 3.0 typ A – micro B. . . . 32

Kapitola 1

Úvod

Fakulta informačních technologií Vysokého učení technického v Brne je riešiteľom projektu *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace*. Projekt má skrátený názov *SEC6NET*. Zameranie projektu je na vývoj prostriedkov pre monitorovanie a analýzu záznamu prevádzky na sieti. Venuje sa tiež výskumu metód zabezpečenia lokálnych sietí. Pri riešení projektu je kladený dôraz predovšetkým na siete s protokolom IPv6.

Jedným z cieľov projektu *SEC6NET* je vytvoriť špeciálne hardwarové zariadenia – sondy umožňujúce monitorovanie a selektívne odpočúvanie v IPv6 sieťach. Je nevyhnutné zabezpečiť, aby sondy zaznamenali všetku požadovanú komunikáciu bez straty jediného paketu. Pre akceleráciu filtrácie sieťovej prevádzky sú sondy sú postavené na platforme FPGA.

Jedným typom sondy vyvíjaným v rámci tohto projektu je mikrosonda, označovaná aj ako μSonda. μSonda je určená pre nasadenie k menšiemu ISP prípadne medzi koncového používateľa a ISP. Dokáže pracovať na 1 Gbps linke v oboch smeroch. Zozbierané dáta sonda posiela do zberného miesta zabezpečeným kanálom.

Ak nastane problém v spojení so zberným miestom, musí sonda ukladať zachytené dáta lokálne. Pre naplnenie tejto požiadavky je na sonde integrované rozhranie USB 3.0. Implementácia USB radiča v FPGA je náročná a zaberala by príliš veľa zdrojov. Preto bol na platformu pridaný mikrokontrolér Cypress EZ-USB FX3, ktorý zabezpečuje USB 3.0 konektivitu.

V mojej práci spájam 2 technológie: FPGA a USB. Predmetom mojej práce je zabezpečiť prenos dát z FPGA čipu do PC prostredníctvom mikrokontroléra Cypress EZ-USB FX3. Cieľom mojej práce bolo implementovať FPGA komponentu riadiacu prenos dát do mikrokontroléra FX3 pomocou GPIF II rozhrania. Následne bolo potrebné upraviť firmware mikrokontroléra FX3, aby bola dosiahnutá čo najvyššia priepustnosť dát. Poslednou časťou implementovanou v rámci mojej práce je PC aplikácia pre operačný systém Linux, ktorá prijíma dáta z mikrokontroléra FX3 cez USB 3.0 rozhranie. Aplikácia ukladá zachytené dáta na disk vo formáte PCAP, ktorý je možné ďalej spracovať sieťovým analyzačným programom, ako je napríklad Whireshark.

V druhej kapitole mojej práce sa venujem popisu USB rozhrania, od jeho vzniku až po súčasný stav. V tretej kapitole bližšie predstavujem projekt SEC6NET a čo je očakávané od mikrosondy. Zároveň tu popisujem hardwarovú platformu mikrosondy a jej USB rozhranie, ktoré predstavuje mikrokontrolér FX3. Vo štvrtej kapitole som sa zameral na implementačnú časť mojej práce. Popisujem v nej implementáciu riešenia jednotlivých súčastí: komponentu riadiacu prenosy z FPGA, firmware pre mikrokontrolér FX3 a klientskú aplikáciu pre PC. V kapitole päť sa zaoberám testovaním implementovaného riešenia a zhŕňam

dosiahnuté výsledky. Možné rozšírenia práce rozoberám v šiestej kapitole. V závere práce hodnotím jej celkový prínos.

Kapitola 2

USB

S rozšírením a vývojom osobných počítačov od osemdesiatych rokov vzrastal ich výkon, narastal počet výpočtových úloh, ktoré na nich bežia a súčasne vzrastal aj počet periférnych zariadení, ktoré chceme k počítaču pripojiť. Počet voľných portov na pripojenie zariadení bol obmedzený a rástla potreba zavedenia jednoduchého rozhrania, ktoré by bolo nenáročné ako aj z hľadiska používateľa, tak z hľadiska vývojára a umožnilo by jednoduchú realizáciu pripojenia nového zariadenia.

Univerzálna sériová zbernica (USB) bola od začiatku konštruovaná ako rozhranie pre komunikáciu s mnohými typmi periférií bez obmedzení predchádzajúcich starších rozhraní, teda bez potreby rozoberať PC kvôli pridaniu nového zariadenia. Dôležité vlastnosti USB rozhrania sú nasledovné [17]:

- Jednoduché pripojenie nového zariadenia – plug-and-play, konektory sú prístupné z vonkajšku počítača. Zariadenie môže byť pripojené za behu PC a po inštalácií nie je nutné znova zavádzať operačný systém. Po pripojení je zariadenie automaticky nakonfigurované.
- Pripojenie rôznych typov zariadení cez rovnaký typ kábla.
- Zariadenia sú napájané priamo z kábla – 5 V; maximálny odber zariadenia je 500 mA pri USB 2.0 a 900 mA pri USB 3.0.
- Možnosť pripojiť 127 zariadení: adresa zariadenia má dĺžku 7 bitov, nultá hodnota je pridelená novopripojenému zariadeniu.
- Podpora zariadení pracujúcich v reálnom čase (multimédiá, spracovanie zvuku).
- Pripojenie cez USB je lacné – nízka cena konektora, nenáročná implementácia.
- Možnosť združovať viacero logických zariadení do jedného fyzického – na zariadenie, ktoré vie komunikovať cez USB je možné pripojiť niekoľko periférnych zariadení.
- Nízka záťaž zbernice samotným komunikačným protokolom – nízka réžia.
- Moňosť prenášať dáta v rýchlostiach rádu niekoľkých kb/s až po jednotky Gb/s.

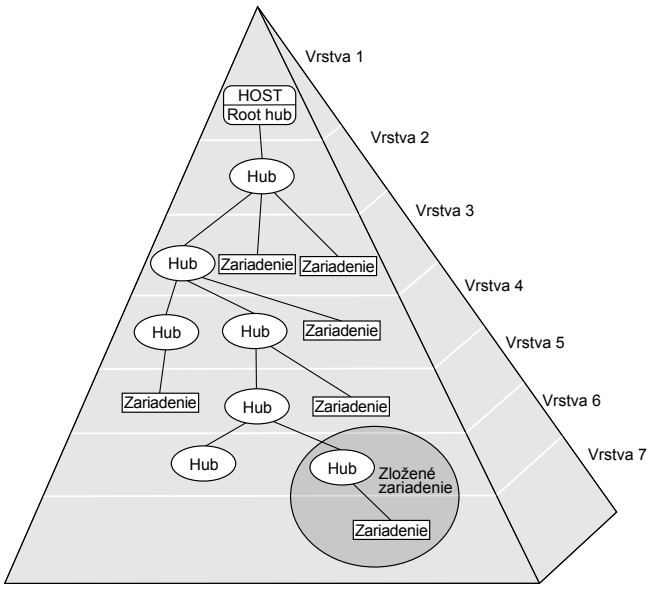
2.1 Verzie a rýchlosti

USB štandard sa vyvíjal niekoľko rokov pred tým, ako bol v roku 1996 oficiálne uvedený vo verzii 1.0. Táto verzia definuje dve rýchlosti prenosov: 1,5 Mb/s – Low Speed a 12 Mb/s – Full speed. Široko rozšírená bola však až revízia 1.1 uvedená v 1998. [9]

V roku 2000 bola uvedená verzia 2.0, ktorá definuje prenosovú rýchlosť 480 Mb/s – High speed. [8] Verzia 3.0 z roku 2008 desaťnásobne zvyšuje prenosovú rýchlosť na 5 Gb/s – SuperSpeed. Revízia 3.1 uvedená v júli 2013 túto rýchlosť zvyšuje dvojnásobne na 10 Gb/s – SuperSpeedPlus. Verzie 2.0, 3.0 a 3.1 sú spätne kompatibilné s verziou 1.1. [13]

2.2 Topológia

Anglická literatúra nazýva všetky prístroje na USB zbernici *zariadeniami – USB devices* a rozlišuje medzi *rozbočovačom – USB hub* a *zariadením*, ktoré pridáva hostiteľskému systému nejaké schopnosti – *USB function*. Doslovný preklad „USB funkcia“ je trocha neobratný, preto ho budem nazývať koncové zariadenie. Zariadenie, ktoré implementuje viacero rôznych funkcií, je nazývané *composite device* (napríklad multifunkčná tlačiareň). *Compound device* je zasa spojenie rozbočovača a jednej alebo viacerých funkcií (zložené zariadenie – klávecnica s rozširujúcimi USB portami).



Obrázek 2.1: USB topológia, znázornená vo forme pyramídy. [8]

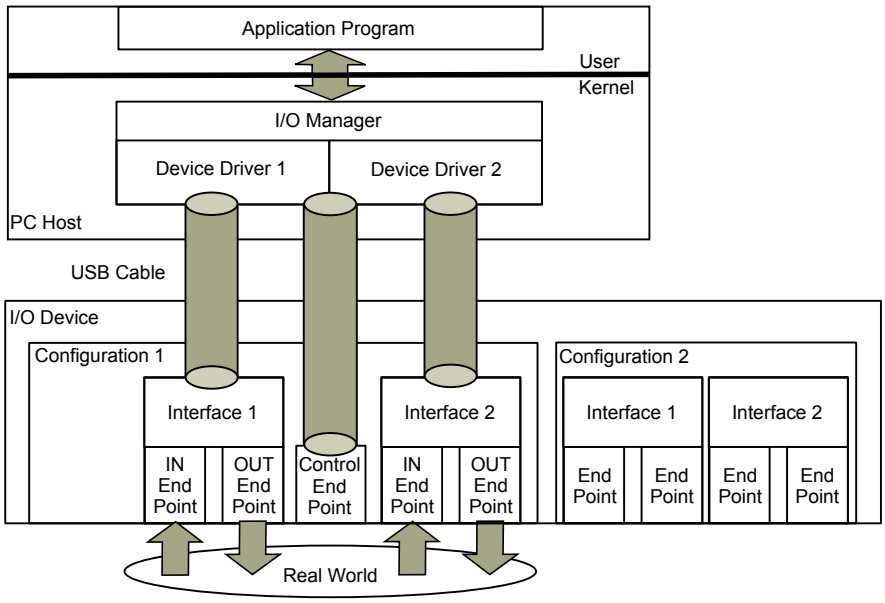
USB 2.0 podľa špecifikácie využíva vrstvovú hviezdicovú topológiu, kde v centre každej hviezdice je USB hub. K tomuto hubu môže byť pripojené na ďalšej úrovni buď koncové zariadenie alebo ďalší hub. Zbernica USB obsahuje jeden koreňový *rozbočovač – root hub*, ktorý je považovaný za najvyššiu úroveň a sú k nemu pripojené ďalšie zariadenia a huby. Rozhranie medzi USB systémom a hostiteľským počítačom sa nazýva *hostiteľský radič – Host Controller*. Koreňový rozbočovač je integrovaný spolu s hostiteľským radičom do hostiteľského systému. S ohľadom na oneskorenie signálu v kábloch a huboch povoluje špecifikácia najviac sedem úrovní vrátane koreňovej vrstvy. To znamená, že medzi koreňovým rozbočovačom a koncovým zariadením môže byť zapojených maximálne päť rozbočovačov.

2.3 Protokol

Všetka komunikácia sa odohráva cez koreňový rozbočovač, priama komunikácia medzi zariadeniami nie je možná.

2.3.1 Rúry, endpointy

Prenosy dát môžu nastať medzi hostiteľským softwarom a logickou entitou nazývanou *koncový bod – endpoint* cez logický kanál nazývaný *rúra – pipe*. Koncové zariadenie môže mať 32 aktívnych rúr, 16 pre prenosy k hostu a 16 od hosta. *Rozhranie – interface* je kolekcia spolupracujúcich koncových bodov implementujúcich špecifickú funkciu.



Obrázek 2.2: Koncové body, rúry a rozhrania. [15]

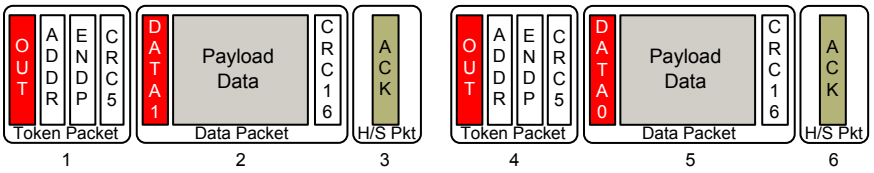
2.3.2 Protokolová vrstva

Všetky dátové prenosy na zbernici USB prebiehajú za pomoci paketov a sú iniciované host controllerom. Väčšina zbernicových transakcií pozostáva z vyslania troch paketov. Každá transakcia začína tým, že host controller vyšle *token paket*, popisujúci typ, smer prenosu, adresu zariadenia a číslo koncového bodu v zariadení. Zariadenie, ktoré rozpozná svoju adresu sa pripraví na prenos. Zdroj dát, zariadenie alebo systém, potom vyšle dátový paket alebo oznámi, že nemá žiadne dáta k odoslaniu. Transakcia býva ukončená tým, že príjemca vyšle *handshake paket*, ktorým potvrdí úspešnosť prenosu. Obrázok 2.3 tento prenos ilustruje.

2.3.3 Typy prenosov – dátových tokov, paketov

USB špecifikácia definuje štyri typy prenosov. Tieto sa zhodujú s potrebami rôznych typov dát, ktoré majú byť po zbernici doručené.

