VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

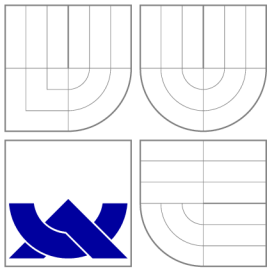
MIKROSONDA NA PLATFORMĚ XILINX ZYNQ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

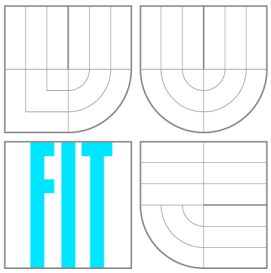
AUTOR PRÁCE
AUTHOR

TOMÁŠ FUKAČ

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

MIKROSONDA NA PLATFORMĚ XILINX ZYNQ

MICROPROBE ON THE XILINX ZYNQ PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ FUKAČ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN VIKTORIN

BRNO 2014

Abstrakt

Tato práce se zabývá portováním firmwaru mikrosondy na platformu Xilinx Zynq. Na základě studia tohoto čipu a desky ZE7000, která je tímto čipem osazena, byl vytvořen návrh firmwaru, který pro komunikaci mezi hardwarem FPGA a softwarem využívá RSoC Framework, který poskytuje podporu pro propojení aplikací běžících na procesorech (ARM Cortex-A9 MPCore) a součástmi, které se nachází v FPGA

Abstract

This work describes portation of microprobe firmware to Xilinx Zynq platform. Based on the study of the target platform of board ZE7000 was created a draft of firmware. RSoC Framework provides interconnections between programmable logic (FPGA) and operating system. With this framework we can find an abstract and universal way to develop applications that are accelerated in the FPGA and running in Linux operating system.

Klíčová slova

FPGA, Zynq, Mikrosonda, NetModule ZE7000, RSoC Framework, Buildroot

Keywords

FPGA, Zynq, Microprobe, NetModule ZE7000, RSoC Framework, Buildroot

Citace

Tomáš Fukač: Mikrosonda na platformě Xilinx Zynq, bakalářská práce, Brno, FIT VUT v Brně, 2014

Mikrosonda na platformě Xilinx Zynq

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Viktorina. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Fukač
19. května 2014

Poděkování

Rád bych poděkoval Ing. Janu Viktorinovi za cenné rady, věcné připomínky a vstřícnost při konzultacích. Jsem rád, že byl jako vedoucí mé bakalářské práce důsledný a pečlivý. Tato práce vnikla za podpory projektu *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace.*

Obsah

Kapitola 1

Úvod

Vestavěné systémy patří k nejrozšířenější variantě počítačových systémů. Nachází se v automobilech, mobilních telefonech, televizorech, pračkách, myčkách nádobí, herních konzolích, tiskárnách, modemech a v mnoha dalších zařízeních. Lidé si při práci s nimi většinou ani neuvědomí, že pracují s počítačovým systémem.

Některé vestavěné systémy jsou určeny pro časově náročné aplikace. Do této oblasti spadá také problematika akcelerace a zpracování síťového provozu. Pro zpracování Ethernetových rámců na plné rychlosti linky 1 Gbps je vyvíjena platforma μG4-150. S využitím této platformy je možné sbírat data pro systémy zajišťující monitorování a bezpečnost počítačové sítě. Základem je technologie FPGA, která umožňuje měnit funkci platformy v závislosti na požadavcích konkrétní aplikace. Jednou z konkrétních aplikací je Mikrosonda.

Mikrosonda je určena pro nasazení u menších poskytovatelů internetového připojení (ISP) nebo přímo do infrastruktury mezi ISP a koncového uživatele, kde monitoruje veškerou komunikaci na síti a sbírá požadovaná data. Data jsou následně posílána do sběrného zařízení pomocí zabezpečeného kanálu.

Pokud data není možné bezpečně přenést do sběrného zařízení po zabezpečeném kanálu, lze Mikrosondu provozovat v offline režimu. V tomto režimu může Mikrosonda prostřednictvím USB portu ukládat data na externí médium (např. USB disk). Dalším rozšířením Mikrosondy je pokročilé zpracovávání vybraného síťového provozu procesorem. Pro tuto možnost je však aktuálně využívaná platforma nedostačující. V logice FPGA je syntetizován procesor MicroBlaze jehož výkon je omezen možnostmi samotného FPGA. Frekvence tohoto procesoru ve firmwaru Mikrosondy je pouze 100 MHz.

Problém s výkonem procesoru lze řešit použitím FPGA, které na jednom čipu obsahuje kromě samotné programovatelné logiky i procesor. Jedním z nich je Xilinx Zynq, další například Altera Cyclone V.

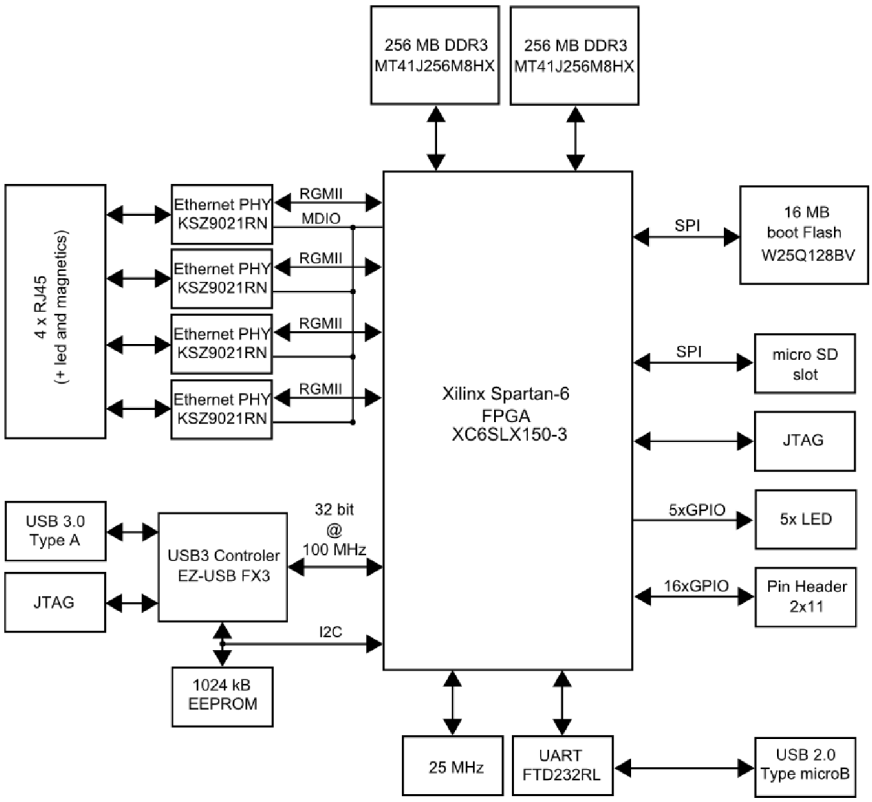
Kapitola 2

Mikrosonda

Mikrosonda je vestavný systém, který je součástí systému pro zákonné odposlechy [1]. Je určena pro monitorování komunikace na síťových linkách o rychlosti 1 Gbps. Mikrosonda je navržena nad hardwarovou platformou μG4-150 [13].

2.1 Hardwarová platforma μG4-150

Platforma μG4-150 obsahuje FPGA Xilinx Spatran-6 s nevyšším dostupným výkonem (speed grade) a počtem logických členů. V tomto FPGA je implementována veškerá funkcionalita zpracovávání síťového provozu. Na desce se nachází paměť flash o kapacitě 16 MB,



Obrázek 2.1: Koncept u4G-150. [13, s. 1]

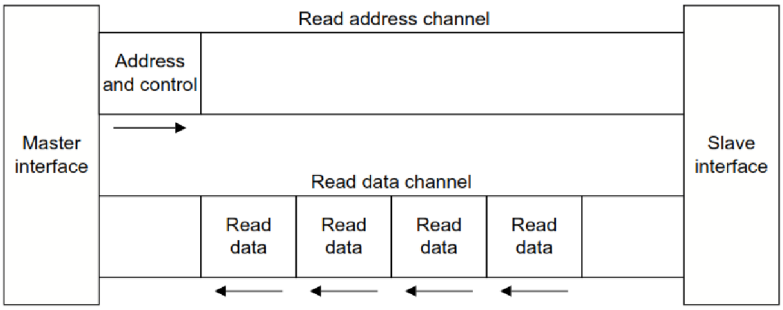
pomocí které je programováno FPGA. Platforma dále obsahuje 4 porty RJ-45, které jsou přímo vedeny přes Ethernetové phytery (Physical Layer Transceiver) do FPGA, dvě DDR3 paměti (každá o kapacitě 256 MB), konektor mikro USB typu B určený pro sériovou komunikaci (UART), USB 3.0 konektor a čip pro připojení periferií (USB disk), konektor JTAG určený pro ladění a programování paměti flash, slot paměťové karty typu SD(HC/HX) a několik dalších nezbytných komponent (krystaly, zdroje apod.). [13, s. 2]

Komunikace s okolím je zprostředkována především prostřednictvím uvedených Ethernetových portů. Dva z nich jsou určeny pro samotné monitorování linky. Veškerá komunikace na těchto portech je zpracována v FPGA. Další port je určen pro export požadovaných dat do sběrného zařízení. Poslední port je určen pro obecné použití. [13, s. 10]

Pro export dat je možno také využít USB 3.0 konektor, pomocí kterého je možné ukládat data na externí paměťové médium (USB flash disk apod.). Tuto variantu lze zvolit například, pokud není možné data bezpečně dopravit do sběrného zařízení (offline režim Mikrosondy). [13, s. 14]

2.2 AMBA AXI4

AMBA AXI4 [4] je rodina sběrnic point-to-point (přímé spojení mezi dvěma uzly) vyvinutá společností ARM. Sběrnice jsou určeny pro komunikaci ve firmwatech s nízkou latencí [4, s. 22] a jsou použity pro přenos dat v komponentách firmwaru aplikace Mikrosonda a v komponentách poskytovaných společností Xilinx.



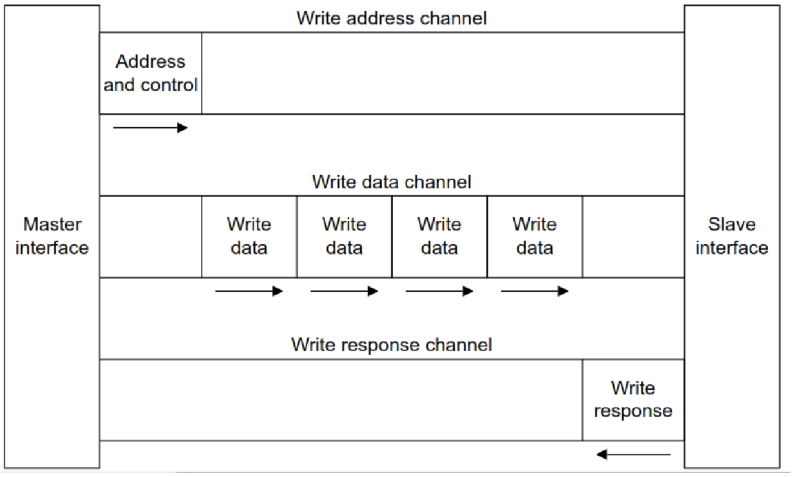
Obrázek 2.2: Čtení dat sběrnicí AXI4. [4, s. 24]

AXI4 sběrnice obsahuje pět oddělených kanálů: adresa určující cíl zápisu, zapisovaná data, adresa specifikující zdroj čtení, odpověď zápisu (potvrzení či hlášení chyb). Čtení či zápis dat probíhá po tzv. shlucích (burst). Master rozhraní specifikuje adresu pro čtení/-zápis. Jedná-li se o čtení, slave začne posílat požadovaná data (viz obr. 2.2). Při zápisu master začne posílat data a slave informuje o úspěchu či neúspěch zápisu (viz obr. 2.3).

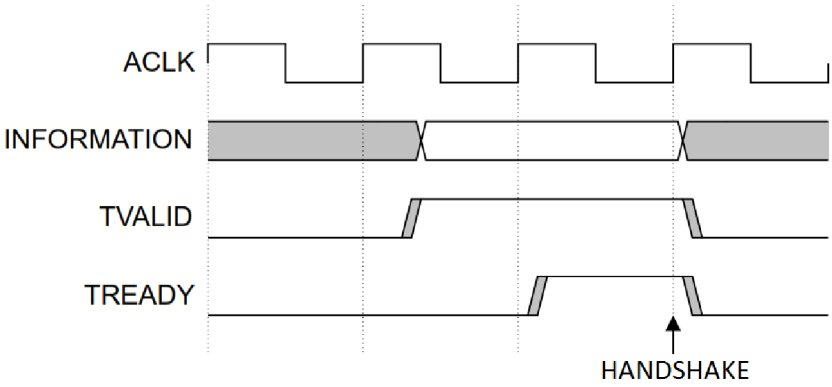
Všechny kanály používají potvrzování handshake pomocí signálů TREADY a TVALID. Výměna dat je úspěšná pokud jsou oba signály aktivní (viz obrázek 2.4).

2.2.1 Sběrnice AXI4-Lite

AXI4-Lite [4, s. 125–127] je zjednodušená verze sběrnice AXI4 a používá se tam, kde není zapotřebí plná funkcionalita AXI4. Je určena pro čtení a zápis pouze jednoho datového slova (shluk o velikosti 1) na požadované adrese (např. práce s registrem, paměťovou položkou).



Obrázek 2.3: Zápis dat sběrnicí AXI4. [4, s. 24]



Obrázek 2.4: Handshake na kanálech sběrnice AXI4. [3, s. 23]

2.2.2 Sběrnice AXI4-Stream

AXI4-Stream [3] je jednosměrná (simplex) sběrnice. Je určená pro transport dat z master zařízení do slave zařízení. Slave zařízení nemůže žádným způsobem komunikovat s master zařízením, může pouze zastavit proud dat prostřednictvím TREADY signálu (viz handshake v kapitole 2.2).

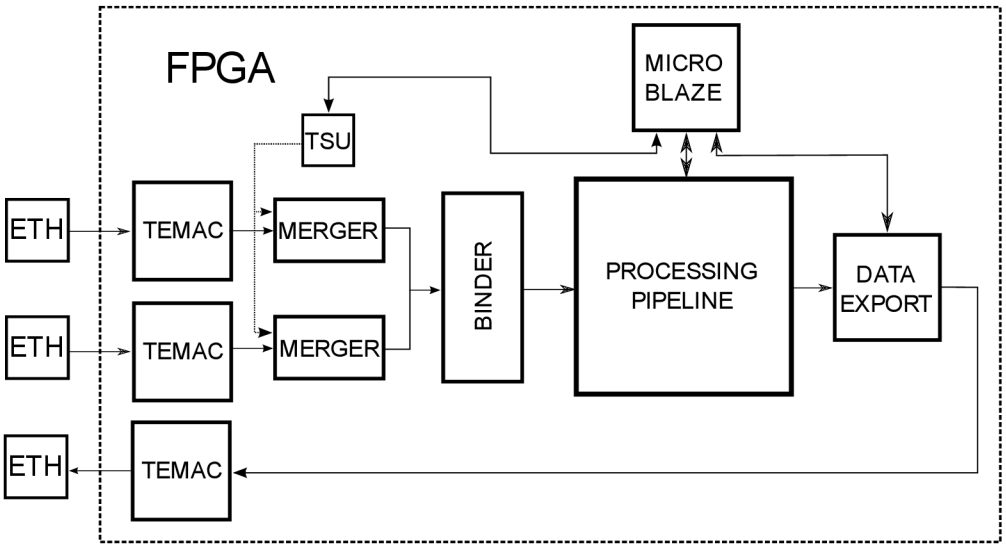
Tato sběrnice je ve firmwaru Mikrosondy využita pro předávání dat v řetězené lince.

2.3 Aplikace Mikrosonda

Filtrování provozu je implementováno v FPGA. Funkcionalita FPGA je určena firmwarem, který je rozčleněn na vstupní část, procesní část a výstupní část.