- *Bulk – hromadné prenosy:* slúžia k prenosom veľkého množstva dát (tlačiareň, modem, pevný disk) a sú na ne kladené najmenšie obmedzenia. Dáta sú prenášané sekvenčne,



Obrázek 2.3: USB pakety, prenos z hostiteľa do zariadenia. [10]

majú spoľahlivosť zaručenú detekciou chýb na hardwarovej úrovni a obmedzeným počtom opakovaných pokusov. Šírka pásma, využitá hromadným prenosom môže byť rôzna a záleží na ostatnej prevádzke na zbernici.

- *Interrupt – prerušovacie prenosy:* slúžia k včasnému a spoľahlivému doručeniu dát, najčastejšie po asynchrónnej udalosti (stlačenie klávesy, pohyb myši). Jedná sa o malé množstvo dát, ktoré sú doručené USB zbernicou v najkratšom možnom čase. Koncové body tohoto typu majú pridelený určitý interval dotazovania a sú hostom dotazované v pravidelných intervaloch.

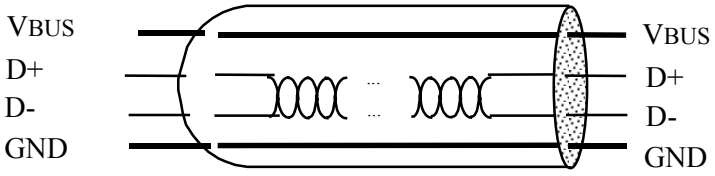
- *Isochronous – izochrónne prenosy:* „stejnodobé“, plynulé. Sú to trvalé prenosy najčastejšie streamov audia a videa, ktoré sú spracovávané v reálnom čase. Majú dopredu dohodnuté množstvo prenosového pásma a povolené oneskorenie. Prenášané dáta sú citlivé na oneskorenie a požiadavky naň sú dané vyrovnávacou pamäťou koncového bodu. Šírka prenosového pásma je často odvodená od vzorkovacej frekvencie daného zariadenia. Nevyužíva sa handshake ani opakovaný pokus o odoslanie – chyby pri prenose sú tolerované a nie je možné ich opraviť iba detekovať.

- *Control – riadiace prenosy:* majú za úlohu nakonfigurovať zariadenie pri jeho prvom pripojení. Ďalej zariadeniam posielajú príkazy, môžu byť použité k riadeniu ďalších komunikačných rúr. Pretože sú také dôležité, majú najrozsiahlejšiu kontrolu chýb. Host má vyhradenú časť každého USB rámca pre riadiace prenosy.

2.3.4 Fyzická a linková vrstva

Linková vrstva má na starosti úlohy zvyšujúce spoľahlivosť prenosu, to zahŕňa usporiadanie bajtov, rámcovanie na linkovej vrstve.

Viac je známa ako elektrické rozhranie. Tieto vrstvy sú tvorené obvodmi na serializáciu a deserializáciu dát, vyvažovacie obvody a obvody na vedenie a detekciu signálov vo vodičoch. Veškerá obsluha chýb nastáva na protokolovej vrstve.



Obrázek 2.4: Štruktúra USB 2.0 kábla. [8]

Kábel je tvorený napäťovým a zemniacim vodičom, cez ktorý môžu byť koncové zariadenia napájané. Ďalej je tvorený krúteným párom dátových vodičov, ktoré vysielajú diferenciálny signál. To znamená, že ak je na D+ vodiči napäťová úroveň logickej 1, na D–

vodiči bude logická 0 a naopak. Použitie krútenej dvojlinky je výhodné kvôli odolnosti voči šumu a súhlasnému rušeniu, kedy sa v oboch vodičoch indukuje rovnaké napätie.

Dáta sú po vodičoch prenášané kódovaním *NRZI* – nulový bit neguje predchádzajúcu hodnotu signálu. Vzhľadom k tomu, že zbernica má vnútornú synchronizáciu na základe prenášaných dát, pri dlhej postupnosti logických 1 by mohlo dôjsť k rozsynchronizovaniu. Preto sa po sekvencii šiestich logických jednotiek pridá do signálu logická 0. [18]

2.3.5 Detekcia novo pripojeného zariadenia – „*Enumeration*“

Hostiteľ alebo hub monitoruje napätie dátových vodičov na každom svojom porte. Zariadenie je detekované, keď je zaznamenané zvýšenie napätia na jednom z dátových vodičov. Zvýšenie napätia na D+ vodiči znamená, že zariadenie je schopné full-speed komunikácie, zatiaľ čo nárast napätia na D– vodiči indikuje low-speed zariadenie.

Aby klientská aplikácia (OS) mohla komunikovať so zariadením, musí sa hostiteľ dozvedieť informácie o zariadení a priradiť mu patričný ovládač. Proces výmeny týchto informácií sa nazýva *Enumeration* a pozostáva z nasledujúcich krokov:

1. Hostiteľ pošle *Get Descriptor-Device* požiadavok na adresu nula – všetky USB zariadenia musia reagovať na adresu nula, keď sú novo pripojené.
2. Zariadenie odpovedá poslaním svojich identifikačných údajov hostiteľovi.
3. Hostiteľ pošle *Set Address* požiadavok, ktorý priradí novo pripojenému zariadeniu jedinečnú adresu na USB zbernici aby mohlo byť rozlíšené od ostatných.
4. Hostiteľ pošle zariadeniu ďalšie *Get Descriptor* požiadavky, zisťujúce doplňujúce informácie. Z nich sa dozvedá všetko podstatné, napríklad počet koncových bodov, napájacie požiadavky, požadovanú šírku pásma a ktorý ovládač je potrebné načítať.

High-speed zariadenie je najprv detekované ako full-speed. Do high-speed režimu sa prepne až po dohode s hostiteľom.

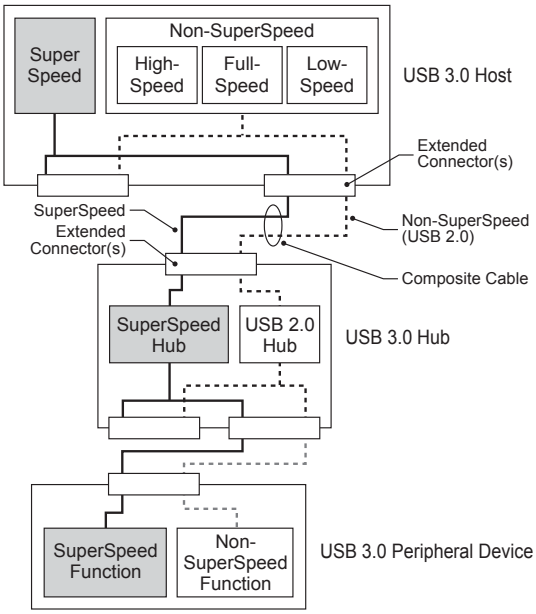
2.4 USB On-The-Go (OTG)

Ako sa USB rozhranie presadzovalo u všetkých typoch periférií, vznikla požiadavka priameho prepojenia týchto zariadení (napríklad spojenie digitálneho fotoaparátu s tlačiarňou). Zariadenie s podporou OTG má dve role: môže vystupovať ako koncové zariadenie a aj ako hostiteľ s obmedzenou funkčnosťou. To umožňuje zariadeniam priamu komunikáciu, no v skutočnosti je zachovaná architektúra hostiteľ – zariadenia. [7]

Táto technológia nachádza využitie v súčasných smartfónoch a tabletoch. Umožňuje rozšírenie ich funkčnosti pripojením rôznych typov zariadení, od klávesnice a myši až po externý pevný disk. Pripojením hubu môžu byť tieto zariadenia aktívne súčasne. Táto technológia má tiež svoje obmedzenia. Zariadenia podporujúce OTG bývajú často napájané z batérie a nie sú schopné dodať dostatok prúdu pre napájanie niektorých periférií. Riešením je napájať preriférne zariadenie externe alebo použiť USB hub s externým napájaním.

2.5 USB 3.0 – rozdiely a vylepšenia

Rozhranie USB 3.0 sa skladá z fyzickej SuperSpeed zbernice doplnenej o fyzickú USB 2.0 zbernicu. Jedná sa o takzvanú *dual-bus* architektúru. Pripojené zariadenia pracujú na najvyššej možnej rýchlosti a nie je im dovolené využívať obe zbernice súčasne.



Obrázek 2.5: USB 3.0: dual-bus architektúra. [12]

Jednou z dôležitých úprav v USB 3.0 špecifikácii je *unicasting* namiesto *broadcastingu*. Pakety v USB 2.0 sú vysielané všetkým zariadeniam na zbernici. To znamená, že každé zariadenie musí z paketu dekódovať adresu a overiť, či je paket adresovaný jemu, čo vedie k vyššej spotrebe. USB 3.0 pakety sú smerované priamo koncovému zariadeniu. Smerovacia informácia pre huby je súčasťou paketu.

Ďalšia zmena sa týka prerušovacích prenosov. V USB 2.0 sú tieto prenosy iniciované hostiteľom každý prerušovací interval, či už je zariadenie pripravené na prenos alebo nie. Pokiaľ SuperSpeed koncové body signalizujú hostiteľovi, že nie sú pripravené, hostiteľ sa na ne prestane dotazovať. Dotazovanie opäť začne, až keď zariadenie samo asynchrónne vyšle signál hostiteľovi.

Dual simplex architektúra umožňuje USB 3.0 protokolu posielať viacero paketov jedným smerom bez čakania na podvrdzovací (ACK) paket od prijímajúcej strany. Toto by pri half-duplex architektúre, akou je aj USB 2.0, spôsobilo „nezhodu na zbernici“, teda obe zariadenia by sa súčasne pokúšali zapisovať na zbernicu. Jednotlivé dátové pakety sú očíslované, takže potvrdzovacie pakety potom určujú, ktoré pakety sa stratili alebo zle preniesli a je nutné ich preniesť znovu. Počet dávkových paketov je dopredu dohodnutý.

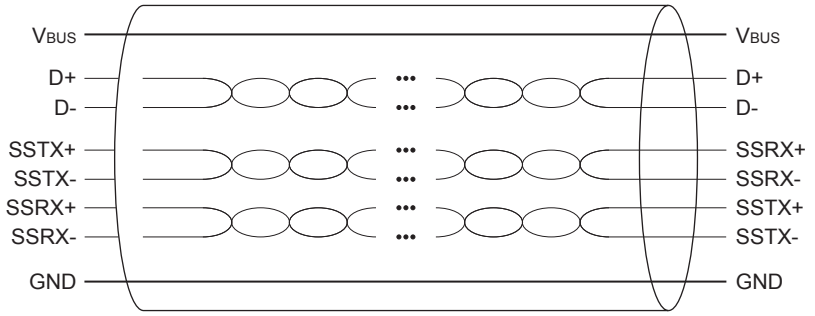
2.5.1 Linková vrstva

Linková vrstva udržuje spojenie a zabezpečuje integritu dát medzi komunikujúcimi zariadeniami. Zaručuje spolahlivé doručenie dát zabalením do paketu na vysielacom konci a detekciou chýb linkovej vrstvy na strane príjemcu. Zabezpečuje taktiež riadenie toku dát.

2.5.2 Fyzická vrstva

Kábel je tvorený dvomi diferenciálnymi pármi vodičov, jeden pre odosielanie a druhý pre príjem dát. Okrem týchto, je káblom vedený aj tretí pár vodičov verzie 2.0 zachovávajúci spätnú kompatibilitu. Pre prenos dát sa používa kódovanie 8b/10b, kde je snaha dosiahnuť

rovnaký počet prenášaných logických núl a jednotiek. Týmto kódovaním je zabezpečená vnútorná synchronizácia a zamedzuje sa pamäťovému efektu vodičov.



Obrázek 2.6: Štruktúra USB 3.0 kábla. [12]

2.6 USB 3.1

Hlavné vylepšenie oproti verzii 3.0 spočíva v zdvojnásobení prenosovej rýchlosti z 5 na 10 GHz. Ďalšou zmenou je efektívnejší prenos dát. Oproti USB 3.0, ktoré používa kódovanie 8b/10b sa prešlo na efektívnejšiu kódovaciu schému 128b/132b. To znamená zníženie prenosu kontrolných bitov z 20 % na 3 %.

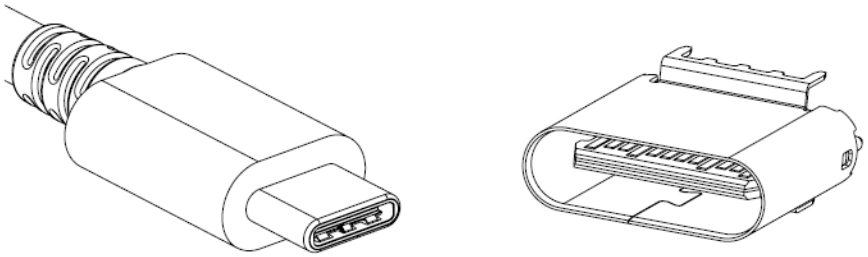
Vyššia prenosová rýchlosť však vyžaduje pri prenose väčšiu spoľahlivosť jak od zariadení, tak aj od prenosového média. Kábel verzie 3.0 má diferenciálny pár dátových SuperSpeed vodičov tienený dohromady. Pri USB 3.1 je každý vodič tienený zvlášť. Dĺžka pasívneho kábla je obmedzená na 1 meter, dlhší kábel vyžaduje opakovač. [20]

2.6.1 Konektor typu C

Ako doplnok k novému štandardu 3.1 bol vyvinutý nový typ konektora, ktorý predstavuje menšiu, tenšiu a odolnejšiu alternatívu k existujúcim typom konektorov a súčasne odstraňuje ich základné neduhy. Staršie typy konektorov zabraňujú inovácií nových produktov, kvôli ich veľkosti a obmedzeniam kladených pri implementácii.