2.3.1 Vstupní část

Úkolem vstupní části je především přijetí síťové komunikace ze dvou Ethernetových rozhraní (jedno rozhraní pro komunikaci směřující od koncového zařízení a druhé pro komunikaci v opačném směru stejné linky) a spojení datových toků z těchto rozhraní v jeden. Před



Obrázek 2.5: Zjednodušené schéma zapojení komponent ve firmwaru. [14]

samotným spojením je každému datovému rámci předřazena hlavička (INI3), která doplní dodatečné informace potřebné pro sběrné zařízení (časová značka, délka datového rámce, číslo vstupního portu). Tuto činnost provádí MERGER. Komponenty TEMAC zprostředkovávají komunikaci po Ethernetu prostřednictvím phyterů.

Pro spojení dvou vstupních datových toků je využita komponenta BINDER. Úkolem komponenty je vybírat vstup (algoritmem round robin), ze kterého je na výstup vysílán jeden celý rámec. Spojení datových toků probíhá bez jakýchkoli ztrát na datech či na rychlosti datových toků. Filtrování je možno provádět na plné rychlosti 1 Gbps obou vstupních linek, na výstupu komponenty BINDER v tom případě dosahuje firmware propustnosti až 2 Gbps. [13, 14]

2.3.2 Procesní část

Úkolem procesní části je klasifikace a filtrování vstupního datového toku podle definovaných pravidel. Každé pravidlo obsahuje položky zdrojová a cílová IP adresa (verze 4 i 6), zdrojový a cílový port, číslo protokolu. Pravidla lze softwarově přidávat a odebírat.

Komponenta FILTER je dimenzována na propustnost až 40 Gbps a ve firmwaru Mikrosondy výkonově dostačuje (při frekvenci 100 MHz).

Na výstup komponenty jsou předány pakety, které vyhoví některému z definovaných pravidel, spolu s čísly pravidel, podle kterých k filtraci došlo. [14]

2.3.3 Výstupní část

Tato část firmwaru nejdříve doplní do INI3 hlavičky každého datového rámce, který vyhověl některému pravidlu v procesní části firmwaru, informaci o použitém filtrovacím pravidle. Takto připravené datové rámce jsou pomocí komponenty DATA_EXPORT odesílány po síti protokoly UDP/IP (verze 4 i 6) do sběrného zařízení k dalšímu zpracování systémem pro zákonné odposlechy. Veškeré informace vkládané do hlaviček (MAC adresy, IP adresy, apod.) lze softwarově nastavovat.

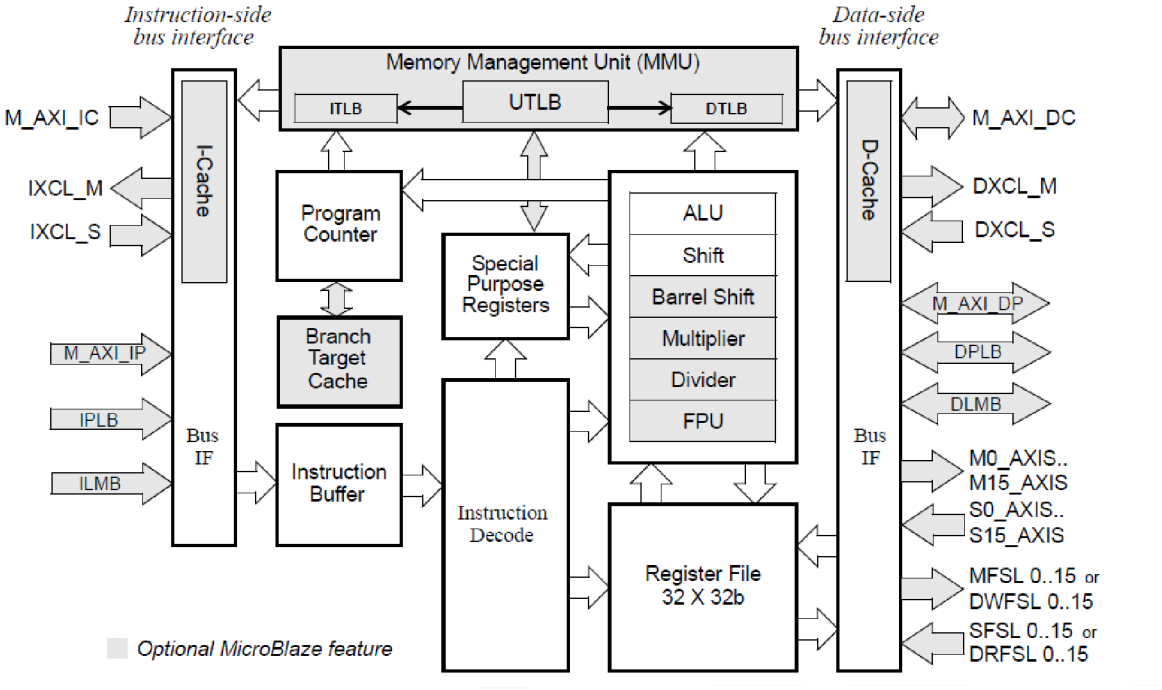
DATA_EXPORT řídí odesílání dat prostřednictvím komponenty TEMAC. [14]

2.4 MicroBlaze

MicroBlaze je soft procesor a lze ho tedy implementovat v logice FPGA firmy Xilinx. Toto procesorové jádro je použito v aplikaci Mikrosonda pro konfiguraci hardwarových komponent. [29]

2.4.1 Architektura

MicroBlaze je široce konfigurovatelný 32-bit procesor harvardské architektury (fyzicky oddělená paměť programu a dat) s redukovanou instrukční sadou (RISC).



Obrázek 2.6: Architektura procesoru MicroBlaze. [26, s. 9]

Jelikož se jedná o komerční produkt, zdrojové kódy tohoto procesoru nejsou volně dostupné. Procesor je poskytován prostřednictvím integrovaného vývojového nástroje Xilinx EDK (Embedded Development Kit). V tomto nástroji je možno provádět veškerou konfiguraci procesoru.

Procesor MicroBlaze ve výchozí konfiguraci nabízí třicet dva 32-bitových registrů, aritmeticko-logickou jednotku (ALU), posuvný registr, sadu speciálních registrů (např. Program Counter, Machine Status Register) [26, s. 25-49] atd. Datovou i instrukční paměť lze adresovat 32 bity, je tedy možno mít až 4 Gb paměti určené pro program a stejnou velikost paměti určené pro data. [26, s. 54]

Mezi konfigurovatelné součásti tohoto procesoru (na obrázku 2.6 vyznačeno šedě) patří především Barrel Shifter (optimalizace posuvu), Multiplier/Divider (optimalizace násobení/dělení), jednotka pro práci s čísly s pohyblivou řádovou čárkou (FPU), Memory Management Unit (MMU), hardwarový debugger. [29]

Pro práci s daty lze volit mezi Big Endian a Little Endian. Pro zvýšení výkonu lze využít datovou a instrukční cache volitelné velikosti. Procesor dále umožňuje volbu hloubky pipeline. [26, s. 13]

2.4.2 Připojení k periferiím

Interakce s periferiemi je zajišťována prostřednictvím sběrnic: Processor Local Bus (PLB), Local Memory Bus (LMB), Fast Simplex Link (FSL), Xilinx CacheLink (XCL), Debug a Trace Interface, AXI4 (Advanced eXtensible Interface).

LMB je synchronní sběrnice primárně používaná pro přístup k block RAM v FPGA. Aby byl zajištěn přístup do block RAM v jednom hodinovém cyklu, sběrnice obsahuje minimum řídicích signálů a pro komunikaci se využívá jednoduchý protokol. FSL je analogie sběrnice AXI4-Stream. Debug a Trace Interface slouží pro ladění a je využíván rozhraním JTAG. Firma Xilinx v procesoru MicroBlaze od verze 8.0 zavádí novou sběrnici AXI4. Procesor obsahuje rozhraní AXI4-Lite, které je mapované do paměti. Prostřednictvím tohoto rozhraní je možno přistupovat k adresovému prostoru komponent. Pro přenos dat procesor obsahuje 16 master a 16 slave rozhraní AXI4-Stream. PLB je vysokorychlostní sběrnice, která je určena pro komunikaci mezi procesorem a ostatními komponentami FPGA. Konfigurace procesoru MicroBlaze umožňuje komunikovat prostřednictvím jen jedné ze sběrnic AXI4 a PLB. PLB je tak často nahrazována AXI4 a téměř se již nepoužívá. [26, s. 94–129]

2.4.3 Výkon procesoru

Nejpoužívanějšími jednotkami pro vyjádření výkonu procesoru se používá MIPS (Million Instruction Per Second) a DMIPS (Dhrystone MIPS).

MicroBlaze Processor v8.40.b			
Device Family	Performance Optimized MicroBlazewith branch optimizations (5-stage pipeline)	Performance Optimized MicroBlaze (5-stage pipeline)	Area Optimized MicroBlaze (3-state pipeline)
Zynq-7000 SoC	228 DMIPs	259 DMIPs	196 DMIPs
Virtex-7 FPGA	293 DMIPs	393 DMIPs	264 DMIPs
Kintex-7 FPGA	317 DMIPs	408 DMIPs	264 DMIPs
Virtex-6 FPGA	306 DMIPs	384 DMIPs	246 DMIPs
Spartan-6 FPGA	166 DMIPs	209 DMIPs	152 DMIPs
	1.38 DMIPs/ MHz	1.30 DMIPs/ MHz	1.03 DMIPS/ MHz

Tabulka 2.1: Výkon procesoru pro nejvýkonnější konfigurace a různá FPGA. [29]

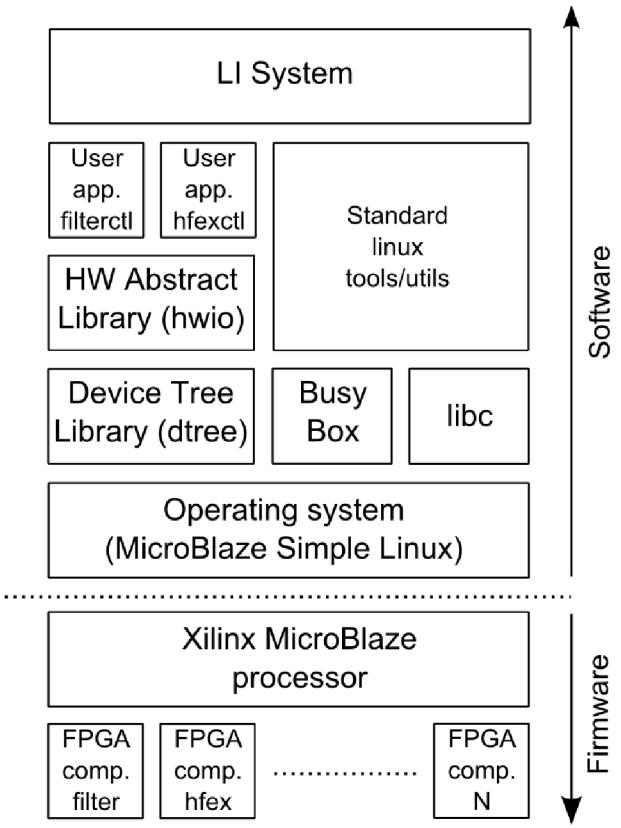
Jednotka MIPS vyjadřuje počet milionů instrukcí, které dokáže procesor vykonat za sekundu. Každá instrukce trvá jiný počet taktů, zvolením jiných instrukcí je možno dosáhnout pro stejný procesor zcela jiných hodnot. Z tohoto důvodu se jednotka MIPS nehodí pro srovnávání.

Tento problém odstraňuje jednotka DMIPS, která udává počet milionů instrukcí provedených za sekundu na konkrétním programu. Tento test je zaměřen pouze na celočíselné operace. Odvozená jednotka DMIPS/MHz zahrnuje i informaci o frekvenci procesoru a udává počet DMIPS procesoru taktovaného na 1 MHz.

Výkon procesoru MicroBlaze se odvíjí od zvolených parametrů a vybraného FPGA. Pro FPGA Spartan-6, který se nachází na platformě μG4-150, a nejvýkonnější konfiguraci dosahuje procesor nejvýše frekvence 160 Mhz a 209 DMIPS (viz tabulka 2.1). [29]

2.5 Software Mikrosondy

Softwarový systém Mikrosondy je založen na operačním systému linuxového typu (konkrétně jde o linuxovou distribuci postavenou nad systémem Buildroot) s linuxovým jádrem upraveným firmou Xilinx. Dle aktuální konfigurace se v operačním systému nachází standardní linuxové nástroje (`cat`, `grep`, `ls` apod.) poskytované programem BusyBox a standartní knihovna `libc`.



Obrázek 2.7: Softwarové vrstvy Mikrosondy. [12]

2.5.1 Flat Device Tree

Flat Device Tree (FDT) je stromová datová struktura určená pro popis hardwaru v systému. Uchovává v sobě informace o platformě, které jsou předány operačnímu systému:

- počet a typ procesorů,
- bázové adresy a velikosti paměťových prostorů (např. RAM),
- sběrnice a bridge,
- dostupná periferních zařízení,
- přerušení.

Jádro operačního systému Linux je schopno v současné době číst informace o stromu zařízení na architekturách ARM, x86, MicroBlaze, PowerPC a SPARC. Obraz FDT může být staticky připojen k jádru systému, nebo předán jádru systému při jeho bootování. [11]

2.5.2 Knihovna HWIO

Z pohledu aplikací je důležitá knihovna Device Tree Library (`dtree`), která v uživatelském prostoru zpřístupňuje informace o komponentách FPGA. Knihovna pracuje se stromem zařízení FDT.

Nad touto knihovnou je pomocí knihovny `hwio` postavena abstraktní vrstva (HW Abstract Library). Knihovna `hwio` využívá knihovnu `dtree` pro vyhledání potřebných informací (ve stromu zařízení FDT), které slouží pro správné mapování adresového prostoru do paměti. S tímto adresovým prostorem je následně možno prostřednictvím knihovny `hwio` pracovat (číst či zapisovat). [23]

Knihovna `hwio` vytváří abstraktní vrstvu, která je nezávislá na hardwaru. Uživatelské aplikace používají shodné rozhraní pro různé platformy (μG4-150, ComboV2[1]).

2.5.3 Aplikace pro konfiguraci komponent

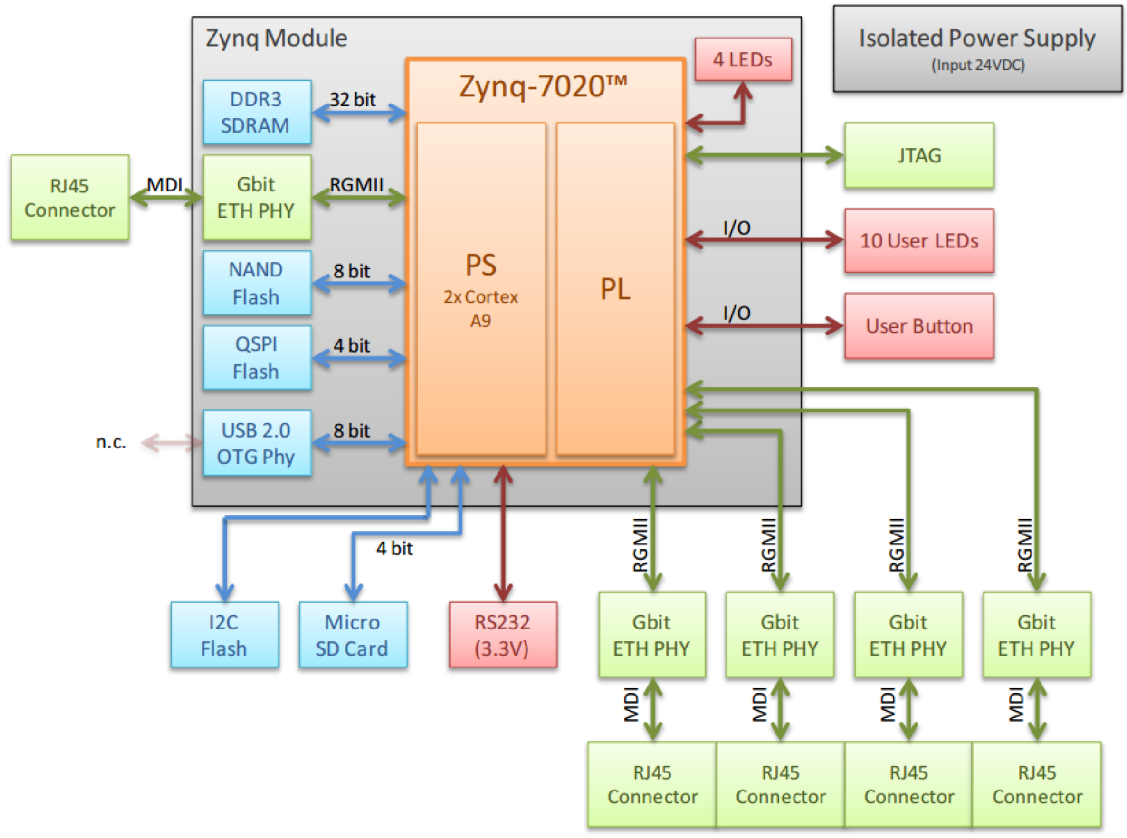
Aplikace pro konfiguraci komponent využívají knihovnu `hwio`. Pro zápis dat do komponent knihovna `hwio` poskytuje funkci `hwio_comp_write()`. Této funkci je předána specifikace komponenty, do které je zapisováno, adresa, na kterou se zapisuje, a data. Knihovna `hwio` najde ve FDT odpovídající komponentu a zapíše do jejího adresového prostoru jedno datové slovo na odpovídající adresu. Čtení dat z komponenty probíhá obdobným způsobem funkcí `hwio_comp_read()`.

[1] https://www.invea.com/cs/produkty-sluzby/fpga-karty/combo-20g

Kapitola 3

Platforma ZE7000

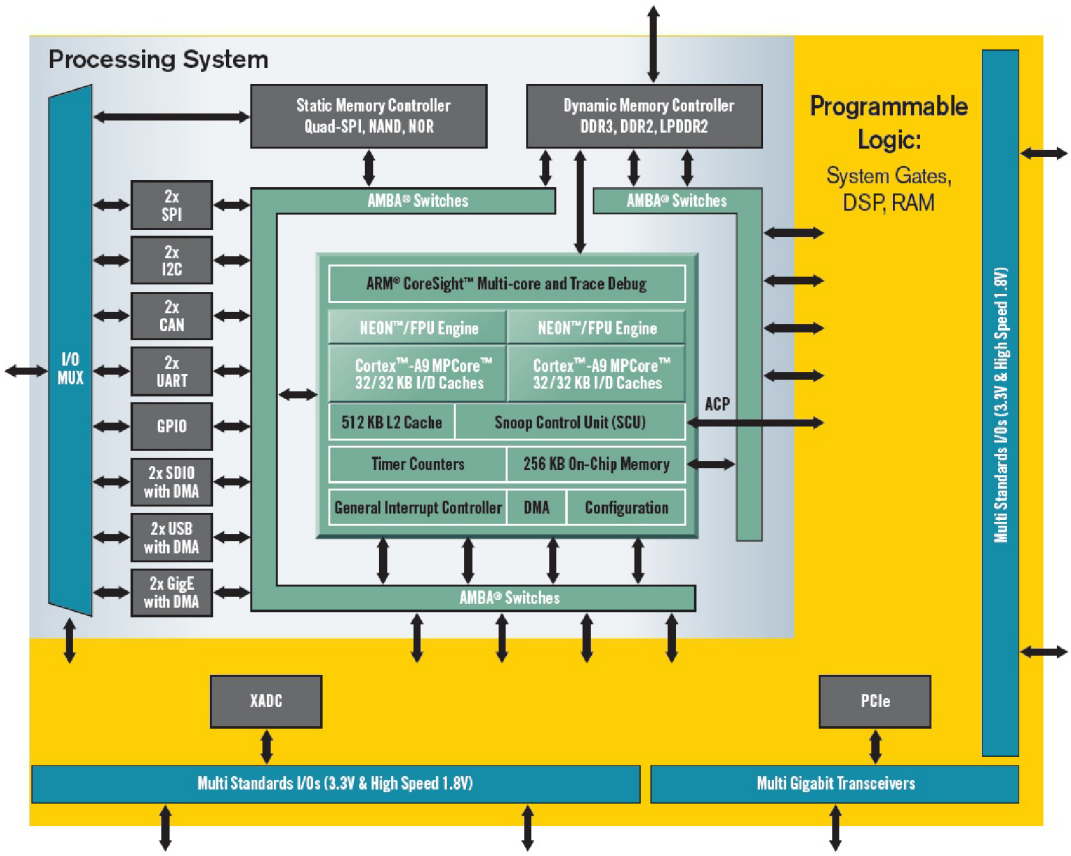
Platforma ZE7000 [15] od firmy NetModule je určena pro náročné síťové aplikace komunikující prostřednictvím rozhraní Ehernet a byla zvolena jako alternativní platforma pro Mikrosondu. Skládá se ze základní desky a samostatného modulu Enclustra Mars ZX3, který obsahuje čip Zynq od firmy Xilinx.



Obrázek 3.1: Schéma platformy ZE7000. [15]

3.1 Xilinx Zynq

Architektura rodiny Xilinx Zynq-7000 je System-on-Chip (SoC), která v sobě integruje dvoujádrový procesor ARM Cortex-A9 MPCore, který je základ procesního systému (PS), a programovatelnou logiku firmy Xilinx (FPGA) do jednoho zařízení, vytvořené 28nm technologií. Rodina Zynq-7000 v sobě spojuje flexibilitu a rozšiřitelnost FPGA s výkonem a použitím, které je často spojováno s ASIC a ASSP. [32]



Obrázek 3.2: Blokové schéma Zynq-7000 AP SoC. [9]

Architektury podobné Xilinx Zynq jsou vyráběny již delší dobu. Příkladem může být například FPGA série Xilinx Virtex-5. Toto FPGA obsahuje volitelný hard procesor spolu s lokální pamětí (block RAM), DSP bloky, AD převodník, atd. Procesor je možno využít pro úkoly, které by bylo obtížné nebo nákladné realizovat v logice FPGA (např. DMA). [21, s. 10] [25]

Zynq se však od tohoto FPGA liší. Procesní systém v podobě hard procesoru je nezbytnou součástí čipu. Zynq je možno vidět jako systém, jehož hlavním prvkem je procesor ARM, který obklopuje programovatelná logika FPGA. [21, s. 10]

3.1.1 Procesní systém

Procesní systém je založen na dvoujádrovém procesoru ARM Cortex-A9 MPCore, který běží na frekvenci 1 GHz. MPCore kromě procesorů obsahuje jednotku Snoop Control (SCU), privátní periferie (watchdogy, časovače) a řadič přerušení. [33, s. 31–37]

3.1.2 ARM Cortex-A9 MPCore

ARM Cortex-A9 MPCore je 32-bitový RISC procesor harvardské architektury a obsahuje šestnáct 32-bitových registrů. Volitelnými jednotkami jsou NEON (provádění instrukcí SIMD) a jednotka pro práci s čísly v plovoucí desetinné čárce.

Jedná se o vícejádrový procesor, který může obsahovat až čtyři procesorová jádra s koherentní vyrovnávací pamětí cache. Velikost cache L1 je 32 kB pro instrukce, 32 kB pro data, velikost cache L2 je 512 kB. [5]

	Maximální frekvence (GHz)	DMIPs/MHz/core
ARM Cortex-A5	1.00	1.6
ARM Cortex-A8	1.00	2.0
ARM Cortex-A9	1.00	2.5
MIPS 24K	1.47	1.4
MIPS 34K	1.45	1.6
MIPS 74K	1,60	2.0
Intel Atom N280	1,67	2.4

Tabulka 3.1: Porovnání výkonů procesorů. [8]

Výkon tohoto procesoru lze vyčíst z tabulky 3.1, který udává počet DMIPS na 1 MHz a na jedno jádro. Výkon jednoho jádra je v porovnání s výkonem procesoru MicroBlaze (udávaný v DMIPS/MHz) asi dvojnásobný (viz zvýrazněná pole v tabulkách 3.1 a 2.1).

Xilinx Zynq obsahuje tato procesorová jádra dvě (pracující na frekvenci přibližně 1 GHz), proto je výkon v porovnání s procesorem MicroBlaze téměř čtyřnásobný.

3.1.3 Jednotka Snoop Control

SCU blok propojuje oba procesory Cortex-A9 s paměťovým podsystémem a obsahuje logiku pro správu koherence dat procesoru mezi vyrovnávací pamětí L1 a L2. Tento blok zodpovídá za arbitraci, komunikaci a přenosy mezi vyrovnávacími pamětmi a systémovou pamětí. [33, s. 90]

3.1.4 Rozhraní paměti

Rozhraní paměti obsahuje dynamický řadič paměti a statické paměťové moduly rozhraní. Dynamický řadič paměti podporuje paměti typu DDR3, DDR3L, DDR2 a LPDDR2. Statické řadiče paměti podporují rozhraní NAND flash, Quad-SPI flash, paralelní sběrnice a paralelní NOR flash.

I/O Periferní zařízení (IOP)

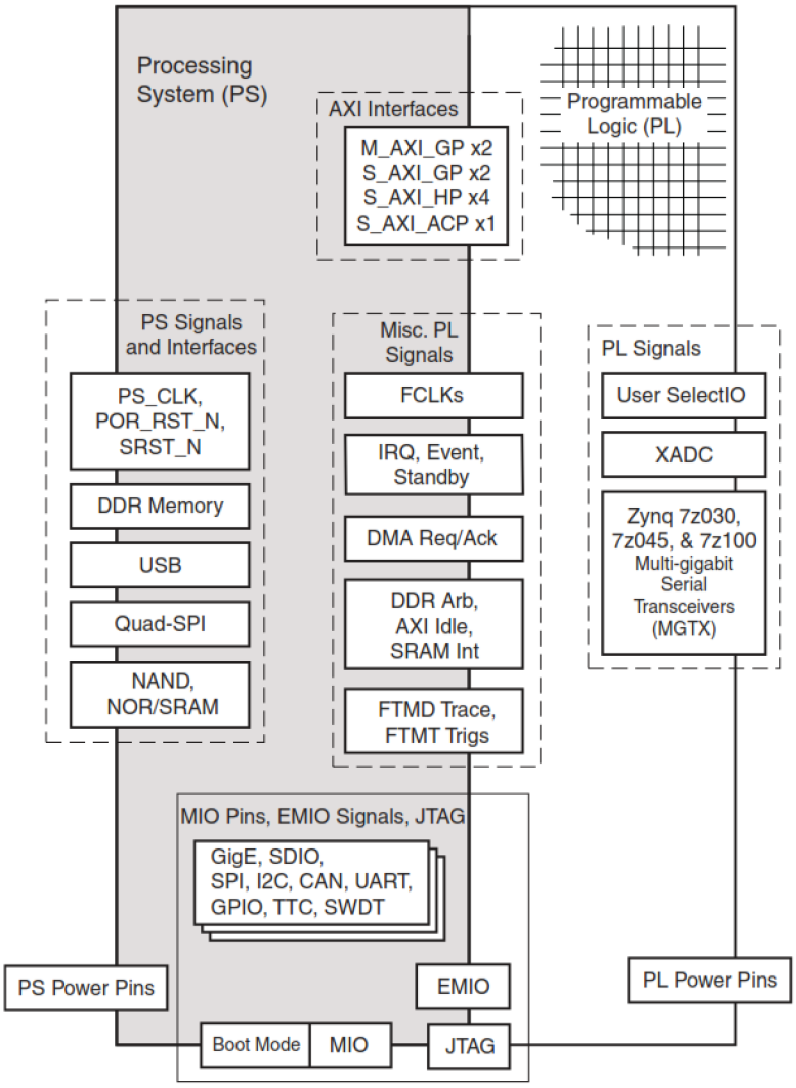
Zynq obsahuje několik rozhraní, pomocí kterých je možno připojit periferní zařízení. Mezi hlavní patří dvě rozhraní tri-mode Ethernet MAC (podporován standard IEEE Std 802.3 a IEEE Std 1588), dvě rozhraní USB 2.0, rozhraní AHB DMA master a AHB slave, dva SD/SDIO 2.0 řadiče s vestavěným DMA, UART, I2C a dvě SPI rozhraní, atd. Pro připojení dalších periferních zařízení je možno využít až 118 GPIO pinů. [33, s. 34–37]

3.1.5 Interconnect

Procesory, rozhraní paměti a IOP jsou vzájemně propojeny s programovatelným polem FPGA prostřednictvím AMBA AXI interconnectu. Propojení je neblokující, podporující současnou přítomnost více master-slave transakcí. [33, s. 39]

3.1.6 Rozhraní procesního systému

Rozhraní procesního systému využívá dedikované piny čipu Xilinx Zynq, které nelze použít pro připojení k FPGA. Mezi ně patří reset, hodinový signál, 32 nebo 16 bitů rozhraní paměti DDR2/DDR3/ DDR3L/LPDDR2 a až 54 dedikovaných pinů MIO (multiuse I/O), kterými jsou multiplexovaně zpřístupněny některé interní I/O periferie. Pokud je 54 pinů MIO nedostačujících, lze toto rozhraní rozšířit pomocí programovatelného pole. Toto rozhraní je označováno jako EMIO (extendable multiplexed I/O). [30, s. 1]



Obrázek 3.3: Rozhraní procesního systému a programovatelné logiky. [33, s. 43]

3.1.7 Programovatelné pole

Programovatelné pole se skládá z bloků CLB (configurable logic Block). Každý blok obsahuje osm šestivstupých look-up tabulek (LUT), ve kterých lze realizovat logickou funkci nebo distribuovanou paměť, šestnáct klopných obvodů (flip-flops) a dvě 4-bitové sčítačky. Programovatelná logika dále obsahuje 60-545 Block RAM každá o kapacitě 36 Kb, 80-900 DSP bloků, dva 12-bitové analogově-digitální převodníky (XADC), rozhraní PCIe a další. [33, s. 37–39]

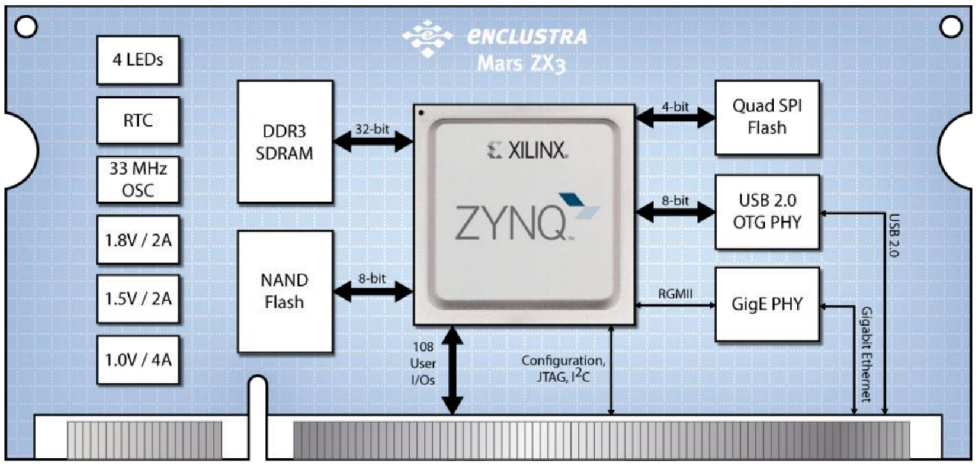
3.1.8 Rozhraní mezi procesním systémem a programovatelným polem

Rozhraní mezi procesním systémem a programovatelným polem obsahuje dvě master a dvě slave 32-bitová rozhraní AXI4, čtyři 64-bit/32-bit konfigurovatelná bufferovaná AXI4 slave rozhraní pro přímý přístup do paměti DDR, signály přerušení, událostí, hodin, resetu atd.