Konektor typu C je symetrický, takže pri pripájaní nemusíme riešiť jeho správnu orientáciu. Konektor na strane hosta je rovnaký ako konektor na strane zariadenia, teda je jedno ktorým smerom je kábel zapojený. Rozmery konektora sú podobné rozmerom Micro-B konektora: 8,4 mm × 2,6 mm. Toto riešenie je cielené hlavne na veľmi tenké platformy, od tenkých notebookov až po smartfóny. Predpokladá sa však široké nasadenie konektora vo všetkej domácej a priemyselnej elektronike, čím sa toto riešenie stáva skutočne univerzálnym. Pre zachovanie kompatibility so staršími konektormi bude treba použiť pasívne redukcie. Nový konektor bol navrhnutý tak, aby bolo možné ďalšie rozširovanie USB špecifikácie. [14]

USB sa z rozhrania pre prenos dát s obmedzenými možnosťami napájania stáva štandardom pre nabíjanie prenosných zariadení. Väčšina týchto zariadení pre chod vyžaduje vyšší prúd ako povoľuje špecifikácia. Preto je v konektore typu C implementovaný štandard *USB Power Delivery 2.0* ktorý rozširuje možnosti napájania zariadení. Tento štandard umožňuje dodať zariadeniu 100 W (20 V, 5 A) elektrického výkonu cez kábel typu C. Je to dostatočné množstvo energie pre napájanie súčasného notebooku. Množstvo dodávanej energie je regulovateľné podľa aktuálnych potrieb zariadenia. Navyše, smer dodávania energie nie je



Obrázek 2.7: USB typ C, konektor a kábel. [14]

fixný. Zariadenia si samé určia, ktoré dodáva a ktoré prijíma energiu. Host teda nemusí byť poskytovateľom energie, ale môže ju prijímať od niektorého z periférnych zariadení. [11]

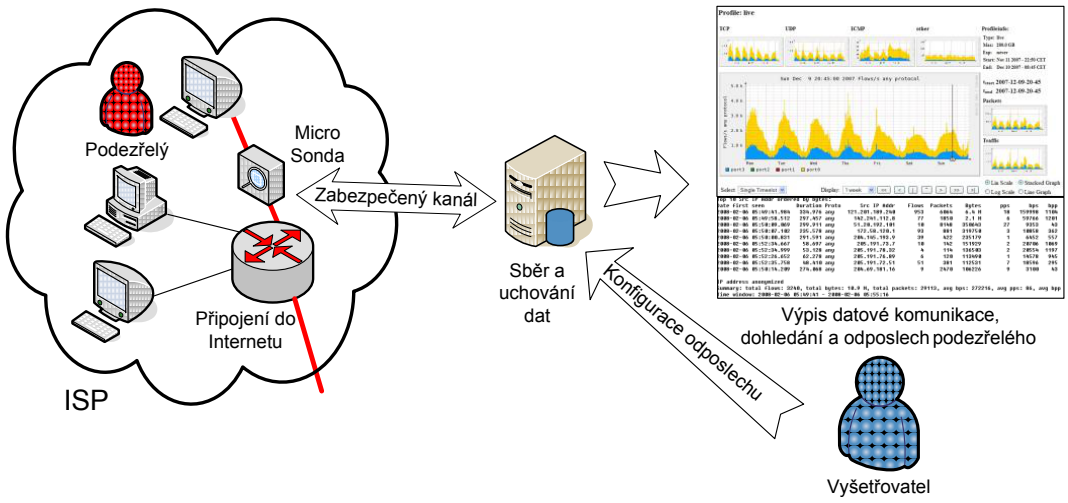
Kapitola 3

USB subsystém na μSonde

3.1 Projekt μSondy

Fakulta informačních technologií Vysokého učení technického v Brne je riešiteľom projektu *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace*. Skrátený názov projektu je SEC6NET. Projekt je zameraný na výskum a vývoj prostriedkov monitorovania prevádzky na sieti, analýzu záznamu sieťovej prevádzky a metód zabezpečenia lokálnych sietí. Dôraz je kladený hlavne na siete novej generácie s protokolom IPv6. Potreba nájsť nové prístupy k zaisteniu kybernetickej bezpečnosti je daná zmenou internetového protokolu na protokol IPv6. Pri nasadení nového protokolu nie je možné aplikovať technické postupy platné pre IPv4. Zároveň sa v prostredí IPv6 vyskytujú nové možnosti prenosov dát a správy sietí, ktoré majú vplyv na bezpečnosť.

Prieskum produktov v oblasti zachytávania a spracovania prenášaných dát v IPv6 sieťach ukazuje, že v súčasnej dobe neexistuje riešenie, ktoré by uspokojovalo požiadavky projektu. Komerčné riešenia sú veľmi drahé a nedá sa tvrdiť, že s cenou rastie aj kvalita a výkon zariadení. Ďalšia nevýhoda je v uzavretosti systému, nízkom výkone pri vysokej priepustnosti a problému s ukladaním množstva zachytených dát. Open source riešenia majú dostatočný výkon, ale sú silne závislé od ich tvorcu a majú mnoho neriešených chýb. [6]



Obrázek 3.1: Architektúra systému pre zákonné odpočúvanie s použitím μSondy. [6]

Jedným z cieľov tohto projektu je preto vytvoriť špeciálne hardwarové zariadenia – sondy, ktoré umožnia monitorovanie a selektívne odpočúvanie vo vysokorýchlostných IPv6 sieťach. Pre možnosť použitia sondy orgánom činným v trestnom konaní musia byť zachytené informácie nespochybniteľné. Je preto nevyhnutné zaznamenať všetkú požadovanú komunikáciu bez straty jediného paketu. Aby mohli sondy pracovať na plnej priepustnosti linky, bola pri ich návrhu použitá technológia FPGA pre akceleráciu filtrácie sieťovej prevádzky. [19]

Vyvíjané sondy sú 2 typov: *vysokorýchlostná sonda* a *mikrosonda*. *Vysokorýchlostná sonda* je určená do sietí veľkých ISP a na chrbticové linky s priepustnosťou v rádoch desiatok Gbps. *Microsonda* označovaná ako μSonda dokáže pracovať na rýchlosti 1 Gbps a je určená pre nasadenie k menším ISP. Môže byť použitá aj priamo v infraštruktúre medzi ISP a koncovým používateľom kedy je umožnené úplne tajné odpočúvanie. [6]

3.2 Účel μSondy

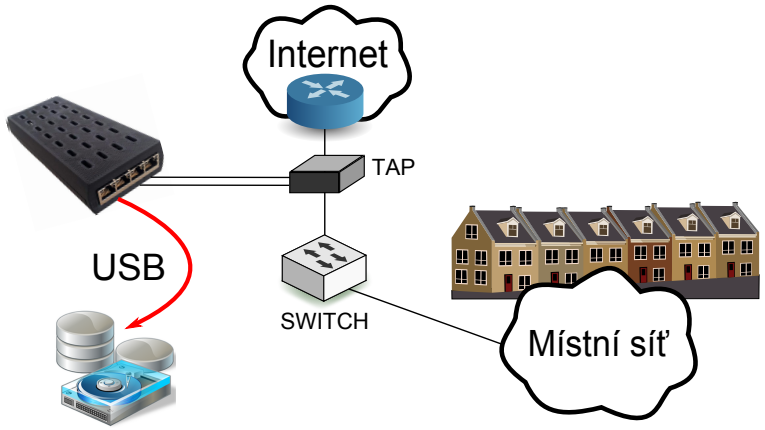
Pre naplnenie cieľov projektu boli stanovené požiadavky pre reálne nasadenie μSondy. Kľúčovou vlastnosťou je spracovanie monitorovanej sieťovej aktivity na plnej priepustnosti 1 Gbps linky v oboch smeroch. Jedná sa hlavne o zaznamenanie všetkých prenesených paketov s možnosťou utajenia sondy v infraštruktúre siete. Zachytené dáta sú posielané na zberné miesto, najlepšie zabezpečeným a spoľahlivým kanálom. Pri výpadku spojenia so zberným miestom musia byť dáta ukladané lokálne na externé pamäťové médium. Sonda by mala mať rozhranie umožňujúce jednoduchú konfiguráciu. Pre potrebu nasadenia v špatne prístupných podmienkach musí mať sonda nízku spotrebu aby mohla byť napájaná z batérií.

Na základe týchto požiadaviek bolo testovaných niekoľko existujúcich platforiem. Testované platformy však nespĺňali základnú požiadavku. Nedokázali zachytiť všetky pakety pri plnej priepustnosti linky. Pristúpilo sa preto k návrhu vlastnej hardwarovej platformy. [19]

3.3 USB na μSonde

Aby platforma mala dostatočný výkon pre spracovanie a filtrovanie sieťovej aktivity na plnej rýchlosti linky, bol ako hlavný výpočtový prvok zvolený čip FPGA doplnený procesorom MicroBlaze. Táto kombinácia umožňuje beh softvérových aplikácií doplnenými o akceleráciu časovo kritických úloh. Zároveň je možná jednoduchá úprava alebo pridanie novej funkčnosti. Pre splnenie požiadavky lokálneho uloženia dát na externé pamäťové médium bolo na sondu pridané rozhranie USB 3.0. Cez toto rozhranie je možné pripojiť externý pevný disk. Implementovanie USB radiča by zabralo veľké množstvo zdrojov v FPGA procesore. Preto bol do platformy pridaný mikrokontrolér *Cypress EZ-USB FX3*, ktorý má dostatočne rýchle a jednoduché rozhranie na prenos dát z FPGA čipu. Je ním 32 bitové *GPIF II* rozhranie pracujúce na frekvencii 100 MHz.

Cieľom mojej práce je preniesť zachytené dáta z FPGA procesora do PC pomocou USB rozhrania cez mikrokontrolér FX3. Práca je rozdelená do troch častí. V prvej časti bolo potrebné upraviť VHDL komponentu komunikujúcu s mikrokontrolérom FX3 cez GPIF II rozhranie. Ďalšou časťou bola úprava firmware mikrokontroléra aby bola dosiahnutá čo najvyššia priepustnosť. V poslednej časti som sa venoval vývoju klientskej aplikácie pre PC, ktorá prijíma zachytené sieťové pakety a ukladá ich na disk. Zachytené pakety sú uložené vo formáte PCAP, ktorý je možné ďalej spracovať programom na analýzu sieťových dát.

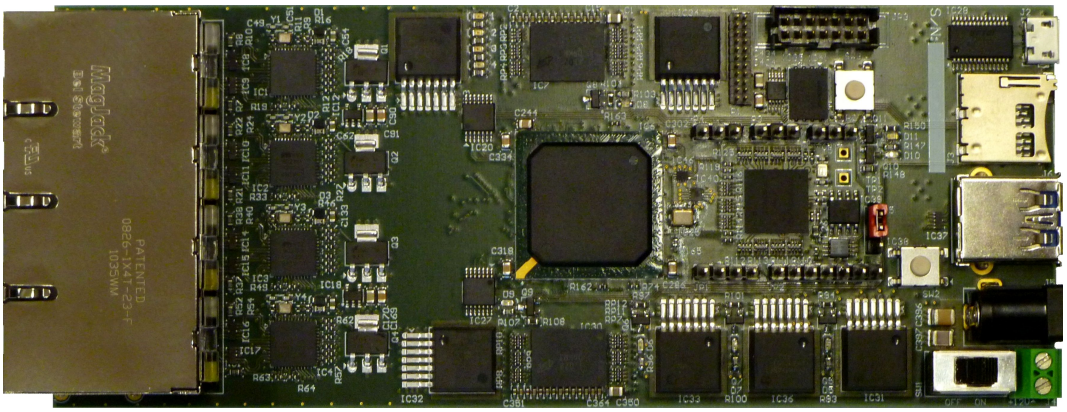


Obrázek 3.2: Ďalšia ukážka použitia μSondy. Použitie s TAPom je bezpečnejšie, pretože pri výpadku napájania alebo poruchy μSondy nebude odpočúvaná linka prerušená. [4]

Schéma implementovaných častí je zobrazená na obrázku 4.1 na strane 21. V mojej práci sa ďalej zameriam na popis USB subsystému μSondy a USB 3.0 čipu Cypress EZ-USB FX3.

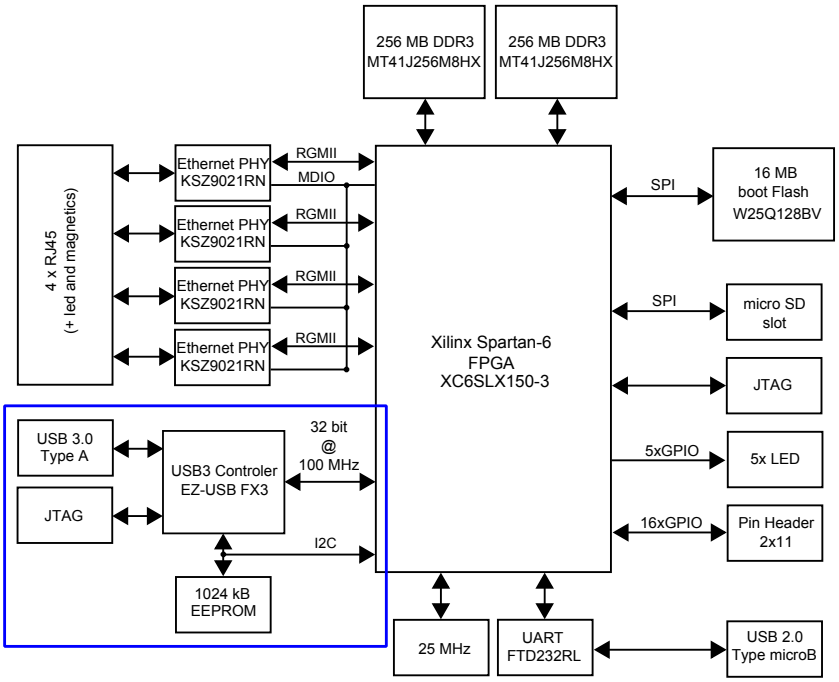
3.4 Hardwarový popis μSondy

μSonda, označovaná aj uG4-150, je vstavaná vývojová platforma primárne určená na spracovanie sieťových paketov do rýchlosti jedného gigabitu. Jej srdcom je FPGA čip Xilinx Spartan-6. Môže byť tiež využitá ako experimentálna platforma.



Obrázek 3.3: Predná strana μSondy [16]

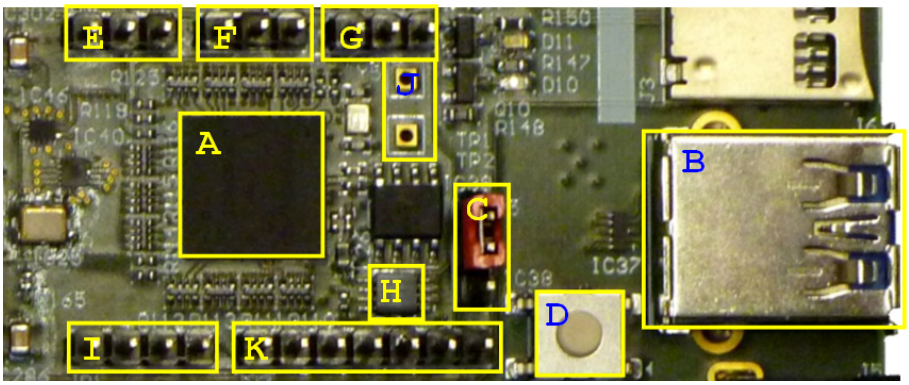
Platforma obsahuje štyri LAN porty pre sieťovú konektivitu, DDR pamäte použiteľné ako rýchle lokálne uložisko dát alebo ako systémová RAM pre softprocesor, ďalej 16 MB bootovaciu flash pamäť pre FPGA, ktorá môže byť použitá aj ako pomalé trvalé úložisko dát, ďalej obsahuje rozhranie USB 3.0 pre rýchle prenosy nazbieraných dát do počítača alebo úložného zariadenia (USB disk), sériové rozhranie (UART), štandardný JTAG konektor používaný pri ladení. Nachádzajú sa tu tiež signalizačné LED diódy a GPIO rozhranie pre pripojenie komponent, ktoré nie sú integrované na platforme. Všetky tieto komponenty sú pripojené k FPGA čipu. [16]



Obrázek 3.4: Schématické znázornenie μSondy. USB subsystém je vyznačený modrým rámikom. [16]

3.4.1 Popis USB subsystému na μSonde – Cypress EZ-USB FX3

USB 3.0 rozhranie na μSonde je obsluhované kontrolérom *Cypress EZ-USB FX3*, ktorý je pripojený k FPGA čipu cez 32 bitové programovateľné rozhranie *Cypress GPIF II* (kapitola 3.5.3). Toto rozhranie môže bežať na frekvencii 100 MHz a poskytuje prenosovú rýchlosť 400 MBps z FPGA do USB radiča a naopak.

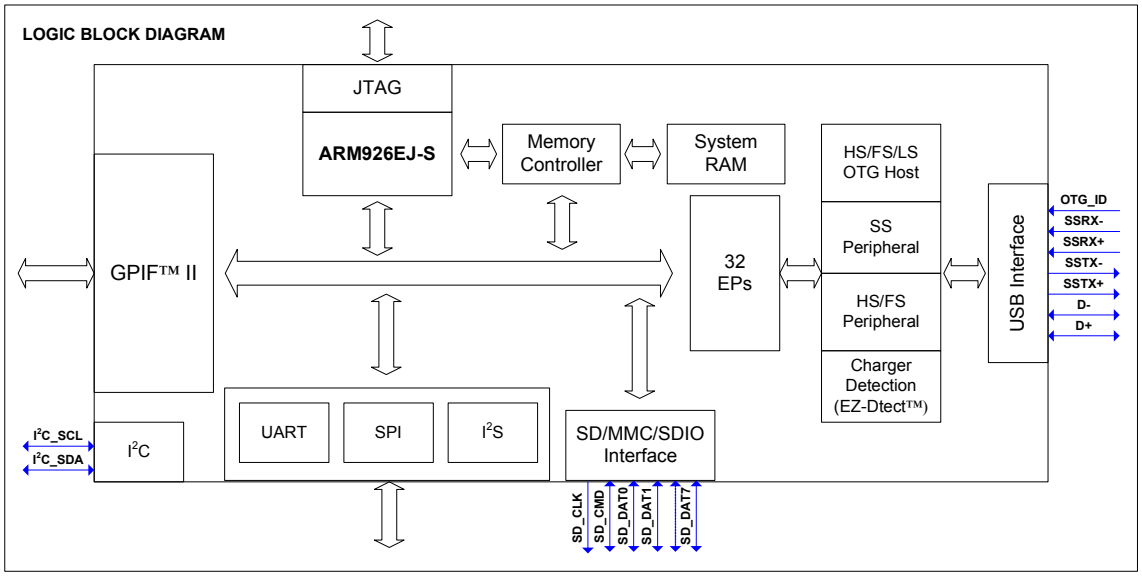


Obrázek 3.5: USB detail subsystému μSondy: A – USB 3.0 kontrolér, B – konektor USB 3.0 typu B, C – reset jumper, D – reset tlačítko, E, F, G – pole PMOD, H – flash pamäť, I – I^2C rozhranie, J – UART, K – JTAG rozhranie. [16]

Reset kontroléra môže byť vyvolaný z FPGA alebo tlačítkom, podľa nastavenia reset jumpra. S kontrolérom je spojená flash pamäť od Microchipu cez I^2C rozhranie, ktorá má kapacitu 1024 kb a určená je pre nabootovanie obslužného firmwaru. Pamäť je tiež

pripojená I^{2}C rozhraním ku FPGA a k externému I^{2}C rozhraniu, za účelom programovania. Pole vývodov PMOD slúži na výber bootovacieho zdroja. Pre účely ladenia je k dispozícií sériové rozhranie UART (signály RX a TX) a plné JTAG rozhranie. [16]

3.5 FX3 hardwarový popis



Obrázek 3.6: FX3 – diagram logických blokov. [10]

FX3 je všeobecne použiteľný integrovaný USB 3.0 SuperSpeed kontrolér so zabudovaným programovateľným rozhraním (GPIF II), ktorý umožňuje vývojárom pridať USB 3.0 k ľubovoľnému systému.

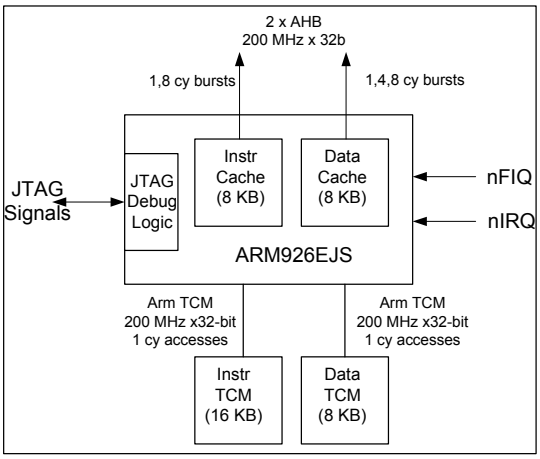
Kontrolér s integrovaným USB 3.0 radičom a 32 bitovým procesorom je použiteľný pri vývoji špecifických aplikácií. Pre aplikácie vyžadujúce rolu hostiteľa je integrovaný USB 2.0 OTG radič. Plne programovateľné, paralelné rozhranie – *General Programmable Interface (GPIF II)* poskytuje pripojenie k akémukoľvek procesoru, ASIC, DSP alebo FPGA čipu. Kontrolér obsahuje 512 kB integrovanej pamäte SRAM pre kód a dáta. Ďalej sa tu nachádzajú pomalšie rozhrania ako UART, SPI, I^{2}C, a I2S pre komunikáciu s inými vstavanými periférami, napríklad pamäťou EEPROM. Procesor riadi dátové prenosy medzi USB zbernicou, GPIF II, I2S, SPI a UART rozhraniami pomocou firmwaru a DMA mechanizmu.

3.5.1 Procesor

FX3 je poháňaný 32 bitovým ARM procesorom, konkrétne *ARM926EJS*. Tento procesor pracuje na frekvencii 200 MHz a výkonnosť má 220 MIPS, čo je postačujúci výkon na kódovanie hudby do MP3 formátu.

Použitá je Harvardská architektúra, teda procesor má oddelené dátové cesty pre kód a dáta cez dedikované 32 bitové štandardné AHB zbernice. Oddelená inštrukčná a dátová cache pamäť sú vstavané v jadre, aby zabezpečili prístup s nízkou odozvou ku často používanému kódu a dátam. Obe cache pamäte majú 8 KB. Sú to 4-cestné asociatívne pamäte.

Ďalej sa na procesore nachádzajú dve pamäte úzko spojené s jadrom (TCM) pre kód a dáta, ktoré majú garantovanú nízku odozvu. Procesor obsahuje tiež jednotku správy pamäte (MMU), ktorá je schopná prekladať virtuálne adresy na fyzické.



Obrázek 3.7: FX3 – hlavné prvky procesora. [10]

3.5.2 DMA

DMA sú označované prenosy dát prebiehajúce bez zásahu CPU. Prenosy môžu prebiehať medzi perifériou a CPU alebo systémovou pamäťou, medzi 2 rôznymi perifériami, medzi dvomi bránami tej istej periférie alebo loop back medzi USB endpointami. Logické dátové cesty nevedú cez pamäť. V praxi však všetky prenosy dát prechádzajú cez systémovú pamäť.

Celým mikrokontrólerom je vedená pokročilá zbernicová architektúra – *Advanced High Performance Bus (AHB)*, ktorá umožňuje jednoduchú integráciu procesora, pamätí a iných periférií. Poskytuje jednoduché spojenie medzi prvkami zapojenými do väčšiny prenosov.

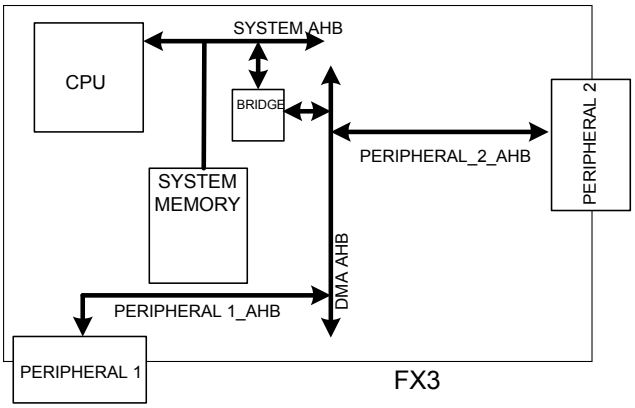
Z pohľadu DMA môže byť periféria rozdelená na tri časti: *DMA adaptér*, *radič vlákna* a *periférne jadro*. V *perifémom jadre* je implementovaná logika periférie. Periféria je pripojená do DMA siete pomocou AHB, ktorej šírka určuje priepustnosť periférie.

Periférie pri komunikácii s vonkajším svetom môžu byť adresovateľné. Jednotlivé adresy špecifikovateľné zvonku sú naviazané na entity nazvané sokety. Sokety sú logicky prezentované ako reťazce buffrov. Podľa potreby aplikácie môžu byť reťazce buffrov zostrojené rôznymi spôsobmi. Buffre nie sú uložené v periférіach. Sú alokované v systémovej pamäti a soketu, ktorému je buffer pridelený je vytvorený ukazovateľ.

Niektoré periférie môžu obsahovať desiatky soketov, ktorým je priradený iný reťazec buffrov. Aby bola dosiahnutá rozumná priepustnosť DMA prenosov cez sokety a udržaná rozumná šírka zbernice, nie sú požiadavky soketov v časovom multiplexe. Sokety sú zoskupené do vlákien, ktorých požiadavky sú v časovom multiplexe. Vzťahy medzi vláknami a soketmi sú riadené *radičom vlákien*. *DMA adaptér* konvertuje požiadavky vlákna na čítanie / zápis na požiadavky o AHB a spúšťa ich. Jeden *radič vlákna* a *DMA adaptér* môžu byť zdieľané niekoľkými perifériami.

Soket môže byť určený buď na čítanie alebo zápis, nie na oboje. Sokety, ktoré zapisujú do systémových buffrov sa nazývajú producenti a tie, ktoré z nich čítajú konzumenti. Každý soket má množinu registrov, ktoré indikujú stav soketu, počet prenesených bajtov, aktuálny napĺňaný alebo vyprázdňovaný buffer a aj to, že žiadny buffer nie je k dispozícii.