Pro propojení procesorů a funkcionality akcelerované v programovatelném poli slouží rozhraní ACP (Accelerator Coherency Port). Jedná se o 64-bitové AXI4 slave rozhraní, které poskytuje přímé spojení mezi programovatelným polem a jednotkou SCU. [33, s. 39–40]

3.2 Enclustra Mars ZX3

Modul Mars ZX3 je osazen čipem Xilinx Zynq, rychlou pamětí DDR3 SDRAM, NAND a SPI flash, Gigabitový Ethernetový phyter a RTC (Real-time clock).



Obrázek 3.4: Schéma modulu Enclustra Mars ZX3. [17, s. 9]

Hlavním prvkem tohoto modulu je FPGA Xilinx Zynq-7020. Většina pinů (108) tohoto FPGA je připojena ke konektoru typu DDR tohoto modulu, zbylé piny slouží pro připojení DDR3 SDRAM, NAND a SPI flash, USB 2.0 a Ethernetového phyteru. Ve standardní konfiguraci je na modulu k dispozici 512 MB NAND flash, 16 MB SPI flash a 256 nebo 512 MB DDR3 SDRAM. Konfigurace programovatelného pole Zynq je prováděna prostřednictvím obrazu uloženého v SPI flash paměti, nebo prostřednictvím konektoru JTAG, který se připojí k modulu. Modul je vybaven zdrojem hodin (RTC), který pomocí krystalového oscilátoru generuje hodinový signál o frekvenci 33 MHz. Modul je připojen k desce prostřednictvím 200 pinů rozhraní DDR2-SODIMM. [17, s. 6–9]

3.3 Základní deska ZE7000

Základní deska platformy ZE7000 je osazena pěti metalickými RJ-45 porty určenými pro Ethernetové rozhraní a čtyřmi phytery pracujícími na rychlostech až 1 Gbps. Phytery jsou připojeny ke čtyřem rozhraním označených jako ETH2-5. Rozhraní označené jako ETH0 je přímo napojeno na slot zásuvného modulu (viz níže).

Pro uložení a čtení souborů (např. operační systém, konfigurační soubory) lze využít SDHC slot pro paměťové karty. Komunikaci s okolím mohou být zprostředkovány pomocí rozhraní I2C, UART a JTAG (pro ladění a programování FPGA).

Deska dále obsahuje deset uživatelských LED diod, dvě tlačítka a piny GPIO (General-purpose input/output).

Platforma ZE7000 byla vytvořena jako modulární systém. Z tohoto důvodu se na desce nachází DDR slot, do kterého se vkládá rozšiřující modul. V našem případě se jedná o modul Enclustra Mars ZX3. [15, 16]

Kapitola 4

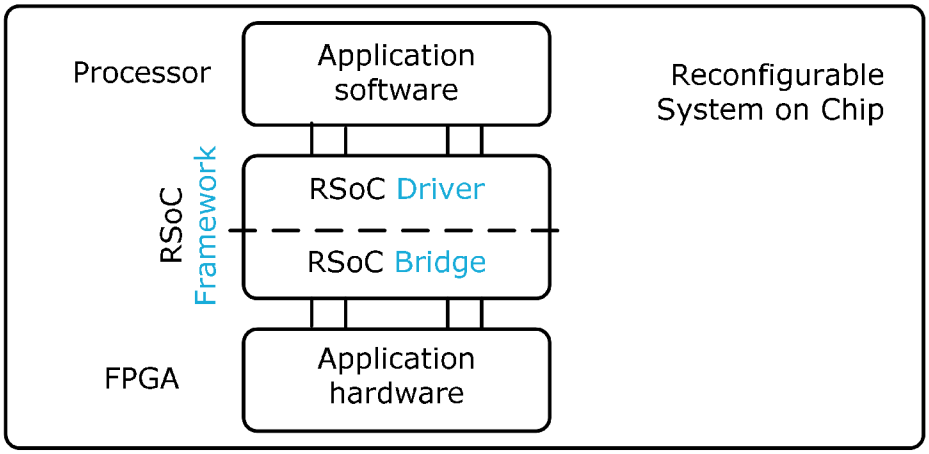
RSoC Framework

RSoC (Reconfigurable System-on-Chip) je platforma s výkonným procesorem (např. ARM Cortex-A9) a programovatelným polem (FPGA). Tyto platformy umožňují HW/SW co-design nebo zrychlení již existujících aplikací. Příkladem jsou Xilinx Zynq All Programmable SoC nebo Altera Cyclone V SoC[1].

RSoC Framework je řešení pro vývoj aplikací na platformách, které obsahují procesor a programovatelné pole. RSoC Framework poskytuje podporu pro propojení aplikací běžících na procesoru s jejich součástmi akcelerovaných v FPGA. Tento Framework lze využít v čipech RSoC nebo v FPGA obsahující soft procesor, jako je například MicroBlaze. [20]

4.1 Architektura

RSoC Framework vytváří vrstvu middleware pro abstrakci hardwaru (rozhraní nezávislé na platformě). Framework se skládá z hardwarové komponenty RSoC Bridge a ovladače RSoC Driver (viz obrázek 4.1).



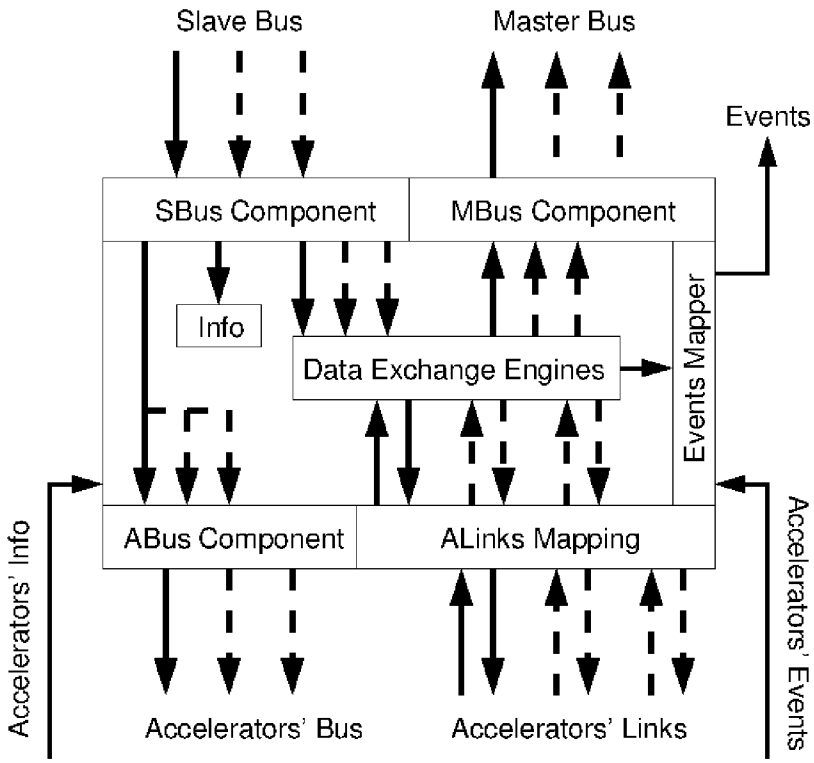
Obrázek 4.1: Schéma RSoC Frameworku. [20]

[1] http://www.altera.com/devices/processor/soc-fpga/cyclone-v-soc/cyclone-v-soc.html

4.1.1 RSoC Bridge

RSoC Bridge je generické IP[2] (Intellectual Property) jádro poskytující hardwarovou vrstvu mezi uživatelskými IP jádry a vybraným operačním systémem. RSoC Bridge obsahuje pro komunikaci několik sběrnicových systémů. Základními sběrnicovými systémy AXI4 jsou:

- **Slave Bus (SBus)** – tato sběrnice je určena pro komunikaci mezi procesorem a akcelerátorem (viz 4.2), jedná se o sběrnici typu AXI4-Full nebo AXI4-Lite. Sběrnice slouží pro přístup k adresovým prostorům, které jsou přístupné pro procesní systém Zynq, na kterém je spuštěn operační systém. Umožňuje také řídit RSoC Bridge a poskytuje přístup k ABus (viz níže).
- **Master Bus (MBus)** – sběrnice typu AXI4-Full je určena pro komunikaci směřující k operačnímu systému. Ve většině případů je MBus datový spoj mezi akcelerátorem a procesním systémem. Každá akcelerátor může být připojen k libovolné sběrnici MBus.
- **Accelerator's Bus (ABus)** – jedná se o sběrnici typu AXI4-Full nebo AXI4-Lite, která je určena pro konfiguraci akcelerátorů. ABus propojuje RSoC Bridge a všechny příslušné akcelerátory.
- **Accelerator's Links (ALinks)** – sběrnice typu AXI-Stream. Slouží pro předávání dat po dvojici simplexních AXI4-Stream sběrnic.



Obrázek 4.2: Sběrnicové systémy RSoC Bridge. [22]

[2] www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ise_c_intellectual_property_cores.htm

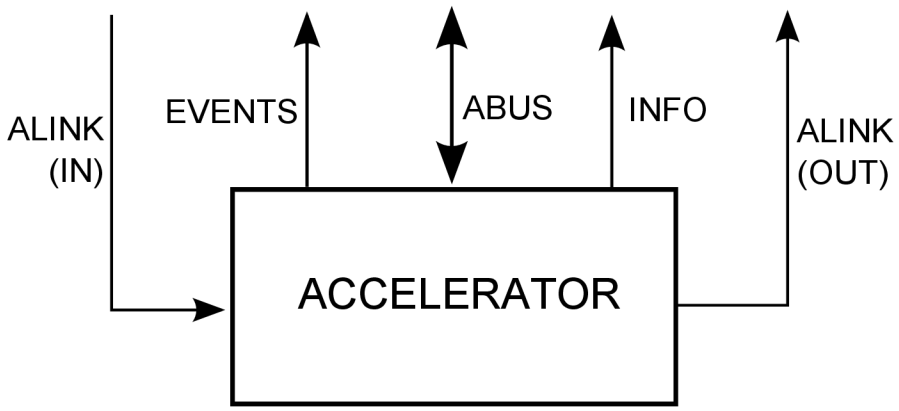
Ke každému akcelerátoru nabízí RSoC Bridge až čtyři (a čtyři skryté) zdroje asynchronních událostí a jeden 32-bitový informační vektor (s informacemi o akcelerátoru), který je vystaven operačnímu systému. [22]

4.1.2 RSoC Driver

RSoC Driver představuje systém ovladačů, pomocí kterých je možno ovládat hardwarovou komponentu RSoC Bridge s připojenými uživatelskými akcelerátory a které vytváří softwarové rozhraní pro přístup k akcelerátorům. Softwarové aplikace využívají tento ovladač pro výměnu dat s akcelerátory nebo k jejich konfiguraci. [24]

4.2 Akcelerátory

RCoC Framework akcelerátor je komponenta, kterou je možno připojit k RSoC Bridge, prostřednictvím kterého lze akcelerátor konfigurovat a pracovat s jeho adresovým prostorem. Pro připojení k RSoC Bridge akcelerátor obsahuje sběrnici ABus (pro přístup k adresovému prostoru), dvě sběrnice ALink (pro příjem a odesílání dat), 32-bitový informační vektor (verze akcelerátoru, atd.) a vektor přerušení.



Obrázek 4.3: Rozhraní akcelerátoru RSoC Framework.

4.3 Komunikace s komponentou

RSoC Driver reprezentuje každý akcelerátor, který je připojen k příslušné komponentě RSoC Bridge, pomocí speciálního souboru (např. `/dev/rsocb_acc0`). Prostřednictvím tohoto souboru je možno přistupovat k adresovému prostoru akcelerátoru pomocí sběrnice ABus a číst či zapisovat data zprostředkovaná rozhraními ALink.

Pro práci s komponentou je třeba otevřít funkcí `open()` odpovídající soubor. Následně pro zpřístupnění adresového prostoru komponenty je třeba mapovat tento soubor do paměti funkcí `mmap()`. Čtení a zápis dat rozhraním ALink je prováděno voláním funkcí `read()` a `write()` nad tímto souborem. Po provedení veškerých činností je funkcí `close()` provedeno zavření souboru, kterému předchází případné odmapování mapovaného souboru funkcí `munmap()`. [24]

Kapitola 5

Mikrosonda na platformě ZE7000

Tato kapitola je věnována portování firmwaru a softwaru aplikace Mikrosonda na platformu ZE7000.

5.1 Komponenta TEMAC v nástroji EDK

Tri-Mode Ethernet Media Access Controller (TEMAC) je konfigurovatelné IP jádro pro realizaci komunikace prostřednictvím Ethernetové sítě dle specifikace IEEE 802.3-2005. Pracuje v režimu 10 Mbps, 100 Mbps nebo 1000 Mbps a podporuje jak half-duplex tak i full-duplex operace. S phyterem komunikuje pomocí rozhraní RGMII nebo GMII. [34]

Pro správnou funkcionalitu rozhraní RGMII a GMII je třeba dodržet shodnou fázi signálů. Pro tyto účely se na výstupu FPGA nachází komponenta IODELAY, které vybranému signálu udělí požadované zpoždění. [28, s. 7]

TEMAC je součástí obálky AXI_ETHERNET, která se nachází ve vývojovém nástroji EDK. Nejnovější verze, která se v tomto vývojovém nástroji nachází, však není podporována pro FPGA Zynq. Při vložení více komponent AXI_ETHERNET do firmwaru, nástroj EDK nepracuje správně s komponentami IODELAY. Pokud se ve firmwaru nachází pouze jedna IODELAY, nástroj ji automaticky připojí na odpovídající fyzický port. Pokud je do firmwaru přidáno více komponent IODELAY, nástroj nespecifikuje, na které fyzické piny se mají komponenty připojit. [27, 35]

Tento problém řeší novější verze komponenty AXI_ETHERNET, které se však nachází ve vývojovém prostředí Vivado.

5.2 Vivado Design Suite

Vývojové prostředí Vivado od firmy Xilinx je nástupcem vývojového nástroje EDK. Nástroj byl implantován zcela od základů, což umožnilo integrovat nové algoritmy syntézy firmwaru (Vivado High-Level Synthesis). Ty umožňují provést syntézu firmwaru až čtyřikrát rychleji. Ten je soustředěn na ploše FPGA, která je až o 20 % menší. Zmenší se tak velikost kritických cest, což umožňuje zvýšit frekvenci. Výkon výsledného firmwaru tak vzroste. [36, s. 11]

Prostředí Vivado nabízí plnou podporu pro FPGA Virtex-7, Kintex-7, Artix-7 a především pro FPGA Zynq-7000, dále pak komplexní prostředí pro hardwarové ladění, které umožňuje sledování všech signálů firmwaru. [37, s. 11]

5.3 Komponenta USONDA_CORE

Pro naportování firmwaru Mikrosondy do prostředí Vivado je třeba do tohoto prostředí importovat jednotlivě všechny komponenty, které se ve firmwaru nachází.