DMA prenos si môžeme predstaviť ako tok dát z produkujúceho soketu do konzumujúceho soketu cez buffre v systémovej pamäti. Synchronizácia medzi soketmi je umožnená špeciálnymi DMA inštrukciami – deskriptormi. Každý deskriptor obsahuje informáciu o jeho adrese, plnom / prázdnom stave a ďalšom buffri / deskriptore v reťazci. Deskriptory sú uložené v špeciálnom mieste v systémovej pamäti a sú monitorované soketmi periférií. [10]



Obrázek 3.8: Zbernicová sieť na mikrokontroléri FX3. [10]

3.5.3 GPIF II

GPIF II je rozhranie, ktoré poskytuje jednoduché prepojenie FX3 kontroléru s rozšírenými rozhraniami ako asynchrónna SRAM, synchrónna SRAM, spojené adresové a dátové rozhranie, PATA... GPIF II vie pracovať na frekvencii 100 MHz. Obsahuje 45 pinov, ktoré môžu byť naprogramované pre dáta, adresy, riadiace signály alebo vstupno/výstupnú funkciu.

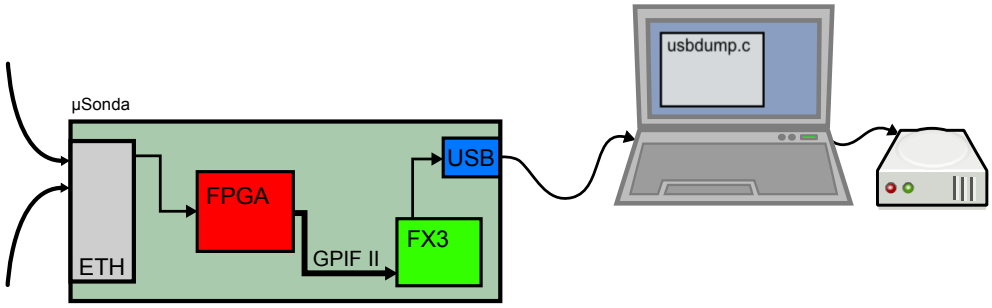
Srdcom GPIF II je stavový automat, schopný uchovať 256 stavov. Vie monitorovať 32 vstupných signálov pre prechod medzi stavmi. Má podporu prechodov do dvoch rôznych stavov zo súčasného stavu. Vie generovať 33 rôznych riadiacich signálov.

GPIF II nie je priamo pripojené k bufferom USB endpointov. Namiesto toho je pripojené do DMA siete. To umožňuje procesoru väčšiu kontrolu nad dátovým tokom.

Kapitola 4

Implementačná časť

V rámci mojej práce boli implementované 3 časti, ktoré sú zobrazené na obrázku 4.1. Dáta prijímané z dvoch ethernetových rozhraní μSondy sú spracované v FPGA. Následne sú z FPGA odosielané do FX3 mikrokontroléra pomocou rozhrania GPIF II. Prvou implementovanou časťou je *FPGA komponenta*, ktorá riadi prenos dát cez toto rozhranie. Druhú časť tvorí *firmware mikrokontroléra FX3*, ktorý prijíma dáta z rozhrania GPIF II a odosiela ich pomocou USB rozhrania do PC. Poslednou časťou je *aplikácia pre PC*, ktorá prijíma dáta z USB 3.0 rozhrania a ukladá ich na pevný disk počítača.



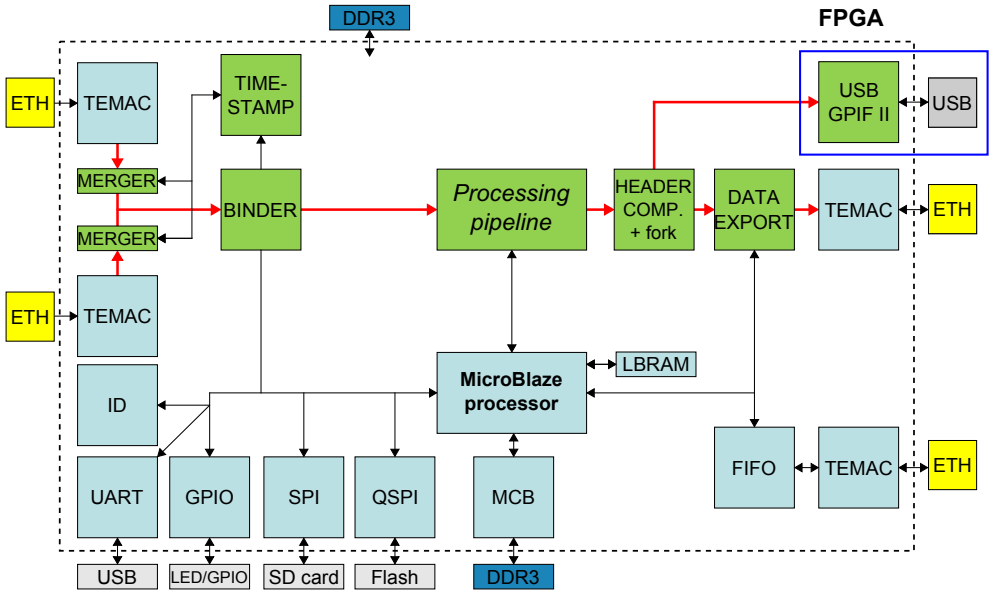
Obrázek 4.1: Schematické znázornenie častí implementovaných v mojej práci. [5]

4.1 Činnosť μSondy a jej komponent

Na obrázku 4.2 pakety vstupujú z dvoch ethernetových vstupov vpravo do FPGA. Dva vstupy sú kvôli zachytávaniu oboch smerov komunikácie. Komponenta *TEMAC* implementuje MAC vrstvu. Jednotka *Timestamp* generuje presné časové značky, ktoré sú pripojené k prijatým paketom. Komponenta *Merger* spája dátový a kontrolný stream. Uloží veľkosť prijatého paketu a presnú časovú značku do INI3 hlavičky, ktorá je popísaná v kapitole 4.4.3. V *Binderi* sa spoja toky z oboch TEMACov.

Processing pipeline je tvorená jednotkou *HFE-X* a *Filtrom*. *HFE-X* má za úlohu vyextrahovať relevantné časti z hlavičiek paketov: IP adresy, protokol a prípadne porty. Tieto informácie sú predané do klasifikačnej jednotky – *Filtra*. Ten na základe vyextrahovaných častí hlavičky rozhodne, či paket zodpovedá niektorému pravidlu podľa nastavených odpočúvaní. Ak áno, je paket prepustený ďalej, v opačnom prípade je zahodený.

Za *Processing pipeline* nasleduje jednotka, ktorá dokončí INI3 hlavičku: pridáva *RID – reason ID*, ktoré reprezentuje pravidlo, na základe ktorého bol paket prepustený. Podľa



Obrázek 4.2: Komponenty implementované v FPGA μSondy. [2]

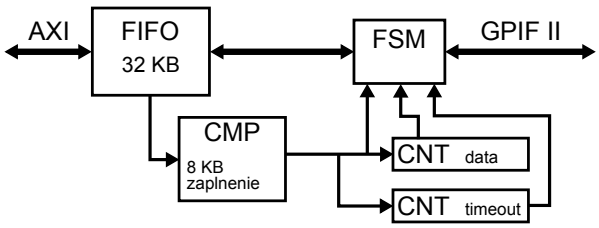
nastavenia je paket buď poslaný do komponenty riadiacej prenos cez USB rozhranie, alebo je ďalej zabalený do TCP / UDP paketu a odoslaný von cez ethernet.

Posledný ethernetový port spojený s procesorom *MicroBlaze* cez *FIFO* slúži ako konfiguračné rozhranie μSondy. V ľavom dolnom rohu blokovej schémy sú menej zaujímavé komponenty, ale potrebné na chod mikrosondy: komunikácia s flash pamäťou, komunikácia s SD kartou, sériová linka, GPIO a identifikácia designu.

V tejto práci je implementovaná komponenta riadiaca prenosy dát z FPGA do mikrokontroléra FX3 cez rozhranie GPIF II. Na obrázku 4.2 je vyznačená modrým rámikom.

4.2 FPGA komponenta

Prvou časťou implementovanou v rámci mojej práce je FPGA komponenta, ktorá riadi prenos dát z FPGA do mikrokontroléra FX3 cez GPIF II rozhranie. Aby bola dosiahnutá čo najvyššia priepustnosť, je nutné zabezpečiť na strane FX3 čo najnižšiu réžiu. Dáta preto nesmú byť posielané z FPGA do mikrokontroléra po malých kúskoch – každý paket zvlášť. Namiesto toho by mali byť posielané po väčších zhlukoch pevnej veľkosti. Veľkosť buffra v FX3 je nastavená na 8 KB, komponenta preto posiela dáta zarovnané na 8 KB bloky.

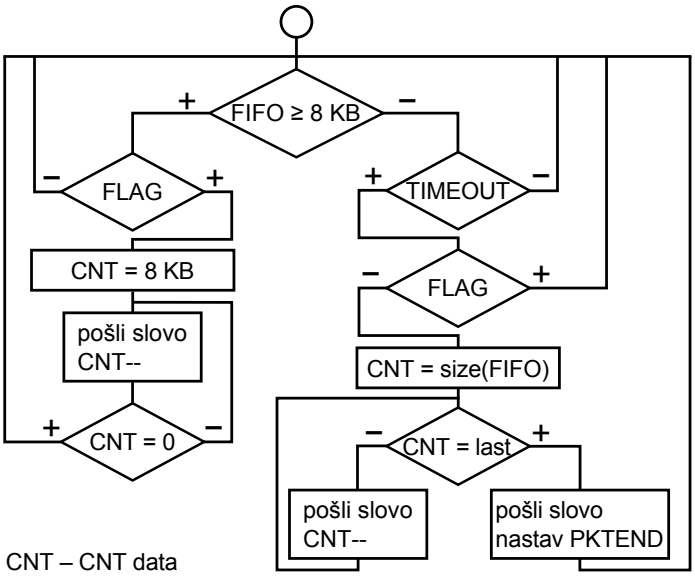


Obrázek 4.3: Štrukrúra komponenty riadiaca prenosy z FPGA.

Komponenta je zobrazená na obrázku 4.3. Je tvorená jednotkou FIFO, schopnou uchovať 32 KB dát. Ďalej sa tu nachádza komparátor CMP, ktorý sleduje zaplnenie FIFO jednotky.

Ďalšou časťou je stavový automat FSM, ktorý riadi prenosy dát cez GPIF II. Poslednými časťami sú dva čítače, jeden určený pre timeout a druhý počíta dátové slová k odoslaniu.

Dáta do komponenty prichádzajú z AXI-Stream rozhrania a hromadia sa v jednotke FIFO. Komparátor porovnáva aktuálne zaplnenie FIFO jednotky. Ak zaplnenie presiahne 8 KB, je odštartovaný prenos. Aby dáta nestáli vo FIFO jednotke pri nízkom toku dát príliš dlho, sú odoslané po vypršaní timeoutu aj v počte menšom ako 8 KB. Čítač pre timeout sa reštartuje po každom odoslaní dát. Druhý čítač zabezpečuje, aby bolo odoslaných presne 8 KB dát.



Obrázek 4.4: Vývojový diagram stavového automatu komponenty.

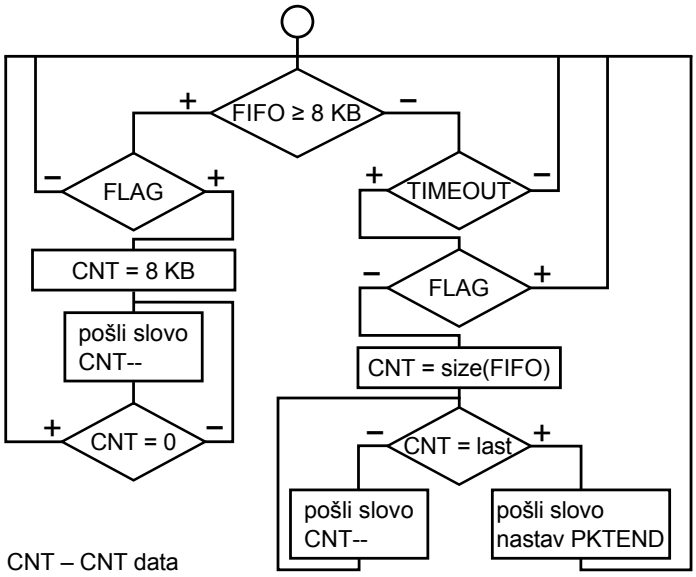
Činnosť stavového automatu znázorňuje obrázok 4.5. Ak je v jednotke FIFO viac ako 8 KB dát, skontroluje sa FLAG. FLAG je riadiaci signál posielaný z mikrokontroléra FX3, ktorý signalizuje pripravenosť pre zápis. Ak je možné zapisovať, nastaví sa čítač dát pre odoslanie 8 KB. Po odoslaní dát sa automat vracia do začiatočného stavu.

Ak je v jednotke FIFO menej ako 8 KB dát a dôjde k timeoutu – timeout čítač dosiahne nulu, opäť sa skontroluje FLAG. Ak môže zapisovať, dátový čítač je nastavený na veľkosť dát aktuálne uložených vo FIFO jednotke. Následne sa odosielajú jednotlivé dátové slová. Pri odosielaní posledného slova je nastavený signál PKTEND, ktorý signalizuje mikrokontroléru FX3 predčasné ukončenie zhlukového prenosu.

4.3 FX3 firmware

Cypress dodáva ku svojím mikrokontrolérom bohatú paletu nástrojov umožňujúcu rýchly vývoj aplikácií. Súčasťou balíku SDK je vývojové prostredie postavené na platforme Eclipse, ukážkové aplikácie a nástroj GPIF II designér, ktorý umožňuje prispôsobiť GPIF II rozhranie aplikačným potrebám.

Aplikácie na mikrokontroléri FX3 bežia pod realtime operačným systémom ThreadX. Vďaka tomu je možné rozdeliť obsluhu periférií do vlákien. Vo firmwari ďalej prebiehajú tieto udalosti: inicializácia periférií, zistenie rýchlosti komunikácie po USB a podľa nej nastavenie veľkosti USB paketu, nastavenie jednotlivých DMA kanálov. Nachádza sa tu aj



Obrázek 4.5: Vývojový diagram stavového automatu komponenty.

vlákno upravujúce režimy napájania. Aplikácia umožňuje nastaviť si vlastné deskriptory v zariadení: Product ID a Vendor ID.

Úprava firmwaru pre použitie v USONDE spočívala v nastavení DMA kanálu. Kanál som nastavil na automatický režim, pri ktorom procesor nezasahuje do prenosov dát. Prenos dát je v plnej réžii DMA. Ako soket producenta som nastavil GPIF II rozhranie a ako konzumenta rozhranie USB. Pre dosiahnutie čo najvyššej priepustnosti som nastavil najvyššiu možnú veľkosť buffra. Veľkosť jedného buffra som nastavil na 8 KB a ich počet na 27.

Ďalej bolo potrebné upraviť nastavenie FLAGu, ktorý na GPIF II rozhraní signalizuje stav zaplnenia FX3. Mikrokontrolér má latenciu 3 takty GPIF II zbernice kým nahodí signál zaplnenia FLAG. Počas týchto 3 taktov by FPGA zapisovalo do plného buffra. Je preto nevyhnutné nastaviť signál FLAG skôr než dôjde k zaplneniu. Signál teda nastavujem o 4 takty skôr, pretože kým ho FPGA zaregistruje, trvá to jeden takt.

4.4 Aplikácia pre PC

Poslednou implementačnou časťou mojej práce bolo vyvinúť a naprogramovať aplikáciu určenú na príjem a ukladanie dát zaznamenaných μSondou. Jedná sa o jednoduchú aplikáciu, ktorej hlavným cieľom je prijať dáta z mikrokontroléra FX3 cez USB rozhranie a uložiť ich do súboru na disku. Keďže μSonda musí pracovať na plnej priepustnosti 1 Gbps linky v oboch smeroch, teoretický najvyšší možný dátový tok je 2 Gbps. Pri návrhu aplikácie som preto kládol dôraz na čo najvyššiu priepustnosť USB rozhrania. Funkčnosť je rozdelená do 2 vlákien: prvé prijme dáta z USB rozhrania a uloží do pamäte, druhé dáta spracuje a uloží na disk. Aplikácia je navrhnutá tak, že pri výpadku spojenia, napríklad odpojením kábla, čaká na jeho obnovenie. Keď je spojenie opäť obnovené, pokračuje v prijme dát. Dáta prijímané zo sondy sú vo formáte INI3 používaným v systéme pre zákonné odpočúvanie. Aplikácia tento formát konvertuje na formát PCAP, ktorý je určený na ukladanie paketov zachytených sieťovým analyzačným programom ako je napríklad Whireshark. Aplikáciu som písal v jazyku C pre operačný systém Linux s využitím knižnice CyUSB.

4.4.1 Knižnica CyUSB

Knižnica *cyusb* je súčasťou balíka nástrojov *CyUSB Suite for Linux*, ktorý vydal Cypress pre pomoc a urýchlenie vývoja klientských aplikácií pre svoje USB mikrokontroléry. Knižnica *cyusb* je postavená nad knižnicou *libusb*, ktorá klientským aplikáciám uľahčuje prístup k USB zariadeniam. Knižnica *libusb* je open source, je napísaná v jazyku C pre rôzne operačné systémy. Balík *CyUSB Suite for Linux* okrem knižnice *cyusb* obsahuje aj rôzne nástroje potrebné pre vývoj a testovanie aplikácií. Zahŕňa to ovládače, zdrojové kódy demonštrujúce funkcie knižnice, dokumentáciu a skripty spúšťané pri pripojení Cypress zariadenia. Obsahuje tiež aplikáciu ktorá umožňuje testovanie prenosov dát a nahratie firmware do mikrokontroléra FX3. Zo skriptov sú zaujímavé dva. Skript *88-cyusb.rules* obsahuje jednoduché pravidlá pre UDEV, ktoré pri detegovaní pripojenia alebo odpojenia Cypress zariadenia spustí skript *cy_renumerate.sh*. Skript *cy_renumerate.sh* posiela signál SIGUSR1 klientskej aplikácii. Moja aplikácia je písaná s využitím knižnice *cyusb*, preto je potrebné nainštalovať ju na cieľový systém.

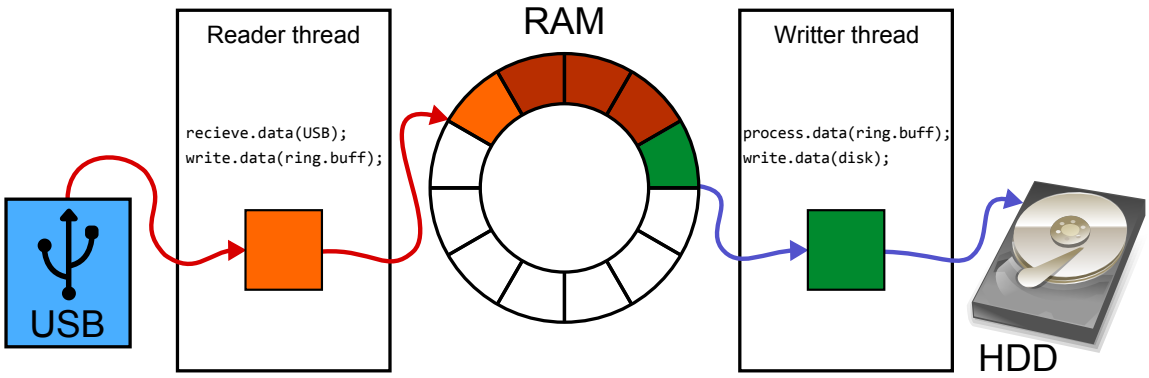
4.4.2 Návrh algoritmu

Účelom mojej aplikácie je prijať dáta z rozhrania USB, spracovať ich a uložiť na disk do súboru. Keďže najvyšší dátový tok z μSondy môže dosiahnuť hodnotu 2 Gbps, hlavnou prioritou bolo dosiahnuť čo najvyššiu priepustnosť z USB rozhrania. Väčšina dnešných počítačov disponuje viacjadrovými CPU. Pri návrhu som sa rozhodol rozdeliť úlohy do dvoch vlákien, aby proces prijímania dát z USB rozhrania nebol obmedzovaný ich spracovaním a uložením na disk.

Prijaté dáta sú dočasne ukladané do kruhového buffra v pamäti. Zdroje pre kruhový buffer sú vyhradené dopredu a o pevnej veľkosti. Alokácia pamäte nemá vplyv na rýchlosť spracovania dát. Spracovanie prebieha jedným priechodom, pri ktorom sú hlavičky INI3 paketov konvertované na PCAP hlavičky. Payload paketov ostáva nezmenený a je zapísaný do súboru spolu s upravenou PCAP hlavičkou. Hlavička má veľkosť 16 bajtov a jej úprava pozostáva z bitových operácií. Spracovanie dát teda spočíva len v prepísaní 16 bajtov v pamäti na jeden prijatý paket a uložení na disk. Kruhový buffer je v pamäti reprezentovaný poľom menších pamäťových blokov. Veľkosť jedného bloku je rovnaká ako veľkosť dát prijatých volaním pre príjem dát z USB rozhrania pomocou bulk prenosu. Veľkosť bloku a aj počet blokov je nastaviteľný pri preklade aplikácie a dá sa prispôsobiť zdrojom dostupným na cieľovej architektúre.

Pre rozdelenie funkčnosti aplikácie som si zvolil vlákna, pretože predstavujú realizáciu procesu na jemnejšej úrovni. Zdieľajú pamäť, premenné, deskriptory a ostatné systémové zdroje. Funkčnosť aplikácie som podľa priority rozdelil do dvoch vlákien. S rodičovským vláknom, ktoré sa stará o doplnkovú funkčnosť beží aplikácia celkom v troch vláknach. Rodičovské vlákno zabezpečuje spracovanie parametrov, inicializuje premenné. Spúšťa ďalšie 2 vlákna hneď po spustení aplikácie alebo po obdržaní signálu o pripojení USB zariadenia. Jeho činnosť je potom pozastavená. Čítacie vlákno má za úlohu ustaviť spojenie s USB zariadením pomocou volaní z knižnice cyusb. Následne v nekonečnom cykle prijíma dáta z USB rozhrania pomocou volania zabezpečujúceho bulk prenos. Prijaté dáta ukladá do kruhového buffra v pamäti. Zapisovacie vlákno jedným priechodom spracuje uložené dáta a uloží ich na disk. Parsovanie hlavičiek paketov obsluhuje stavový automat.

μSonda posiela v rámci jednotlivých svojich komponent pakety zarovnané na 4 bajty. Je to preto, že zbernice po ktorých sú pakety posielané majú šírku 32 bitov, tak isto aj GPIF II má šírku 32 bitov. Neexistuje jednoduchý spôsob ako predať informáciu, ktoré



Obrázek 4.6: Znázornenie činnosti vlákien v PC aplikácii. [5]

bajty z jedného slova sú platné, pakety sú preto zarovnané na 4 bajty. Pokiaľ ich veľkosť v bajtoch nie je deliteľná 4, sú na koniec pridané jeden, dva alebo 3 nulové bajty. Tieto nulové bajty je potrebné pri ukladaní do formátu PCAP odrezať, pretože by ich sieťový analyzátor nerozpoznal. Dáta je preto potrebné zapisovať do súboru po jednotlivých paketoch. Na urýchlenie zápisu na disk som použil vektorový zápis do súboru.

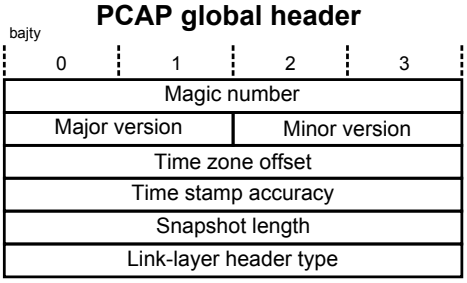
4.4.3 Formáty INI3 a PCAP

Aby bolo umožnené pohodlné prezeranie a analyzovanie dát zachytených μSondou, je potrebné tieto dáta previesť z formátu INI3 na formát PCAP. Formát INI3 je používaný v systéme pre zákonné odpočúvanie a μSonda exportuje prúd paketov v tomto formáte. Formát PCAP je používaný nástrojmi pre analýzu a ukladanie sieťových dát. Existujú knižnice pre ukladanie sieťových dát v PCAP formáte. Prevod formátu INI3 na formát PCAP je však natoľko jednoduchý, že použitie týchto knižníc by bolo zložitejšie ako riešenie implementované v aplikácii.

Oba formáty sa skladajú z *globálnej hlavičky*, ktorá sa nachádza na začiatku prúdu alebo súboru. Nasledujú samotné zachytené dáta v podobe *hlavička paketu* a za ňou samotné zachytené dáta – *payload paketu*. Payload paketu je v oboch formátoch rovnaký a nie je v ňom potrebné robiť zmenu. Hlavičky paketov majú rovnakú veľkosť, ktorá je 16 bajtov. Pri prevode je teda potrebné prepísať INI3 hlavičku na PCAP hlavičku a pripojiť k nej payload. INI3 formát má globálnu hlavičku, tá sa však v prúde dát posielaných μSondou na USB rozhranie nevyskytne. Globálna hlavička formátu PCAP má 24 bajtov. Aby sieťový analyzátor rozpoznal endianitu a iné vlastnosti zachytených dát, je potrebné aby bola na uložená na začiatku každého PCAP súboru.

PCAP globálna hlavička

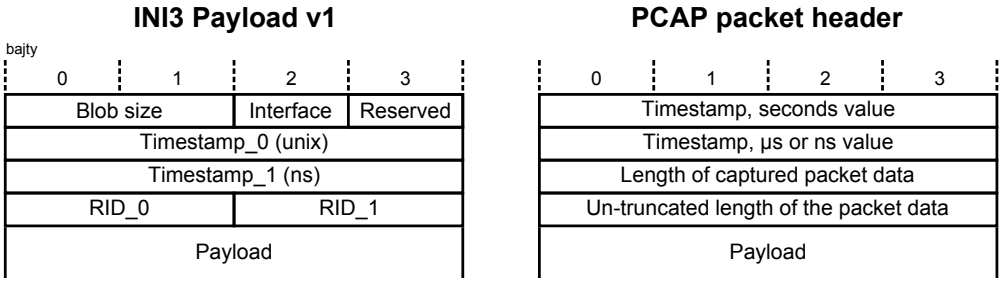
Na začiatku globálnej hlavičky je *Magic number* o dĺžke 4 bajtov, ktoré určuje endianitu uloženia dát a presnosť časových značiek. Konštanta `0xa1b2c3d4` znamená uloženie ďalších dát vo formáte *Big Endian*. Ak je poradie bajtov opačné – `0xd4c3b2a1`, reprezentácia dát je v *Little Endian*. Časové značky sú v tomto prípade uložené s presnosťou na mikrosekundy. Presnosť na nanosekundy pri *Big Endian* je určená tak, že poradie dvoch nibblov dvoch nižších bajtov je vymenené. Pre *Big Endian* to je ako `0xa1b23c4d` a pre *Little Endian* `0x4d3cb2a1`. μSonda posiela INI3 hlavičky vo formáte *Little Endian*. Časové značky ukladám s presnosťou na mikrosekundy, preto som *Magic number* nastavil na `0xd4c3b2a1`.