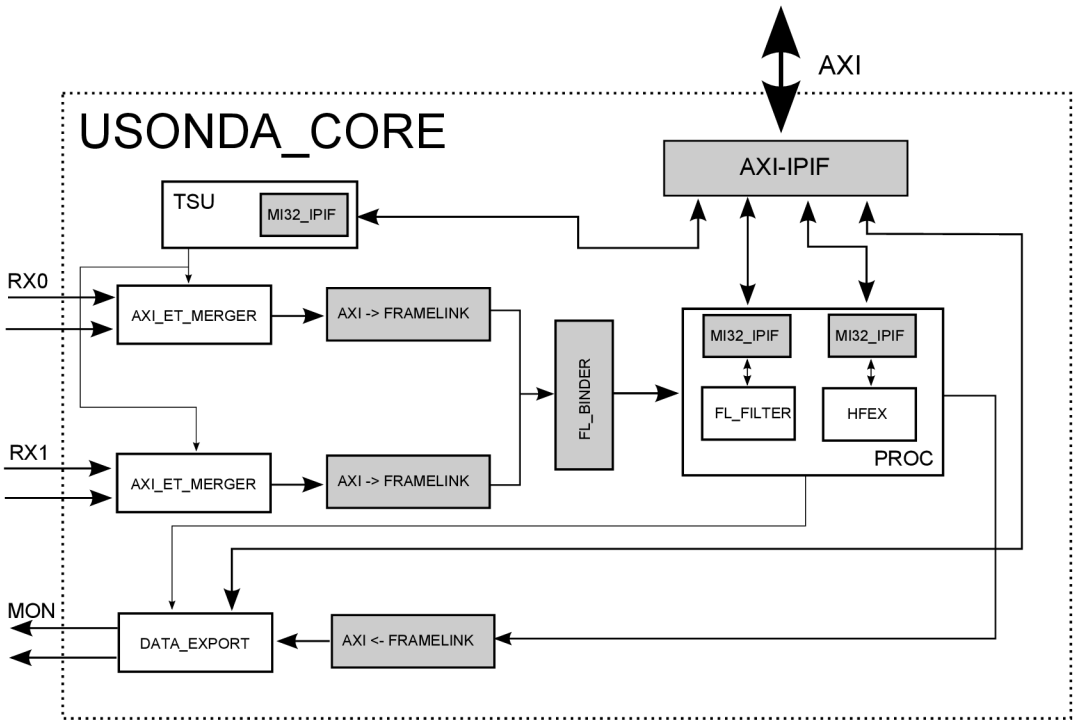
Komponenty, které byly převzaty z projektu Liberouter[1], pro přenos dat využívají sběrnice s protokolem FrameLink [6, s. 12–15]. Pro jejich konfiguraci je využita sběrnice MI32 [6, s. 15–16]. Konfigurace ostatních komponent je prováděna pomocí rozhraní IPIF[2].

Komponenty firmwaru Mikrosondy však komunikují po sběrnicích AXI4. Z tohoto důvodu se v každé z nich nachází převodníky mezi sběrnicemi AXI4-Stream a FrameLink, AXI4-Lite a MI32, AXI4-Lite a IPIF. Opakovaný převod mezi sběrnicemi AXI4-Stream a FrameLink je zbytečný a převodníky zabírají zbytečné zdroje programovatelného pole FPGA.

Import firmwaru Mikrosondy do vývojového prostředí Vivado jako jedné komponenty je výhodný, protože umožňuje odstranit uvedené nedostatky. Firmware je tak možno optimalizovat, a tím ušetřit zdroje programovatelného pole. S komponentou je možno pracovat jako s celkem, což umožňuje snadný import funkcionality Mikrosondy na jinou platformu.

5.3.1 Návrh komponenty USONDA_CORE

Prvotním úkolem návrhu je optimalizovat stávající design. Cílem optimalizací je odstranit nadbytečné převodníky mezi rozhraním FrameLink a AXI4-Stream a nadbytečné komponenty AXI_LITE_IPIF, které se nachází v každé softwarově konfigurovatelné komponentě. Přes sběrnici AXI4-Lite je realizována komunikace s AXI_LITE_IPIF, která rozhraním IPIF



Obrázek 5.1: Optimalizovaná achitektura komponenty USONDA_CORE.

[1] http://www.liberouter.org/

[2] http://www.xilinx.com/support/documentation/ip_documentation/axi_lite_ipif_ds765.pdf

konfiguruje komponentu, případně z ní čte požadovaná data.

K AXI_LITE_IPIF lze rozhraním IPIF připojit více komponent. Ve firmwaru Mikrosondy tak lze mít tuto komponentu pouze jednu. S tou pak komunikují všechny komponenty firmwaru Mikrosondy přes rozhraní IPIF. DATA_EXPORT tak v sobě nemusí mít AXI_LITE_IPIF.

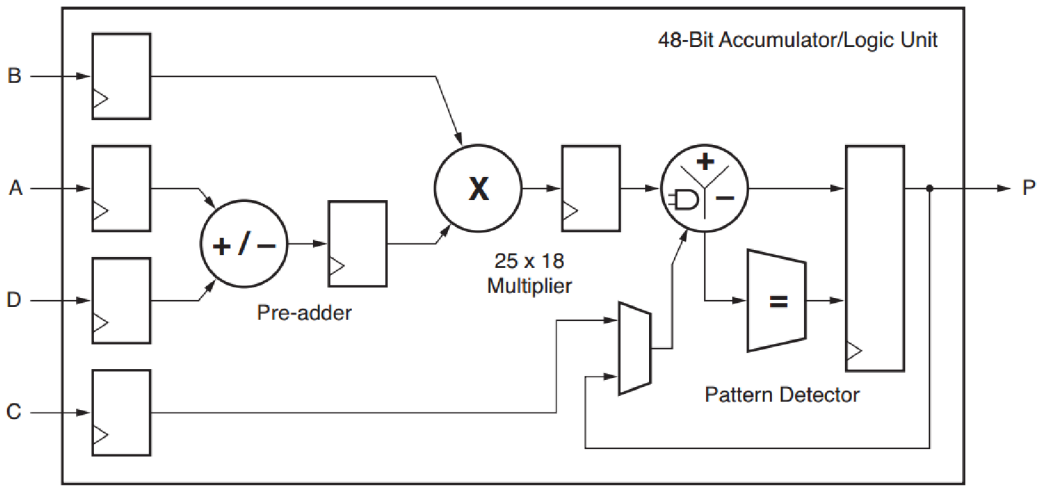
Aby bylo možno ke společné AXI_LITE_IPIF připojit komponenty, které pro svoji konfiguraci obsahují rozhraní MI32 (místo IPIF), je převodník mezi rozraním AXI4-Lite a MI32 nahrazen převodníkem mezi rozhraním MI32 a IPIF, který bylo nutné vytvořit. Rozhraním IPIF jsou komponenty napojeny do AXI_LITE_IPIF, která je nyní společná pro všechny komponenty. Změny je nutné provést u komponent HFEX a FILTER, které jsou součástí procesní části firmwaru, a u komponenty pro zdroj přesného času (TSU).

Pro předávání dat mezi komponentami je využito rozhraní AXI4-Stream. Aby BINDER a procesní část firmwaru mohli předávat data po této sběrnici, musí obsahovat převodníky mezi rozhraními AXI4-Stream a FrameLink. Komponenty však mohou mezi sebou komunikovat prostřednictvím rozhraní FrameLink a převodníky se mohou nacházet pouze na vstupu komponenty BINDER a na výstupu z procesní části firmwaru.

Komponenta USONDA_CORE v sobě zahrnuje stávající firmware Mikrosondy s výše uvedenými změnami. Komponenta obsahuje i DATA_EXPORT, který může být v budoucnosti nahrazen jinou alternativou exportu dat. Při nahrazování komponenty DATA_EXPORT jinou, bude výhodné pracovat přímo s daty z procesní části firmwaru. Z tohoto důvodu je možné DATA_EXPORT z komponenty odstranit generickým parametrem.

5.3.2 DSP48

Digital Signal Processing Block (DSP blok) je primitivum nacházející se v FPGA. Architektura DSP bloku je optimalizována pro provádění DSP funkcí s maximálním výkonem a nízkou spotřebou. Mezi DSP funkce patří především násobení, sčítání, odečítání. DSP blok proto obsahuje násobičku, sčítačky, odčítačky, akumulátory. Bloky jsou v DSP bloku zapojeny za sebou (viz obr. 5.2), což umožňuje výpočet složených výrazů (např. $X = A + B * C$). Implementace výpočtu DSP funkcí v programovatelné logice by spotřebovala velké množ-



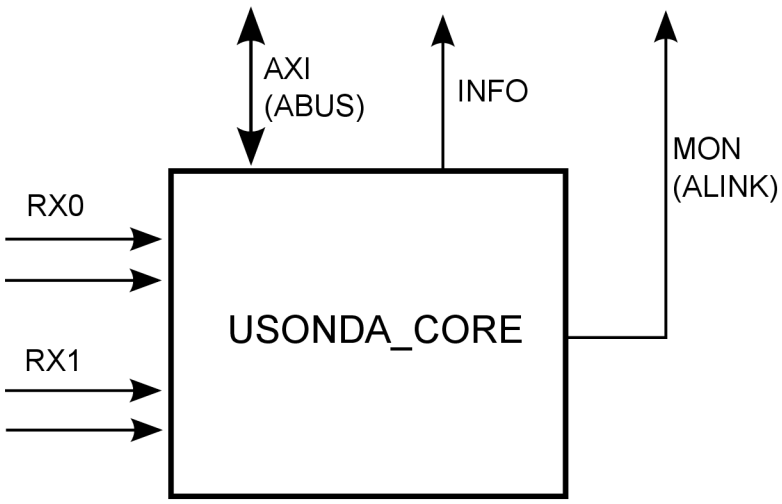
Obrázek 5.2: Funkcionalita bloku DSP48E1. [31, s. 9]

ství zdrojů logiky a dosažená frekvence výpočtu by byla řádově nižší (frekvence DSP bloku je přibližně 400 MHz). [2]

Blok DSP48 je schopen pracovat se 48-bitovými čísly. Tyto bloky se nachází jak v FPGA Spartan-6 (DSP48E), tak i v FPGA Zynq (DSP48E1). Pro práci s tímto blokem je možné využít existující makro, které je součástí vývojového prostředí. Pro FPGA Spartan-6 však toto makro obsahuje chybu. DSP blok, který se nachází TSU, je tedy v původní implementaci napsán pomocí primitiva bloku DSP48E. Protože se rozhraní DSP48E a DSP48E1 mírně liší, je třeba komponentu TSU mírně upravit. Makro pro FPGA Zynq chybu již neobsahuje, a mohlo být tedy použito. [31, s. 22]

5.3.3 Implementace

Implementace komponenty v sobě zahrnuje všechny změny uvedené v návrhu. Rozhraní výsledné komponenty (viz obr. 5.3) se skládá ze dvou vstupních (monitorovacích) portů (RX0, RX1), jednoho výstupního portu (MON) a jednoho portu pro konfiguraci (AXI4). Každý z vstupních i výstupních portů (RX0, RX1, MON) se skládá ze dvou AXI4-Stream rozhraní, jedno je určeno pro přenos dat, druhé pro řízení přenosu. Porty jsou určeny primárně pro připojení ke komponentám AXI_ETHERNET. Výjimkou je konfigurace komponenty USONDA_CORE bez DATA_EXPORT. V tom případě jsou data předávána pouze po datové části portu MON. USONDA_CORE byla implementován jako RSoC Framework akcelerátor bez vstupního rozhraní AXI4-Stream, který není využit.



Obrázek 5.3: Rozhraní komponenty USONDA_CORE.

Pro konfiguraci obsažených komponent složí generické parametry. Pro správnou funkčnost je třeba zvolit především:

- **C_DATA_WIDTH** – datová šířka sběrnice AXI4-Stream,
- **C_USER_WIDTH** – šířka signálu USER na sběrnici AXI4-Streamu,
- **TSU_REF_CLK_FREQ** – frekvence vstupního hodinového signálu,
- **RX_BINDER_METHOD** – metoda výběru vstupu komponenty BINDER,
  - 0 – most_occupied,

- 1 – round_robin,
- 2 – framed,

- **EXPORT_IPVER** – verze IP protokolu,
  - 4 – IPv4,
  - 6 – IPv6,
  - 0 – nepoužívat DATA_EXPORT.

Komponentu USONDA_CORE lze ve firmwaru pro Xilinx Zynq taktovat až na frekvenci 203,493 MHz (zjištěno nástrojem Xilinx ISE) a využije dle odhadů z nástrojů Xilinx ISE a Vivado množství logických zdrojů, které je uvedeno v tabulce 5.1.

	ISE		Vivado	
	S HW exportem	Se SW exportem	S HW exportem	Se SW exportem
Slice LUT	10712 (20 %)	10040 (18 %)	11123 (21 %)	9919 (19 %)
LUT Flip Flop	3618 (30 %)	3207 (29 %)	5840 (5 %)	4957 (5 %)
DSP48	9 (4 %)	9 (4 %)	9 (4 %)	9 (4 %)
Block RAM	3 (2 %)	3 (2 %)	34 (24 %)	34 (24 %)
BUFG	1 (3 %)	1 (3 %)	1 (3 %)	1 (3 %)

Tabulka 5.1: Využití zdrojů Xilinx Zynq komponentou USONDA_CORE.

Pro AXI_LITE_IPIF je třeba definovat adresové prostory jednotlivých komponent, které se nachází v USONDA_CORE. Rozdělení adresové prostoru komponenty je uvedeno v tabulce 5.2.

5.3.4 Integrace do prostředí EDK a Vivado

Komponentu USONDA_CORE je třeba importovat do prostředí EDK (pro použití na stávající platformě) a do vývojového prostředí Vivado. Pro prostředí EDK je třeba definovat soubor `MPD`, ve kterém se specifikuje rozhraní komponenty a generické konstanty, a soubor `PAO`, ve kterém jsou hierarchicky definovány použité komponenty. Pro portování komponenty do prostředí Vivado je nutné využít integrované grafické prostředí, ve kterém je definováno jak rozhraní, tak i použité komponenty. K importování komponenty do vývojového prostředí Vivado lze využít návod uvedený v dokumentu *Packaging Custom AXI IP for Vivado IP Integrator* [7].

5.4 Firmware pro Xilinx Zynq

Firmware pro Xilinx Zynq obsahuje vytvořenou komponentu USONDA_CORE a komponenty AXI_ETHERNET (TEMAC), které zprostředkovávají připojení komponenty USONDA_CORE k Ethernetovým rozhraním. Ve firmwaru se dále nachází RSOC_BRIDGE, prostřednictvím které je komponenta USONDA_CORE konfigurována.

Jednotka	Bázová adresa	Nejvyšší adresa	Typ	Poznámka
HFEX	0x00000	0x3FFFF	R/W	-
FILTER	0x40000	0x7FFFF	R/W	-
TSU	0x80000	0x8FFFF	R/W	-
DATA_EXPORT	0x90000	0x9FFFF	R/W	-
IN0_WORDS	0xA0000	0xA0003	R	počet přijatých slov rozhraním 0
IN0_FRAMES	0xA0004	0xA0007	R	počet přijatých rámců rozhraním 0
IN1_WORDS	0xA0008	0xA000B	R	počet přijatých slov rozhraním 1
IN1_FRAMES	0xA000C	0xA000F	R	počet přijatých rámců rozhraním 1
BINDER_WORDS	0xA0010	0xA0013	R	počet slov přijatých komponentou BINDER
BINDER_FRAMES	0xA0014	0xA0017	R	počet rámců přijatých komponentou BINDER
PROC_WORDS	0xA0018	0xA001B	R	počet slov přijatých procesní linkou
PROC_FRAMES	0xA001C	0xA001F	R	počet rámců přijatých procesní linkou
OUT_DATA_WORDS	0xA0020	0xA0023	R	počet slov odeslaných dat
OUT_DATA_FRAMES	0xA0024	0xA0027	R	počet rámců odeslaných dat
OUT_CTL_WORDS	0xA0028	0xA002B	R	počet slov odeslaných kontrolních dat
OUT_CTL_FRAMES	0xA002C	0xA002F	R	počet rámců odeslaných kontrolních dat
NEGATION	0xA0030	0xA0033	R/W	negace

Tabulka 5.2: Adresový prostor komponenty USONDA_CORE.