Obrázek 4.7: Globálna hlavička, umiestnená na začiatku PCAP súboru. [1]

Ďalej nasledujú dve dvojbajtové hodnoty udávajúce verziu. Verzia implementovaného PCAP formátu nesie označenie 2.4, preto do poľa *Major version* vložím `0x0200` a do *Minor version* `0x0400`. Ďalšou hodnotou na 4 bajtoch je vyrovnanie časového pásma udávané v sekundách. Za ňou nasledujú 4 bajty, ktoré určujú presnosť časových značiek. Obe tieto hodnoty sú implicitne nula, preto za hodnotami verzie nasleduje 8 nulových bajtov. *Snapshot length* určuje maximálnu veľkosť zaznamenaného paketu v bajtoch. INI3 formát má veľkosť paketu uloženú na 2 bajtoch. Nepredpokladám preto, že by bol μSondou prijatý paket o veľkosti väčšej ako 65535 bajtov a nastavím hodnotu *Snapshot length* na `0xffff0000`. Posledná hodnota globálnej hlavičky určuje typ hlavičky linkovej vrstvy. Túto som nastavil na `0x01000000`, čo znamená hlavička Ethernetu. [1]

Hlavičky paketov



Obrázek 4.8: Porovnanie hlavičiek paketov vo formáte INI3 a PCAP. [3][1]

Zachytené dáta sú z μSondy posielané protokolom INI3 verzie 1. *Blob size* uložený na prvých 2 bajtoch určuje veľkosť zachyteného paketu v bajtoch. Hodnota *Interface* o veľkosti 1 bajt určuje rozhranie μSondy, na ktorom bol paket zachytený. Za touto hodnotou je 1 bajt vyhradený na použitie do budúcnosti. Nasledujú dve 4 bajtové hodnoty časovej značky. Prvá určuje počet sekúnd od začiatku Unixovej éry, teda od 01.01.1970. Druhá je spresnenie času zachytenia na jednotky nanosekúnd. Ďalej nasledujú 2 dvojbajtové hodnoty *RID – Reason ID*. Tieto hodnoty určujú číslo pravidlá filtra, ktorými bol paket zachytený. [3]

Hlavička PCAP formátu má 4 hodnoty o veľkosti 4 bajtov. Prvá je unixová časová značka udávaná v sekundách. Druhá hodnota je spresnenie časovej značky na mikrosekundy alebo nanosekundy. Nasledujú dve hodnoty veľkosti zachyteného paketu. Prvá určuje skutočnú veľkosť paketu, ktorá bude uložená v PCAP súbore. Druhá udáva veľkosť, ktorú by mal paket, keby nebol skrátený na najväčšiu možnú veľkosť udávanú v *Snapshot length*. Ak je táto veľkosť menšia alebo rovná *Snapshot length*, tak sú obe hodnoty rovnaké. [1]

Algoritmus prevodu

Pri prevode INI3 hlavičky na PCAP hlavičku považujem oblasť pamäti s hlavičkou za pole bajtov o veľkosti 16. Najprv načítam veľkosť z hodnoty Blob size. Pretože pracujem s dátami v reprezentácii *Little Endian*, načítanie hodnoty prebieha tak, že k unsigned premennej, ktorá je inicializovaná na nulu pripočítam hodnotu druhého bajtu *Blob size*. Následne spravím bitový posun o 8 bitov doľava a pripočítam hodnotu prvého bajtu *Blob size*.

V druhom kroku skopírujem 4 bajty unixovej časovej značky z pozícií 4 až 7 na pozíciu 0 až 3. Táto hodnota a jej uloženie je v oboch formátoch rovnaké. Ďalej načítam do premennej hodnotu nanosekundovej presnosti. Načítanie prebieha rovnakým spôsobom ako načítanie hodnoty veľkosti. Do vynulovanej unsigned premennej s rozsahom 32 bitov pripočítam najprv hodnotu najvyššieho významového bajtu z pozície 11. Nasleduje bitový posun o 8 bitov doľava a pripočítanie hodnoty bajtu z pozície 10. Tak isto sú pričítané aj ďalšie 2 bajty.

Spresňujúcu časovú značku ukladám v PCAP formáte v mikrosekundách. Hodnotu premennej v nanosekundovej presnosti preto vydelím 1000 a uložím ju v hlavičke na pozície 3 až 7. Uloženie prebieha v správnom poradí pomocou bitového posunu doprava a operácie bitového AND s konštantou `0xFF`.

Podobne pomocou bitového posunu doľava a bitového AND uložím hodnotu veľkosti paketu na pozície 8 a 9. Pozície 10 a 11 doplním nulovými bajtmi. Rovnaký postup aplikujem aj pre zápis hodnoty veľkosti neskráteného paketu na pozíciach 12 až 15. Obe hodnoty veľkostí sú zhodné.

4.4.4 Kruhový buffer a synchronizácia vlákien

Kruhový buffer je implementovaný ako pole štruktúr. Štruktúra obsahuje vyhradený *blok pamäte* pre zápis dát, *premennú* reprezentujúcu počet zapísaných bajtov a *mutex* povolujúci prístup len jednému vláknu. Obe vlákna začínajú pristupovať k položkám buffra od indexu nula a keď dosiahnu posledného prvku, idú opäť od nuly. Tým je zabezpečená kruhovosť buffra. Buffer obsahuje ešte jeden *globálny semafór*, ktorý určuje počet voľných blokov.

Čitateľ

```
i = 0
wait(sem)
lock(i)
obsluha(i)
i++
štart-zapisovateľ()

while(!error){
  wait(sem)
  lock(i)
  unlock(i - 1)
  obsluha(i)
  i = (i + 1) % BUFFCOUNT
}
```

Zapisovateľ

```
j = 0
pause()




while(!error){
  lock(j)
  obsluha(j)
  unlock(j)
  post(sem)
  j = (j + 1) % BUFFCOUNT
}
```

Obrázek 4.9: Pseudokód vlákien čitateľa a zapisovateľa.

Synchronizácia vlákien je zabezpečená tak, že pri prístupe do kruhového buffra vlákno zapisovateľa nemôže predbehnúť vlákno čitateľa. Pri štarte oboch vlákien pred zahájením prenosu, je vlákno zapisovateľa pozastavené. Vlákno čitateľa sa pokúsi nadviazať spojenie

s FX3 mikrokontrolérom. Ak je spojenie nadviazané, prvý blok dát je prenesený do kruhového buffra. Pri tomto kroku je taktiež globálny semafór reprezentujúci počet voľných blokov znížený o jedna.

Po dokončení prenosu je obnovená činnosť zapisovateľa. Ten sa pokúša pristúpiť k prvému bloku buffra, ale stojí na mutexe, pretože ho čitateľ neodomkol. Čitateľ v tomto momente vstúpi do cyklu, kedy najprv zamkne mutex nasledujúceho voľného bloku a až potom odblokuje mutex predchádzajúci. Teda ten, ku ktorému sa snaží pristúpiť zapisovateľ. Čitateľ teraz zapisuje dáta do ďalšieho bloku a zapisovateľ spracováva dáta z bloku, kde čitateľ pred tým zapísal.

Ak by zapisovateľ nestíhal spracovávať dáta a čitateľ by naplnil všetky dostupné bloky buffra, mohlo by sa stať, že by predbehol zapisovateľa a začal by prepisovať nespracované bloky. Preto je prítomný v buffri semafór, ktorý reflektuje aktuálny počet voľných blokov. Čitateľ ho pred zápisom vždy zníži a zapisovateľ po dokončení spracovania bloku zvýši.

Vlákno zapisovateľa predbehne vlákno čitateľa len v prípade, ak nastane chyba pri prenose dát a vlákno čitateľa je ukončené. V tomto prípade pristúpi zapisovateľ k bloku buffra, ktorý je označený ako nepoužitý. Ak je nastavený príznak chyby prenosu, je vlákno zapisovateľa tiež ukončené.

4.4.5 Stavový automat

Pri spracovávaní dát po blokoch môžu nastať situácie, kedy prijaté pakety nebudú presne zarovnané v jednotlivých blokoch, ale budú rozdelené medzi blokmi. Tieto situácie riešim stavovým automatom.

Sú tri spôsoby zarovnania prijatých paketov v jednotlivých blokoch kruhového buffra. Pakety môžu byť v dvoch blokoch zarovnané – jeden paket končí v jednom bloku a ďalší začína v druhom. Alebo je jedna časť hlavičky paketu v jednom bloku a jej pokračovanie v nasledujúcom bloku. Posledná možnosť je, že má paket rozdelený payload v dvoch blokoch. Stavový automat má 3 hlavné stavy: zarovnané dáta, rozdelená hlavička a rozdelený payload.

Stav zarovnané dáta je východzí stav, automat v ňom trávi najviac času, pretože všetky pakety vo vnútri bloku, okrem prvého a posledného paketu budú vždy obslúžené týmto stavom. Z tohto stavu je najprv kontrolované, či sa hlavička nasledujúceho paketu nachádza celá v aktuálnom bloku. Ak sa nenachádza, je časť hlavičky, ktorá je v aktuálnom bloku skopírovaná do dočasnej premennej. Automat prechádza do stavu *rozdelenej hlavičky* a posúva sa na ďalší blok kruhového buffra.

Ak sa hlavička paketu nachádza celá v aktuálnom bloku, je spracovaná. Ďalej je na prítomnosť v aktuálnom bloku kontrolovaný payload. Ak payload nie je celý uložený v aktuálnom bloku, je do súboru zapísaná hlavička a časť payloadu prítomná v aktuálnom bloku. Automat prechádza do stavu *rozdelený payload* a pokračuje so spracovaním ďalšieho bloku buffra.

Ak je payload paketu prítomný v aktuálnom bloku, je spolu s hlavičkou paketu zapísaný do súboru. Automat prejde znovu do stavu *zarovnané dáta* a pokračuje spracovaním ďalšieho paketu z aktuálneho bloku.

V stave rozdelený payload automat najprv skontroluje, či je v aktuálnom bloku uložený celý zvyšok payloadu, potrebný pre dokončenie zápisu z predchádzajúceho bloku

kruhového buffra. Ak nie je zvyšok celý, je do súboru zapísaná časť payloadu nachádzajúca sa v aktuálnom bloku. Automat opäť prejde do stavu *rozdelený payload* a pokračuje spracovaním nasledujúceho bloku.

Ak je prítomný celý zvyšok payloadu v aktuálnom buffri, automat ho zapíše do súboru a prechádza do stavu *zarovnaná dáta*. Pokračuje v aktuálnom bloku.

V stave rozdelená hlavička automat kontroluje, či je zvyšok hlavičky prítomný v aktuálnom bloku. Ak nie je, uloží časť hlavičky v aktuálnom bloku do dočasnej premennej. Automat prechádza do stavu *rozdelenej hlavičky* a pokračuje spracovaním nasledujúceho bloku kruhového buffra.

Ak je celý zvyšok hlavičky prítomný v aktuálnom bloku, je tento zvyšok skopírovaný do dočasnej premennej. Následne je hlavička spracovaná v dočasnej premennej. V ďalšom kroku je skontrolovaná prítomnosť celého payloadu. Ak sa v aktuálnom bloku nenachádza celý payload, je do súboru zapísaná hlavička z dočasnej premennej a časť payloadu nachádzajúca sa v aktuálnom bloku. Automat prechádza do stavu *rozdelený payload* a pokračuje spracovaním nasledujúceho bloku buffra.

Ak je celá časť payloadu prítomná v aktuálnom bloku, je do súboru zapísaná hlavička z dočasnej premennej spolu s payloadom. Automat prechádza do stavu *zarovnané dáta* a pokračuje spracovávaním aktuálneho bloku kruhového buffra.

Vektorový zápis do súboru

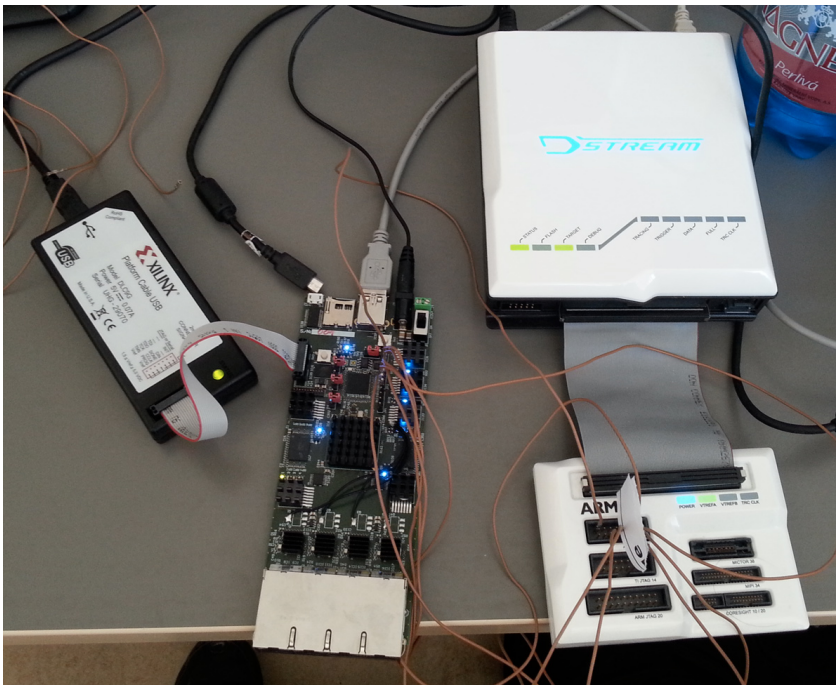
Ak stavový automat zapisuje dáta do súboru, neznamená to, že ich zápis prebieha okamžite. Namiesto toho sú do systémovej štruktúry uložené ukazovatele na miesto v pamäti, odkiaľ majú byť dáta zapísané a aj ich počet. Štruktúra dokáže takto uchovať 1024 položiek pre zápis. Pri dosiahnutí tohto počtu alebo pri dokončení spracovania bloku kruhového buffra sú dáta zapísané na disk naraz.

Pri zápise na disk je zapísaná len platná časť paketu. Zapísané nie sú nulové bajty, ktoré sa môžu vyskytnúť na konci paketu a sú pridané kvôli prenosu po 32 bitových zberniciach μSondy.

Kapitola 5

Testovanie

Počet zabraných zdrojov komponentou v FPGA časti je: LUT: 244, FF: 116, block ram: 16. Maximálna frekvencia komponenty dosahuje 198 MHz.



Obrázek 5.1: Testovanie FX3 mikrokontroléru na μSonde pomocou JTAG debuggera D-STREAM.

Komunikácia μSondy bola testovaná na vzorke zachytenej sieťovej prevádzky v PCAP súbore. Tá bola posielaná do μSondy pomocou nástroja Tcpreplay. Najprv bol prenos dát testovaný pri nízkych rýchlostiach aby sa overila správnosť prijatých dát. Porovnanie originálneho a zachyteného PCAP súboru bolo realizované pomocou hex editora, ale aj pomocou nástroja diffPCAP.

Následne bol systém testovaný na dosiahnutie čo najvyššej priepustnosti. Architektúra na ktorej bola testovaná aplikácia pre príjem dát na čo najvyššiu priepustnosť: procesor: Intel(R) Core(TM) i5-2500 CPU @ 3.30 GHz, RAM: 8 GB DDR3, HDD: Western Digital RE2 500 GB.

Testovaný bol prenos 4 GB dát a cieľom bolo mať nulovú stratu prijatých paketov na

strane príjemcu. Priepustnosť bez straty paketu dosiahla hodnotu 344 Mbit/s. Komunikácia prebiehala cez USB 2.0 rozhranie a táto rýchlosť sa blíži jej teoretickému maximu.

Komunikáciu cez USB 3.0 nebolo možné nadviazať z dôvodov popísaných v nasledujúcej kapitole. Teoretické maximum priepistnosti v implementovanom systéme som preto nebol schopný zmerať.

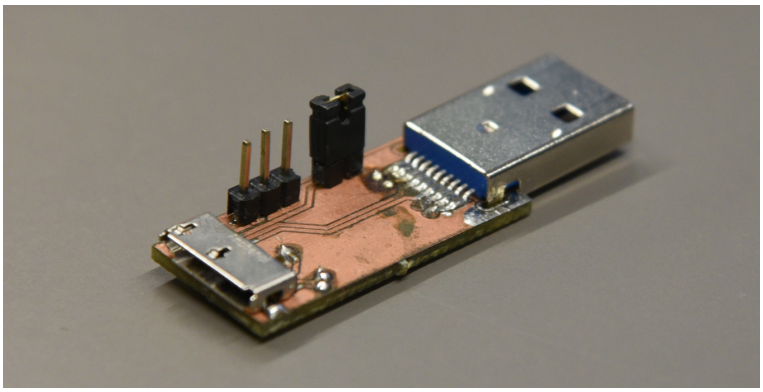
Záťaž procesora bola s použitím vektorového zápisu na disk o 50 % nižšia.

5.1 Problémy

V pôvodnom návrhu bolo zamýšľané použitie mikrokontroléra FX3 na v režime host. Preto bol na μSondu umiestnený konektor typu A namiesto štandardného konektoru pre zariadenie typu B alebo micro-B. Prepojenie s počítačom musí byť preto realizované použitím USB 3.0 A-A kábla. Tento kábel však podľa špecifikácie nemá prepojené dôležité linky, konkrétne VBUS a USB 2.0 linky. Spojenie s počítačom preto nie je pomocou tohto kábla nadviazané.

Na konektore nie je zapojený signál OTG_ID. Používa sa s konektorom micro-B pre detekciu OTG. Tým pádom sa FX3 vie prepnúť z režimu device do režimu host.

Pamäť I^2C EEPROM, určená pre uloženie firmwaru pre FX3 je špatne zapojená. Z tohto dôvodu nie je možné permanentne uložiť firmware v USONDE. Pred použitím sa musí nahrať. Aby bolo možné komunikovať s FX3 cez USB 3.0, bola vyvinutá prepojka zobrazená



Obrázek 5.2: Prepojka USB 3.0 typ A – micro B.

na obrázku 5.2. Komunikácia s počítačom však nebola úspešná. Pravdepodobným dôvodom, prečo komunikácia po USB 3.0 nefunguje môže byť chyba pri návrhu dosky. USB 3.0 pracuje na vysokej frekvencii a pravdepodobné nesplnenie elektrických špecifikácii má za následok reset mikrokontroléra pri pokuse o nadviazanie komunikácie. Ďalším dôvodom tohto chovania môže byť, že na sonde nie je integrovaná finálna verzia čipu, ale engeneering sample. Môžu obsahovať neopravené chyby ktoré boli vo finálnej verzii čipu opravené.

Kapitola 6

Možnosti rozšírenia práce

Splnením riešenia mojej práce sa naskytli ďalšie možnosti, ako pokračovať vo vývoji.

- Pakety prichádzajúce z FPGA by mohli byť spracované priamo v Cypress mikrokontroléri. FX3 má dostatočný výkon na to, aby mohol previesť konverziu z INI3 na PCAP formát. Prípadne by sa spracovanie týkalo len odrezania nulových bajtov. Otázka je ako by spracovanie dát v FX3 ovplyvnilo priepustnosť.
- Mikrokontrolér FX3 disponuje USB 2.0 OTG radičom. Znamená to, že môže zohrávať aj rolu USB hosta. Ak by bol v jeho firmwari implementovaný file systém, mohol by prijaté dáta ukladať priamo na disk bez použitia počítača. Táto alternatíva je obmedzená USB 2.0 priepustnosťou. Zároveň je tu menšia kontrola nad tým, ktoré dáta sa stihnú zapísať.
- Pamäť I^2C určená pre uchovanie firmvéru FX3 je na na μSonde špatne zapojená. Nedá sa preto naprogramovať. Riešením by bolo napísať ovládač pre automatické nahratie firmwaru pri pripojení zariadenia k počítaču.
- Pre konfiguráciu μSondy sa používa rozhranie UART, ktoré je vyvedené na samostatný micro-USB port. Aby nemusel byť k počítaču pripojený ďalší USB kábel, šlo by vyviesť konfiguračný UART z FPGA na UART rozhranie FX3. Prípadne implementovať túto komunikáciu pomocou GPIF II rozhrania. Do FX3 firmwaru by bolo pridané nové rozhranie s endpointami pre sériovú komunikáciu. FX3 by počítaču sa javil ako 2 zariadenia.
- Sieťové analyzátory nedokážu otvoriť príliš veľký PCAP súbor. Riešením by bolo rozdeľovať po určitých časových intervaloch zapisovaný prúd zachytených paketov, aby boli sieťové dáta uložené po menších častiach.

Kapitola 7

Záver

Vo svojej práci som riešil zabezpečenie prenosu dát z FPGA čipu USONDY do PC pomocou USB rozhrania. USB rozhranie je na USONDE zabezpečuje mikrokontrolér Cypress EZ-USB FX3.

Počas mojej práce boli implementované 3 časti. Prvou je komponenta riadiaca prenosy z FPGA do mikrokontroléra FX3. Ďalšou časťou je upravený firmvér pre mikrokontrolér FX3. Poslednou časťou práce bola implementovaná aplikácia pre operačný systém Linux ktorá prijíma dáta z USB rozhrania USONDY a ukladá ich na disk vo formáte PCAP. Aby bola zabezpečená čo najvyššia priepustnosť, aplikácia pracuje v dvoch vláknach.

Komunikáciu jednotlivých častí sa podarilo zabezpečiť. USONDA vie exportovať zaznamenané dáta cez USB do počítača. Komunikácia však prebieha len cez USB 2.0 rozhranie. Maximálna priepustnosť dosahuje hodnotu 344 Mbit/s. Táto hodnota sa približuje teoretickému maximu USB 2.0 rozhrania. Komunikáciu cez rozhranie USB 3.0 sa nepodarilo nadviazať, pravdepodobne kvôli nedodržaniu elektrických špecifikácií pri návrhu USONDY. Nebolo preto možné zmerať skutočné maximum priepustnosti, ktorú by implementovaný systém mohol cez rozhranie USB 3.0 dosiahnuť.

Literatura

[1] File Formats Manual: libpcap savefile format. 2013-07-29 [cit. 2015-07-25], http://www.tcpdump.org/manpages/pcap-savefile.5.txt.

[2] Wiki projektu SEC6NET: Firmware uSonda. 2014-06-04 [cit. 2015-07-25], https://merlin.fit.vutbr.cz/wiki-sec6net/index.php/Firmware_uSonda#Varianta_.C4.8D.6.

[3] Wiki projektu SEC6NET: INI3 protokol. 2014-11-24 [cit. 2015-07-25], https://merlin.fit.vutbr.cz/wiki-sec6net/index.php/INI3_Protocol#INI3_Payload_v1.

[4] SVN projektu SEC6NET. 2014-11-24 [cit. 2015-07-26], https://merlin.fit.vutbr.cz/svn/sec6net/usonda/propagation/presentations/2014/demo_11092014_poster.odt.

[5] Openclipart: zdroj vektorových obrázkov. [cit. 2015-07-25], http://openclipart.org/.

[6] SEC6NET: projektová dokumentácia. [cit. 2015-07-25], https://merlin.fit.vutbr.cz/wiki-sec6net/images/6/69/Text_projektu.pdf.

[7] Axelson, J.: *USB complete: everything you need to develop custom USB peripherals.* New York: Lakeview Research, třetí vydání, 2005, ISBN 978-1-931448-02-4, 572 s.

[8] Compaq, Hewlett-Packard Comp., Intel Corp., Lucent, Microsoft Corp., NEC Corp., Philips: *Universal Serial Bus Specification Revision 2.0.* 2000-04-27 [cit. 2014-02-04], http://www.usb.org/developers/docs/usb20_docs/usb_20_012314.zip.

[9] Compaq, Intel Corp., Microsoft Corp., NEC Corp.: *Universal Serial Bus Specification Revision 1.1.* 1998-09-23 [cit. 2014-02-04], http://mprolab.teipir.gr/vivlio80X86/usb11.pdf.

[10] Cypress Semiconductor: *FX3 Programmers Manual.* 2013-12-20 [cit. 2014-02-04], http://www.cypress.com/?docID=47220.

[11] Hewlett-Packard Comp., Intel Corp., LSI Corp., Microsoft Corp., Renesas Corp., STMicroelectronics, Texas Instruments: *Universal Serial Bus Power Delivery Specification Revision 2.0, Version 1.1.* 2015-05-07 [cit. 2015-06-31], http://www.usb.org/developers/docs/usb_31_060115.zip.

[12] Hewlett-Packard Comp., Intel Corp., Microsoft Corp., NEC Corp., ST-Ericsson, Texas Instruments: *Universal Serial Bus 3.0 Specification.* 2011-06-06 [cit. 2014-02-04], http://www.usb.org/developers/docs/usb_31_012314.zip.

[13] Hewlett-Packard Comp., Intel Corp., Microsoft Corp., Renesas Corp., ST-Ericsson, Texas Instruments: *Universal Serial Bus 3.1 Specification*. 2013-07-26 [cit. 2014-02-04], `http://www.usb.org/developers/docs/usb_31_012314.zip`.

[14] Hewlett-Packard Comp., Intel Corp., Microsoft Corp., Renesas Corp., STMicroelectronics, Texas Instruments: *Universal Serial Bus Type-C Cable and Connector Specification*. 2015-04-03 [cit. 2015-06-31], `http://www.usb.org/developers/docs/usb_31_060115.zip`.

[15] Hyde, J.: *USB design by example*. Boston: Intel Press, první vydání, 2001, ISBN 0-9702846-5-9, 510 s.

[16] Korček, P.: uG4-150 embedded platform for wire-speed network packet processing. FIT BUT Technical report FIT-TR-2013-03, Faculty of Information Technology, Brno University of Technology, 2013-07-18 [cit. 2015-07-25], `http://www.fit.vutbr.cz/~ikorcek/pubs.php?file=%2Fpub%2F10402%2Ftr.pdf&id=10402`.

[17] Kotásek, Z.: Študijné materiály k predmetu IPZ. 2013 [cit. 2014-02-04], `http://www.fit.vutbr.cz/study/courses/IPZ/public/texty/usb/usb2013.pdf`.

[18] Kořenek, J.: Študijné materiály k predmetu NAV. 2013 [cit. 2014-02-04], `https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/NAV-IT/lectures`.

[19] Kořenek, J.; Korček, P.; Kaštil, J.: Sondy pro monitorování provozu. FIT VUT Technický report FIT-TR-2011-009, Fakulta informačních technologií, Vysoké učení technické v Brně, 2011-12-09 [cit. 2015-07-25], `http://www.fit.vutbr.cz/research/view_pub.php?file=%2Fpub%2F9817%2Ftr.pdf&id=9817`.

[20] Mandau, M.: Technický lexikon: USB 3.1 přinese ještě větší rychlost. *Chip: magazín informačních technologií*, , č. 2, 2014: s. 112–113, ISSN 1210-0684.