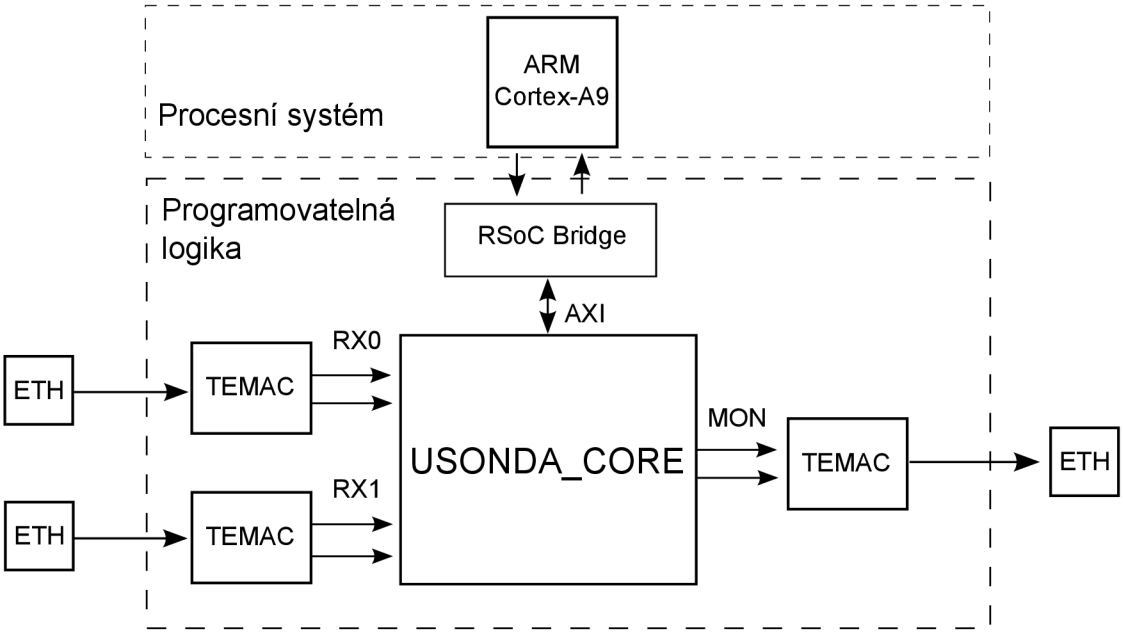
5.4.1 Firmware s hardwarovým exportérem dat

Pro export dat do sběrného zařízení je využita komponenta DATA_EXPORT, která se nachází v USONDA_CORE. Ve firmwaru se nachází tři komponenty AXI_ETHERNET (dvě na odposlouchávací porty a jedna pro port určený k exportu dat). RSOC_BRIDGE, který je připojen k procesoru ARM, je určen ke konfiguraci komponent. Architektura firmwaru je zachycena na obrázku 5.4.

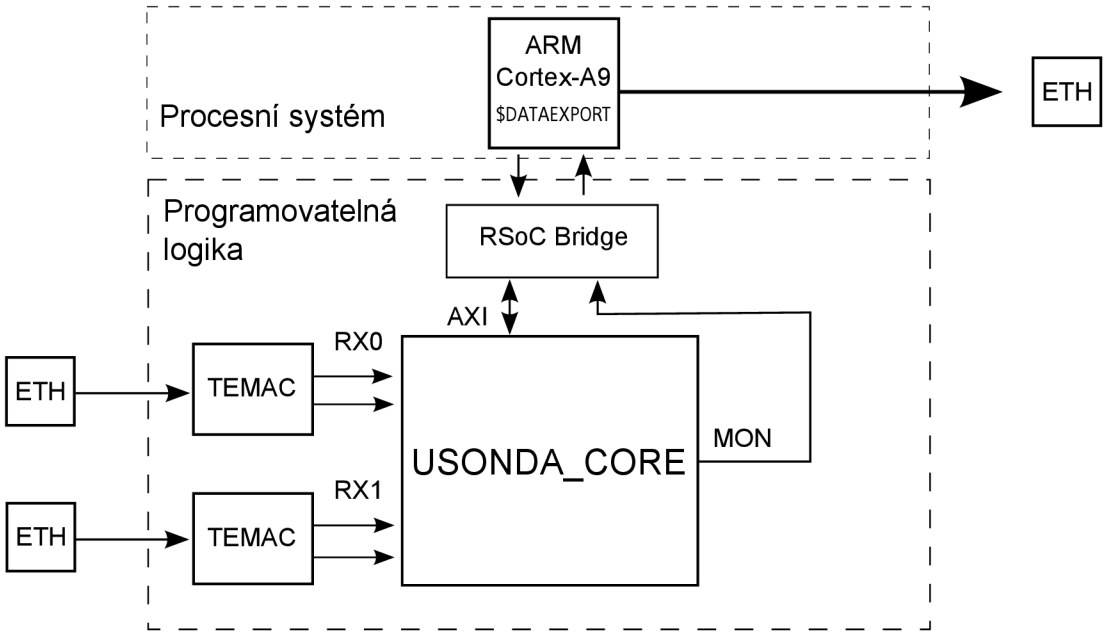
5.4.2 Firmware se softwarovým exportérem dat

Předchozí implementace firmwaru koresponduje s původní implementací firmwaru aplikace Mikrosonda. Aby však byly využity možnosti, které nabízí procesor ARM a RSOC_BRIDGE, byla původní implementace firmwaru změněna.

Firmware obsahuje pouze dvě komponenty AXI_ETHERNET, které přijímají data z od-



Obrázek 5.4: Architektura firmwaru s hardwarovým exportem dat.



Obrázek 5.5: Architektura firmwaru se softwarovým exportem dat.

poslechových portů. V USONDA_CORE je vypnutý hardwarový export dat komponentou DATA_EXPORT a filtrovaná data jsou předávána do RSOC_BRIDGE. Tato data jsou zpřístupněná z procesoru ARM aplikaci softwarového exportu dat prostřednictvím ovladačů RSoC Driver. Firmware je znázorněn na obrázku 5.5 a aplikace pro export dat je popsána v kapitole 5.12.

V tabulce 5.3 je porovnáno využití zdrojů FPGA původním firmwarem Mikrosondy a vytvořeným. Vytvořený firmware (díky procesnímu systému Zynq) oproti původnímu ne-

musí obsahovat procesor MicroBaze, dvě komponenty TEMAC, MCB (Memory Controller Block) a spousty řadičů periferních zařízení (UART, GPIO, atd.).

	Firmware s USONDA_CORE a s SW exportem dat (Zynq)	Původní firmware (Spartan-6)
Slice LUT	26852 (50 %)	34965 (37 %)
LUT Flip Flop	29969 (28 %)	30085 (16 %)
DSP48	9 (4 %)	14 (7 %)
Block RAM	61 (44 %)	105 (39 %)
BUFG	6 (19 %)	10 (62 %)

Tabulka 5.3: Využití zdrojů Xilinx Zynq firmwary Mikrosondy.

5.5 HWIO pro RSoC Framework

Protože všechny komponenty, které se nachází v programovatelném poli FPGA, jsou k procesoru ARM připojeny prostřednictvím RSoC Frameworku, není možné použít stávající knihovnu `hwio`. Ta pracuje nad strukturou FDT (popisující hardware), a se souborem `/dev/mem` mapovaným do paměti, který umožňuje přistupovat k adresovému prostoru komponent.

RSoC Framework poskytuje celý adresový prostor akcelerátoru prostřednictvím mapovatelného souboru (např. `/dev/rsocb_acc0`), který se do paměti mapuje jako jeden celek. Adresové prostory komponent, které se nachází v USONDA_CORE, není možné mapovat do paměti odděleně, jak tomu bylo u původní knihovna `hwio`.

Specifikace komponent, které jsou připojeny k RSoC Bridge, není uvedena v FDT. Aby bylo možné využívat stávající aplikace pro konfigurování komponent, bylo nutné upravit knihovnu hwio pro práci s RSoC Frameworkem.

5.5.1 Specifikace komponent v HW

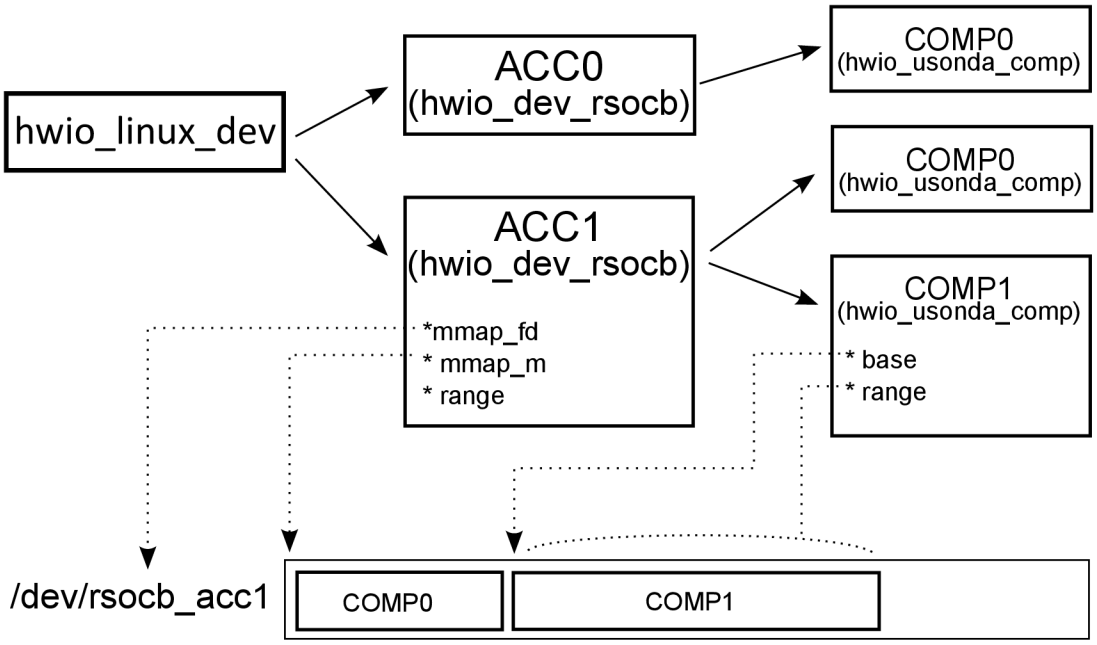
Akcelerátory, které se nachází v hardwaru, jsou popsány strukturou `hwio_dev_rsocb` (např. celá komponenta USONDA_CORE). Komponenty, které se nacházejí v akcelerátorech, jsou popsány strukturou `hwio_usonda_comp`. Těmito strukturami jsou předem definovány popisy akcelerátorů (může existovat více verzí jednoho akcelerátoru) a komponent. Výběr popisu akcelerátoru je prováděn na základě jeho verze, která je dána hodnotou v souboru

`/sys/bus/rsocb_bus/devices/rsocb_accX/user_info`,

kterou definuje akcelerátor (X značí číslo akcelerátoru).

Při inicializaci knihovny `hwio` se najdou všechny dostupné akcelerátory (podle existence souboru `user_info`), z příslušného souboru `user_info` se zjistí aktuální verze akcelerátoru a ze souboru `user_abus_range` se získá velikost adresového prostoru. Pro každý nalezený akcelerátor je do struktury `hwio_linux_dev` vložena jedna struktura `hwio_dev_rsocb` (viz obr. 5.6), která dle verze akcelerátoru obsahuje informace o obsažených komponentách (struktury `hwio_usonda_comp`). Se strukturou `hwio_linux_dev` pak přímo pracuje knihovna `hwio`.

Komponenta USONDA_CORE je tedy reprezentovaná strukturou `hwio_linux_dev`, která obsahuje popis jedné verze akcelerátoru. Popis obsahuje komponenty, které se nachází v USONDA_CORE, reprezentované strukturami `hwio_usonda_comp`.



Obrázek 5.6: Struktura knihovny `hwio` pro RSoC Framework.

5.5.2 Čtení a zápis dat

Po inicializaci knihovny struktura `hwio_linux_dev` obsahuje informace o dostupných akcelerátorech a komponentách, které se v akcelerátorech nachází. Pro čtení či zápis do konkrétní komponenty se prochází všechny akcelerátory a hledá se odpovídající komponenta. Po jejím nalezení je nutné otevřít a namapovat soubor akcelerátoru, kterého je komponenta součástí, do paměti (pokud tak již nebylo učiněno).

Mapovatelný soubor každého akcelerátoru RSoC Frameworku lze do paměti mapovat pouze od počátku (od báze 0). Z toho důvodu není možné, aby každá komponenta měla mapovaný svůj adresový prostor do paměti odděleně (jak tomu bylo u stávající verze knihovny `hwio`). Soubor akcelerátoru je otevřen pouze jednou a do paměti mapován jako jeden celek o velikosti, která byla načtena při inicializaci knihovny ze souboru `user_abus_range`. Do struktury akcelerátoru je uložen file descriptor otevřeného souboru (návratová hodnota funkce `open()`) a ukazatel (získaný funkcí `mmap()`), který specifikuje umístění mapovaného souboru v paměti. Každé komponentě tohoto akcelerátoru je dle její specifikace přidělen ukazatel (offset), který specifikuje umístění adresového prostoru komponenty v adresovém prostoru akcelerátoru (viz obr. 5.6).

Po nalezení odpovídající komponenty a jejím namapování do paměti je možné pracovat s jejím adresovým prostorem (číst či zapisovat data). Pro zápis či čtení jednoho slova z určité adresy se k této adrese přičte offset komponenty, součet definuje adresu paměti, do které se následně zapíše, případně ze které se data přečtou.

5.6 Operační systém založený na linuxové distribuci

Pro vytvoření operačního systému založeného na linuxové distribuci je zapotřebí provést následující kroky:

1. Získáme toolchain, který obsahuje potřebné křížové kompilátory (kompilátor pro spe-

cifickou instrukční sadu cílové architektury), linkery a základní knihovny. Toolchain může být již předkompilovaný (dostupný ve formě binárních souborů) nebo je dostupný ve formě zdrojový kódů, jejichž kompilace je pořízena kompilaci linuxového jádra (např. GNU toolchain).

2. Provedeme kompilaci linuxového jádra získaným křížovým kompilátorem.

3. Vytvoříme obraz root file systému, který obsahuje aplikace a knihovny kompilované křížovým kompilátorem.

4. Vytvoříme bootloaderu pro cílovou platformu. Nejpoužívanějším je U-Boot, který je schopen bootovat a zavést linuxové jádro.

Protože každý z uvedených kroků může být velmi složitý a může lehce selhat, byly vytvořeny nástroje, které vytvoří veškeré součásti operačního systému. Příkladem takového systému jsou například Yocto a Buildroot. [21]

5.7 Das U-Boot

Das U-Boot (U-Boot) je open source zavaděč systému, který je určený především pro vestavné systémy a operační systém Linux. Je podporován v řadě architektur včetně ARM, PowerPC, MIPS, x86 a MicroBlaze. Díky bohaté sadě ovladačů různých periferních zařízení je možno zavést operační systém z různých zdrojů (například z SD karty nebo pomocí TFTP). [10]

Proces bootování lze přizpůsobit pomocí příkazové řádky a to bez nutnosti rekompilace zavaděče. Zavedení operačního systému Linux prostřednictvím TFTP lze provést následujícími příkazy:

```
1  > setenv ipaddr 192.168.0.3
2  > setenv serverip 192.168.0.2
3  > setenv kernel_image uImage
4  > setenv devicetree_image ze7000.dtb
5  > setenv ramdisk_image urootfs.cpio.gz
6  > setenv kernel_addr 0x2000000
7  > setenv dtb_addr 0x3000000
8  > setenv rootfs_addr 0x4000000
9  > tftp ${kernel_addr} ${kernel_image}
10 > tftp ${rootfs_addr} ${ramdisk_image}
11 > tftp ${dtb_addr} ${devicetree_image}
12 > setenv bootargs console=ttyPS0,115200 root=/dev/ram rw
13 > bootm ${kernel_addr} ${rootfs_addr} ${dtb_addr}
```

Na řádcích 1–2 probíhá konfigurace IP adresy zařízení a serveru, ze kterého bude probíhat stahování bootovacích souborů. Na řádcích 3–5 jsou do proměnných uloženy názvy souborů, které se budou ze serveru stahovat. Na řádcích 6–8 jsou nastaveny proměnné, které udávají adresu paměti, kam bude umístěn stažený soubor. Řádky 9–11 jsou soubory staženy ze serveru a umístěny na odpovídající adresu v paměti. Řádek 12 nastavuje argumenty bootování operačního systému a řádkem 13 je spuštěn samotný proces bootování. Pro uložení konfigurace je možno využít příkaz `saveenv`, následné stažení souborů a bootování lze spustit jediným příkazem `run jtagboot` .

5.8 Operační systém Linux pro platformu ZE7000

Aby bylo možné Mikrosondu konfigurovat pomocí stávajících aplikací, je nutné na procesoru ARM spustit operační systém Linux. Operační systém Linux, který je vytvořen pro platformu μG4-150 a její procesor MicroBlaze, nelze použít pro procesor ARM. Z tohoto důvodu je nutné pro platformu ZE7000 vytvořit nový operační systém Linux.

Pro vytvoření operačního systému lze využít prostředí Yocto nebo Buildroot. Firma Xilinx pro Zynq a jeho procesor ARM upravila jádro operačního systému Linux. Konfigurace jádra pro platformu ZE7000 je dostupná přímo ze stránek výrobce ve formě záplat a souboru `DTS`, který definuje strukturu FDT (viz kapitola 2.5.1). Konfigurace se importuje do prostředí Yocto nebo Buildroot.

5.9 Yocto

Yocto je open source projekt, který vznikl roku 2010 spoluprací mnoha výrobců hardwaru a dodavatelů open source operačních systémů.

Projekt Yocto poskytuje nástroje a metadata pro vytvoření vlastního operačního systému linuxového typu pro vestavěné zařízení bez ohledu na hardwarovou architekturu. Nástroji je možno vytvořit operační systém Linux pro architektury ARM, MIPS, PowerPC, x86 a x86-64. Vytváření systému je založené na architektuře OpenEmbedded, která umožňuje vývojářům vytvořit vlastní linuxovou distribuci, která je specifická pro jejich prostředí. Tato OpenEmbedded implementace se nazývá Poky.

Firma NetModule využívá nástroj Yocto pro tvorbu operačního systému Linux pro své platformy. Pro platformu ZE7000 jsou dostupné jak binární soubory již vytvořeného systému, tak i zdrojové soubory konfigurace. Vytvoření systému pro platformu ZE7000 je popsán podrobně v dokumentu [18].

Vytváření operačního systému Linux pro platformu ZE7000 v nástroji Yocto dle uvedeného návodu je však velmi zdlouhavé (asi 5 hodin na běžném PC) a zabere velkou část místa na disku (přibližně 15 GB). Z tohoto důvodu bylo výhodné využít alternativní nástroj Buildroot i přes to, že firma NetModule upřednostňuje nástroj Yocto.

5.10 Buildroot

Buildroot je nástroj, který zjednodušuje a automatizuje proces vytváření kompletního operačního systému Linux pro vestavné systémy. Je schopen generovat křížový překladač (GNU toolchain), linuxové jádro, root file systém a zavaděč pro vestavné systémy s architekturou procesoru ARM, PowerPC, MIPS, x86 atd. Buildroot nabízí základní konfiguraci pro několik nejpoužívanějších platforem (např. Raspberry Pi, Zedboard). [19, s. 1]

Cílový systém lze snadno nastavit díky konfiguračním rozhraním `menuconfig`, `gconfig` a `xconfig`. Buildroot obsahuje podporu pro velké množství balíčků aplikací a knihoven, které lze konfigurací snadno integrovat do uživatelského prostoru. Integrovat je možno i vlastní aplikace importováním vlastních balíčků. Samotný proces tvorby operačního systému oproti nástroji Yocto trvá mnohem kratší dobu (přibližně 15–30 min).

5.11 Buildroot pro ZE7000

Pro účely této práce byl operační systém Linux vytvořen v nástroji Buildroot. Nástroj však neobsahuje žádnou podporu pro platformu ZE7000. Aby bylo možné pro platformu vytvořit operační systém, bude nutné tuto podporu do nástroje Buildroot ručně vytvořit.

5.11.1 Podpora pro ZE7000

Podpora pro platformy je uložena v adresáři `board/`. V tomto adresáři se nachází již konfigurační soubory a záplaty pro některé platformy. Pro vytvoření podpory platformy ZE7000 je nutné provést následující kroky:

1. Vytvoříme adresář pro novou platformu (`netmodule/ze7000/`), ve kterém budou umístěny potřebné soubory.
2. Do adresáře umístíme záplaty z repositáře firmy NetModule[3]. Záplaty je výhodné upravit tak, aby v sobě neobsahovali úpravy pro soubor `ze7000.dts` (obsahující popis zařízení). Do souboru `ze7000.dts` pak vložíme všechny informace, které byly odstraněny ze záplat. Tento krok je výhodný pro pozdější upravování a přidávání dalších jednotek do souboru `DTS`.
3. V některém z konfiguračních rozhraní (např. `menuconfig` vyvolané příkazem `make menuconfig`) je nutné nastavit některé parametry, především je třeba upřesnit zdroj, ze kterého bude staženo jádro operačního systému Linux. Jako zdroj je nastaven repositář firmy Xilinx[4], ve kterém se nachází upravené linuxové jádro `xilinx-v14.5`.

Nezbytné parametry konfigurace pro Buildroot jsou:

```
1  BR2_arm=y
2  BR2_cortex_a9=y
3
4  BR2_TOOLCHAIN_EXTERNAL=y
5  BR2_TOOLCHAIN_EXTERNAL_CODESOURCERY_ARM201311=y
6
7  BR2_LINUX_KERNEL_CUSTOM_TARBALL=y
8  BR2_LINUX_KERNEL_CUSTOM_TARBALL_LOCATION=https://github.com/
     Xilinx/linux-xlnx/archive/52
     ae18dc7cc4f093fbef75e0ee9fa2e21ae08f72.tar.gz
9  BR2_LINUX_KERNEL_VERSION="custom"
10 BR2_LINUX_KERNEL_PATCH="board/netmodule/ze7000"
11 BR2_LINUX_KERNEL_USE_CUSTOM_CONFIG=y
12 BR2_LINUX_KERNEL_CUSTOM_CONFIG_FILE="board/netmodule/ze7000/
     ze7000_defconfig"
13 BR2_LINUX_KERNEL_UBOOT_IMAGE=y
14 BR2_LINUX_KERNEL_UIMAGE_LOADADDR="8000"
15 BR2_LINUX_KERNEL_DTS_SUPPORT=y
16 BR2_LINUX_KERNEL_CUSTOM_DTS_PATH="board/netmodule/ze7000/ze7000.
     dts"
```

[3] https://github.com/netmodule/meta-netmodule/tree/master/recipes-kernel/linux/linux-zynq
[4] https://github.com/Xilinx/linux-xlnx/archive/52ae18dc7cc4f093fbef75e0ee9fa2e21ae08f72.tar.gz

```
17
18 BR2_TARGET_ROOTFS_CPIO=y
19 BR2_TARGET_ROOTFS_CPIO_GZIP=y
20 BR2_TARGET_GENERIC_GETTY_PORT="ttyPS0"
```

Řádky 1–5 je zvolena kompilace pro procesor ARM Cortex-A9 a definován toolchain. Řádky 7–9 je specifikován repositář s linuxovým jádrem firmy Xilinx (zabalené v archivu), řádky 10–11 konfigurují linuxové jádro. Na řádku 12 je specifikován adresář, ve kterém se nachází záplaty pro jádro. Soubor se specifikací struktury FDT je nastaven řádky 13–14. Na řádcích 15–16 je povolen vytvoření obrazu (obsahující jádro) pro zavaděč U-Boot a je uvedena adresa paměti, kam zavaděč U-Boot umístí jádro operačního systému.

Povolení vytvoření root file systému ve formátu CPIO a zvolení jeho zabalení do archivu algoritmem GZIP je uvedeno na řádcích 18–19. Na řádku 20 je specifikován USB UART (`/dev/ttyPS0`) jako defaultní zařízení pro konzoli.

5.11.2 Sestavení

Po dokončení konfigurace (např. použitím rozhraní `menuconfig`) je možno sestavení operačního systému zahájit voláním příkazu `make`. Po dokončení operace se výsledné soubory nachází v adresáři `ouput/images/`. Pro bootování z TFTP serveru je nutné na tento server zkopírovat následující soubory:

- `uImage` – jádro operačního systému Linux připravené pro zavedení zavaděčem U-Boot,
- `urootfs.cpio.gz` – root file systém,
- `ze7000.dtb` – binární soubor FDT.

5.11.3 Tvorba balíčků

Pro integraci uživatelských aplikací do uživatelského prostoru je nutné zdrojové kódy aplikace přeložit křížovým překladačem, výsledný binární soubor zkopírovat do root filesystému. Tento postup je zdlouhavý a obtížný. Snazší alternativou je vytvoření balíčku, který je následně integrován do nástroje Buildroot. Při vytváření operačního systému Linux je aplikace přeložena (křížovým překladačem) a vložena do uživatelského prostoru. [19, 33–38]

Pro vytvoření balíčku pro aplikaci `xxx` a jeho integraci je nutné:

- Vytvořit archiv obsahující zdrojové soubory aplikace, a vložit ho do adresáře `dl/`.
- Vytvořit adresář pro nový balíček a příslušné soubory `Config.in` a `xxx.mk`:

```
1 $ mkdir package/xxx
2 $ touch package/xxx/Config.in
3 $ touch package/xxx/xxx.mk
```

- Specifikovat konfigurační volby v souboru `Config.in`, pro jednoduchost stačí pouze volba pro použití balíčku:

```
1 config BR2_PACKAGE_XXX
2    bool "xxx"
3    help
4      Example program xxx.
```

- Specifikovat pravidla pro překlad v souboru `xxx.mk`:

```
1  XXX_VERSION = 1.0
2
3  define XXX_BUILD_CMDS
4     $(MAKE) CC=$(TARGET_CC) LD=$(TARGET_LD) -C $(@D) all
5  endef
6
7  define XXX_INSTAL_TARGET_CMDS
8     $(INSTALL) -D -m 0755 $(@D)/xxx $(TARGET_DIR)/usr/bin
9  endef
10
11 define XXX_UNINSTALL_TARGET_CMDS
12    rm -f $(TARGET_DIR)/usr/bin/xxx
13 endef
14
15 define XXX_CLEAN_CMDS
16   $(MAKE) -C $(@D) clean
17 endef
18
19 $(eval $(generic-package))
```

- Zpřístupnit balík z konfiguračního rozhraní `menuconfig` přidáním následujího řádku na vhodné místo v souboru `package/Config.in`:

```
1  source "package/xxx/Config.in"
```

5.11.4 Integrace ovladačů RSoC Frameworku

Pro práci s RSoC Frameworkem je nutné integrovat do nástroje Buildroot ovladač RSoC Driver. Pro účely této práce byl firmou RehiveTech poskytnut ovladač s hotovou podporou pro Buildroot.

Pro správnou funkcionalitu RSoC Frameworku je nutné do souboru `DTS` (`ze7000.dts`) dodat následující řádky (se správnou bázovou adresou):

```
1  rsoc_bridge: rsoc_bridge@80000000 {
2          compatible = "xlnx,rsoc-bridge-1.00.a";
3          reg = < 0x80000000 0x10000 >;
4  };
```

Po bootování operačního systému s instalovaným ovladačem je možné načíst ovladač RSoC Bridge (řádek 1) a jeho řadiče DMA (řádek 2):

```
1  # modprobe rsocb_drv
2  # modprobe xdma_drv
```

5.12 Aplikace pro export dat

Firmware s RSoC Frameworkem umožňuje export dat do procesoru ARM. Filtrovaná data není nutné z hardwaru odesílat do sběrného zařízení, ale lze je předat procesoru ARM, kde jsou dále zpracována. Příkladem je softwarový export dat do sběrného budu.

Aplikace pro export dat (označena jako `soft_dataexport`) musí v první řadě otevřít zařízení (např. `/dev/rsocb_acc0`):

```
1 | int device = open(path, O_RDWR);
```

Po úspěšném otevření zařízení je možno navázat spojení s cílovým serverem. Ustanovením spolehlivého spojení je započato vyčítání a odesílání dat v nekonečné smyčce:

```
1 | while(true)
2 | {
3 |         if(( n = read(device, buffer, BUF_SIZE)) > 0)
4 |         {
5 |                 write(socket, buffer, n);
6 |         }
7 | }
```

Na řádku 3 probíhá čtení dat z hardwaru. Pokud jsou přečtena nějaká data, jsou na řádku 5 zápisem do socketu odeslána. Před použitím aplikace je nutné mít načtené ovladače pro RSoC Framework. Aplikaci je pak možné spustit například příkazem tvaru:

```
1 | # soft_dataexport /dev/rsocb_acc0 192.168.0.2 55555
```

První parametr specifikuje soubor zařízení, druhý adresu cílového serveru a třetí číslo portu.

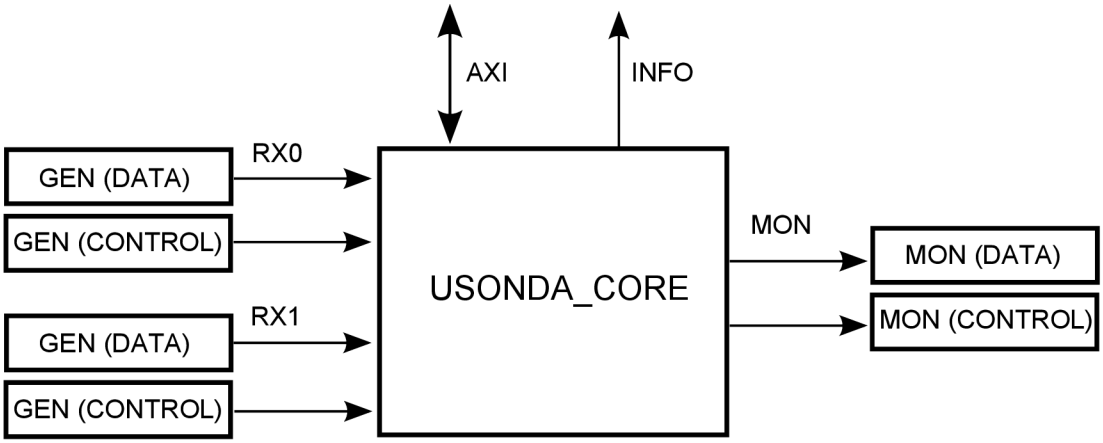
Kapitola 6

Testování

Následující kapitola popisuje testování, které má za úkol ověřit funkcionalitu všech vytvářených částí, simulační prostředí pro komponentu USONDA_CORE, ladění designu v hardwaru s využitím ladicího nástroje Vivado Logic Analyzer a otestování funkcionality celá aplikace Mikrosonda na reálném síťovém provozu.

6.1 Simulace komponenty USONDA_CORE

Pro komponentu USONDA_CORE je vytvořeno behaviorální simulační prostředí pro otestování základní funkčnosti obvodu. Simulace obsahuje komponenty pro generování a příjem provozu na sběrnicích AXI4-Stream. Komponenty pro generování odesílají data ze vstupních souborů do USONDA_CORE, kde jsou zpracována. USONDA_CORE odesílá data do přijímacích komponent, které ukládají data do výstupních souborů. Vstupní a výstupní soubory je možno porovnávat, a tak ověřit funkcionalitu.



Obrázek 6.1: Behaviorální simulační prostředí pro USONDA_CORE

6.2 Ladění firmwaru v hardwaru

Při testování vytvořeného firmwaru byly oproti simulacím zjištěny nesrovnalosti, které způsobovaly zastavení exportu dat. Pro nalezení problému bylo třeba využít ladicí nástroj Vi-

vado Logic Analyzer, který se nachází v prostředí Vivado a který umožňuje snímat stav hardwaru v reálném čase.

Pro hardwarové ladění je nutné při návrhu firmwaru v prostředí Vivado požadované signály označit jako ladicí. Při syntéze jsou do firmwaru automaticky vloženy monitory (ILA), do kterých jsou zapojeny ladicí signály, a řadič monitorů (DBG_HUB).

Ladicí nástroj je schopen zachytit krátký časový interval (1024 hodinových taktů) dění na fyzické úrovni od okamžiku, který je definován podmínkami (stavy vybraných signálů). Prostředí Vivado slouží pro definování podmínek a k interpretaci zachycených dat.

6.3 Testování na testovacím provozu

Funkcionalitu konečné podoby aplikace Mikrosonda, která vnikla v rámci této práce, je nutné otestovat na reálném provozu. Před samotným testováním je třeba nakonfigurovat platformu dle následujících kroků:

- nahrát firmware do FPGA,
- spustit operační systém Linux,
- nakonfigurovat komponenty firmwaru a načíst ovladače RSoC Driver (viz níže),
- spustit aplikaci pro export dat.

Následující příkazy slouží pro nastavení celé sondy a jejich spuštění je nezbytné po každém startu systému:

```
1  modprobe rsocb_drv
2  modprobe xdma_drv
3
4  abus_tool rsocb_acc0 0x00000000 0x00000001
5  abus_tool rsocb_acc0 0x00000004
6  abus_tool rsocb_acc0 0x00008074 0x41400000
7  abus_tool rsocb_acc0 0x00008078 0x45444342
8  abus_tool rsocb_acc0 0x000080A0 0x4d4c0000
9  abus_tool rsocb_acc0 0x000080A4 0x49484746
10 abus_tool rsocb_acc0 0x000080A8 0x5d5c4f4e
11 abus_tool rsocb_acc0 0x000080B0 0x4d4c4b4a
12 abus_tool rsocb_acc0 0x000080C8 0x00005f5e
13 abus_tool rsocb_acc0 0x000080CC 0x00005f5e
14 abus_tool rsocb_acc0 0x000080E4 0x51504f4e
15 abus_tool rsocb_acc0 0x000080F8 0x55545352
16 abus_tool rsocb_acc0 0x000080FC 0x43424140
17 abus_tool rsocb_acc0 0x00008104 0x4f4e4d4c
18 abus_tool rsocb_acc0 0x00008108 0x5f5e5d5c
19 abus_tool rsocb_acc0 0x0000810C 0x59585756
20 abus_tool rsocb_acc0 0x00008110 0x47464544
21 abus_tool rsocb_acc0 0x00008118 0x5d5c5b5a
22 abus_tool rsocb_acc0 0x0000811C 0x4b4a4948
23 abus_tool rsocb_acc0 0x00008120 0x00005f5e
```

```
24 abus_tool rsocb_acc0 0x00008124 0x00005f5e
25 abus_tool rsocb_acc0 0x00008128 0x00005f5e
26 abus_tool rsocb_acc0 0x0000812C 0x4f4e4d4c
27 abus_tool rsocb_acc0 0x00008134 0x53525150
28 abus_tool rsocb_acc0 0x00008148 0x57565554
29 abus_tool rsocb_acc0 0x00008160 0x5b5a5958
30 abus_tool rsocb_acc0 0x00008174 0x5f5e5d5c
31 abus_tool rsocb_acc0 0x00000008 0x00000003
32 abus_tool rsocb_acc0 0x0000000C 0x00000007
33 abus_tool rsocb_acc0 0x00000000 0x00000000
34
35 ifconfig eth0 192.168.0.3
36
37 mkdir -p  /etc/liberouter/run
38 atsuctl -v -t 217.31.205.226 -w &
```

Tyto příkazy řeší načtení ovladačů RSoC Driver (řádky 1–2), konfiguraci procesní části firmwaru (řádky 4–33), spuštění a nakonfiguraci exportovacího ethernetového rozhraní (řádek 35) a spuštění aplikace pro konfiguraci TSU (řádky 37–38). Konfigurace procesní části firmwaru je automaticky vytvořena při generování zdrojových kódů a nemusí se vždy shodovat.

Při testování byla platforma ZE7000 (s aplikací Mikrosonda) připojena ke dvěma PC. Jedno z nich bylo připojeno k monitorovacímu ethernetovému portu a sloužilo jako generátor provozu. Pro testování byl zachycen běžný provoz nástrojem `wireshark`. Tento provoz je nástrojem `tcpreplay` odesílán do Mikrosondy.

Druhé PC je připojeno k exportovacímu ethernetovému portu Mikrosondy a slouží pro příjem exportovaných dat. Příjem dat je realizován nástrojem `netcat`, pro sledování komunikace je spuštěn nástroj `wireshark`. Nástroj `netcat` je spuštěn příkazem:

```
1 $ nc -ul 7000
```

a aplikace pro export dat je použita následovně:

```
1 # soft_dataexport /dev/rsocb_acc0 192.168.0.2 7000
```

Exportovaná jsou pouze data (s předřazenou INI3 hlavičkou viz 2.3.1), která jsou přijata vstupním rozhraním Mikrosondy a která vyhoví některému z filtračních pravidel. Porovnání dat z nástrojů `wireshark` a `netcat` potvrzuje funkčnost řešení i na reálném provozu.

Kapitola 7

Závěr

Úkolem této práce bylo seznámit se s aplikací Mikrosonda vyvíjenou v projektu *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace* a s platformou Xilinx Zynq a možnostmi vývoje aplikací na této platformě, dále na základě studia navrhnout a implementovat prototyp Mikrosondy integrující stávající funkcionalitu na platformě Xilinx Zynq.

První část práce popisuje hardwarové i softwarové součásti aplikace Mikrosonda, platformu μG4-150 a platformu NetModule ZE7000 obsahující Xilinx Zynq. Na základě studia platforem a aplikace Mikrosonda byla provedena teoretická analýza problému.

Druhá část práce se zabývá návrhem a implementací prototypu aplikace Mikrosonda, jehož architektura byla upravena tak, aby reflektovala integrovaný procesor ARM Cortex-A9. Hlavní funkcionalita aplikace Mikrosonda je integrována v komponentě USONDA_CORE. Kvůli omezené podpoře architektury Xilinx Zynq v prostředí EDK byla mikrosonda portována do nového vývojového prostředí Vivado. Pro komunikaci komponenty USONDA_CORE, která se nachází v programovatelném poli FPGA, a procesoru ARM je využit RSoC Framework. Pro procesor ARM byl v nástroji Buildroot vytvořen operační systém Linux obsahující ovladač RSoC Driver a knihovnu `hwio`, která byla upravena pro práci s RSoC Frameworkem.

Prototyp byl rozšířen o možnost vypnutí hardwarového exportu dat. Tato varianta umožňuje zpřístupnit filtrovaná data prostřednictvím RSoC Frameworku procesoru ARM, kde je možné data dále zpracovávat. V průběhu tvorby této práce byla vytvořena softwarová aplikace, která vyčítá filtrovaná data z hardwaru a odesílá je spolehlivým protokolem TCP/IP do sběrného zařízení.

Přechod na platformu Xilinx Zynq, která obsahuje procesor ARM Cortex-A9, poskytuje výkon pro realizaci úkolů, jejichž implementace v programovatelném poli FPGA by zabrala velké množství logických zdrojů, byla náročná, případně dokonce nemožná.

Implementovaný prototyp je možné využít pro spolehlivé doručení dat po sítí protokolem TCP/IP případně šifrovaným kanálem (např. SSH) do sběrného zařízení. Díky flexibilitě programování procesoru a jeho výkonu lze implementovat algoritmy pracující na aplikační vrstvě L7. Export dat na externí médium prostřednictvím USB 2.0 rozhraní a vytvoření webové rozhraní pro konfiguraci Mikrosondy lze s výhodou implementovat díky operačnímu systému Linux. Tyto aspekty umožní provozovat zařízení bez nutnosti připojení ke zbytku systému pro zákonné odposlechy. Na platformě ZE7000 se ovšem nyní nenachází USB konektor. Ten bude pravděpodobně doplněn v příští verzi platformy ZE7000.

Literatura

[1] *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace* [online]. 2014 [cit. 2014-05-04].
URL http://www.fit.vutbr.cz/research/grants/index.php.cs?id=517

[2] Altera: *Digital Signal Processing Blocks in Stratix Series FPGAs* [online]. 2014 [cit. 2014-05-06].
URL http://www.altera.com/devices/fpga/stratix-fpgas/about/dsp/stx-dsp-block.html

[3] ARM: *AMBA 4 AXI4-Stream Protocol Specification* [online]. 2010 [cit. 2014-05-03], ID030510.
URL https://silver.arm.com/download/download.tm?pv=1198016

[4] ARM: *AMBA AXI and ACE AXI3 Protocol Specification: AXI3, AXI4, and AXI-4Lite, ACE and ACE-Lite* [online]. 2013 [cit. 2014-05-03], ID022613.
URL https://silver.arm.com/download/download.tm?pv=1377613

[5] ARM: *Cortex-A9 Processor* [online]. 2014 [cit. 2014-01-23].
URL http://www.arm.com/products/processors/cortex-a/cortex-a9.php

[6] Benáček, P.: *Ethernetový tester pro vysokorychlostní sítě*. Diplomová práce, České vysoké učení technické v Praze, Praha, 2012.

[7] Buli, D.; Corbett, K.: *Packaging Custom AXI IP for Vivado IP Integrator* [online]. June 2013 [cit. 2014-05-06], XAPP1168 (v1.0).
URL http://www.xilinx.com/support/documentation/application_notes/xapp1168-axi-ip-integrator.pdf

[8] Byrne, J.: *ARM Outmuscles Atom on Benchmark* [online]. April 2011 [cit. 2014-01-25].
URL http://parisbocek.typepad.com/blog/2011/04/arm-outmuscles-atom-on-benchmark-1.html/

[9] CNXSoft: *Xilinx Zynq-7000 Extensible Processing Platform (EPP): Dual Cortex A9 + FPGA SoC* [online]. March 2012 [cit. 2014-05-04].
URL http://www.cnx-software.com/2012/03/29/xilinx-zynq-7000-extensible-processing-platform-epp-dual-cortex-a9-fpga-soc/

[10] DENX: *The Universal Boot Loader („Das U-Boot“)* [online]. May 2012 [cit. 2014-05-10].
URL http://www.denx.de/wiki/publish/U-Bootdoc/U-Bootdoc.pdf

[11] eLinux.org: *Device Tree* [online]. January 2014 [cit. 2014-05-04].
URL http://elinux.org/Device_Tree

[12] Korček, P.: *Softvérové vrstvy pre uSondu* [online]. June 2012 [cit. 2014-05-04].
URL merlin.fit.vutbr.cz/wiki-sec6net/index.php/USonda_uBlaze_software

[13] Korček, P.: *uG4-150 embedded platform for wire-speed network packet processing* [online]. 2013 [cit. 2013-12-24].
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10402

[14] Korček, P.; Viktorin, J.; Košař, V.: *Usonda prototyp* [online]. April 2014 [cit. 2014-05-05].
URL https://merlin.fit.vutbr.cz/wiki-sec6net/index.php/USonda_prototyp

[15] NetModule: *ZE7000: Zynq Platform for Ethernet-centric Applications* [online]. 2013 [cit. 2014-01-21].
URL http://www.netmodule.com/products/sbc-eval-sys/ZE7000.html

[16] NetModule: *User Manual: NetModule „ZE7000 Platform“* [online]. 2013 [cit. 2014-01-21], v1.2.
URL
merlin.fit.vutbr.cz/wiki-sec6net/images/f/fe/Usr_man_ZE7000_v1.2.pdf

[17] NetModule: *Mars ZX3: SoC Module* [online]. March 2013 [cit. 2014-01-23], v0.42.
URL
merlin.fit.vutbr.cz/wiki-sec6net/images/b/b0/Mars_ZX3_UserManual.pdf

[18] NetModule: *Yocto* [online]. 2014 [cit. 2014-05-10].
URL https://github.com/netmodule/meta-netmodule/wiki/Yocto

[19] Petazzoni, T.: *The Buildroot user manual* [online]. February 2014 [cit. 2014-05-10].
URL http://buildroot.uclibc.org/downloads/manual/manual.pdf

[20] RehiveTech: *RSoC Framework: Description* [online]. 2014 [cit. 2014-01-30].
URL http://rsoc-framework.com/description/

[21] Viktorin, J.: *HW/SW Co-design for the Xilinx Zynq Platform*. Diplomová práce, Vysoké učení technické v Brně, Brno, 2013.

[22] Viktorin, J.: *RSoC Bridge for Zynq* [online]. November 2013 [cit. 2014-01-30].
URL
https://merlin.fit.vutbr.cz/wiki-iot/index.php/RSoC_Bridge_for_Zynq

[23] Viktorin, J.: *Knihovna hwio* [online]. April 2013 [cit. 2014-05-04].
URL https://merlin.fit.vutbr.cz/wiki-sec6net/index.php/Knihovna_hwio

[24] Viktorin, J.: *RSoC Driver* [online]. January 2014 [cit. 2014-01-30].
URL https://merlin.fit.vutbr.cz/wiki-iot/index.php/RSoC_Driver

[25] Xilinx: *Embedded Processor: Block in Virtex-5 FPGAs* [online]. February 2010 [cit. 2014-01-23], UG200 (v1.8).
URL http://www.xilinx.com/support/documentation/user_guides/ug200.pdf

[26] Xilinx: *MicroBlaze Processor Reference Guide* [online]. 2012 [cit. 2014-01-21], UG081 (v14.1).
URL http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/mb_ref_guide.pdf

[27] Xilinx: *EDK – AXI Ethernet v3.01.a Support for Zynq Devices* [online]. May 2012 [cit. 2014-04-29].
URL http://www.xilinx.com/support/answers/47770.htm

[28] Xilinx: *Virtex-6 Family Overview* [online]. January 2012 [cit. 2014-04-29], DS150 (v2.4).
URL
pdf.datasheetarchive.com/indexerfiles/Datasheets-EC3/DSAQ00405383.pdf

[29] Xilinx: *MicroBlaze Soft Processor Core* [online]. 2013 [cit. 2014-01-17].
URL http://www.xilinx.com/tools/microblaze.htm

[30] Xilinx: *Zynq-7000 All Programmable SoC Overview* [online]. December 2013 [cit. 2014-01-23], DS190 (v1.6).
URL http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[31] Xilinx: *7 Series DSP48E1 Slice: User manual* [online]. August 2013 [cit. 2014-05-06], v1.6.
URL http://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf

[32] Xilinx: *Zynq-7000 Silicon Devices* [online]. 2014 [cit. 2014-01-23].
URL http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/silicon-devices/index.htm

[33] Xilinx: *Zynq-7000 All Programmable SoC: Technical Reference Manual* [online]. February 2014 [cit. 2014-02-23], UG585 (v1.7).
URL http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf

[34] Xilinx: *Tri-Mode Ethernet Media Access Controller (TEMAC)* [online]. 2014 [cit. 2014-04-29].
URL http://www.xilinx.com/products/intellectual-property/TEMAC.htm

[35] Xilinx: *Xilinx Tri-Mode Ethernet MAC Offerings and Software Requirements* [online]. 2014 [cit. 2014-04-29].
URL http://www.xilinx.com/ipcenter/temac/temac_offerings_and_system_requirements.htm

[36] Xilinx: *Vivado Design Suite* [online]. 2014 [cit. 2014-04-30].
URL http://www.xilinx.com/products/design-tools/vivado/

[37] Xilinx: *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* [online]. January 2014 [cit. 2014-04-30], UG973 (v2013.4).
URL http://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_4/ug973-vivado-release-notes-install-license.pdf

Příloha A

Obsah CD

CD obsahuje následující adresáře:

- `binary`: hotové binární soubory firmwaru a operačního systému Linux.
- `Buildroot`: zdrojové kódy přednastaveného nástroje Buildroot.
- `Vivado`: projekt pro vývojový nástroj Vivado.
- `usonda_core`: zdrojové kódy komponenty USONDA_CORE s metadaty pro integraci do nástroje EDK a vývojového nástroje Vivado.

Nezbytné kroky pro testování:

1. Vytvoříme TFTP server kam zkopírujeme soubory operačního systému Linux (adresář `binary/Linux/`).
2. Spustíme desku a zastavíme automatické bootování.
3. Naprogramujeme FPGA (souborem z adresáře `binary/firmware/`).
4. Spustíme bootování operačního systému Linux a nakonfigurujeme platformu podle návodů v kapitolách 5.7 a 6.3.