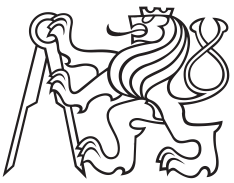
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra mikroelektroniky

Internetové rozhraní využívající FPGA

Internet interface using FPGA

Vojtěch Kouřil
Otevřené elektronické systémy

Praha, 2017
Vedoucí práce: prof. Ing. Pavel Hazdra, CSc.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kouřil** Jméno: **Vojtěch** Osobní číslo: **434893**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Otevřené elektronické systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Internetové rozhraní využívající FPGA

Název bakalářské práce anglicky:

FPGA Internet Interface

Pokyny pro vypracování:

1. Seznamte se s vnitřní architekturou FPGA řady SPARTAN-3E firmy Xilinx a metodikou jejich návrhu v jazyce VHSIC HDL.
2. Seznamte se s principy internetové komunikace a jejich hardwarové implementace.
3. Navrhněte a implementujte systém umožňující zasílání dat mezi FPGA a osobním počítačem. Pro implementaci využijte FPGA řady SPARTAN-3E a návrhový systém ISE a EDK.

Seznam doporučené literatury:

[1] P. J. Ashenden, The Designer's Guide to VHDL, Morgan Kaufmann, 2008
[2] P. Chu, RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability, Wiley, 2006
[3] Spartan-3 Generation FPGA User Guide, Dostupné z https://www.xilinx.com/support/documentation/user_guides/ug331.pdf
[4] LogiCORE IP XPS Ethernet Lite Media Access Controller, Dostupné z https://www.xilinx.com/support/documentation/ip_documentation/xps_ethernetlite.pdf

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. Ing. Pavel Hazdra CSc., katedra mikroelektroniky FEL

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

Datum zadání bakalářské práce: **17.02.2017** Termín odevzdání bakalářské práce: **26.05.2017**

Platnost zadání bakalářské práce: **31.08.2018**

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání | Podpis studenta

Poděkování / Prohlášení

Na tomto místě bych chtěl poděkovat vedoucímu bakalářské práce prof. Ing. Pavlu Hazdrovi CSc. za odborné vedení, cenné připomínky a podnětné návrhy při vypracovávání práce. Dále bych chtěl poděkovat panu Petru Jaškovi za zapůjčení senzoru a poskytnutí semestrální práce. Na posledním místě bych chtěl poděkovat rodině za její cennou podporu během mého studia.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 24. května 2017

Abstrakt / Abstract

Cílem této práce je využití pole FPGA k přenosu dat po síti. První část práce pojednává o hradlových polích FPGA, jejich stavbě a využití. Věnuje se také návrhu v poli FPGA a pojednává o návrhovém jazyce VHDL. Práce obsahuje popis technologie Ethernet a používaných síťových komunikačních protokolů.

V praktické části je realizováno propojení teplotního senzoru DS18B20 s FPGA vývojovou deskou Spartan-3E. Hradlové pole FPGA je využito ke zpracování naměřených dat a s využitím implementovaného mikroprocesorového jádra jsou data odesílána přes Ethernetový port do počítače.

The aim of this thesis is to use the FPGA for data network transfer. The first part of the thesis focuses on the FPGA gate arrays, their architecture and utilization. It also deals with the FPGA design and the VHDL design language. The thesis contains description of Ethernet technology and commonly used network communications protocols.

In the practical section of the thesis a connection is implemented between the digital thermometer DS18B20 and the FPGA development board Spartan-3E. The FPGA gate array is used for processing of the measured data, and using the implemented microprocessor core, data is sent over the Ethernet port to the computer.

Obsah /

Kapitola 1
Úvod

Technologie FPGA je na světě již něco přes 30 let, a za tu dobu se jí podařilo prokázat své kvality, výhody a uplatnit se na trhu. V současnosti je to technologie, která jde ruku v ruce s dalšími oblastmi hardwarového vývoje, hustotou integrace si nezadá s procesory, rychlostí s mikrokontrolery a díky své univerzálnosti dokáže nahradit jakýkoliv digitální obvod, nebo přímo celý počítač. FPGA je dnes dostupná i pro běžného uživatele, a má velké uplatnění v oblastech průmyslu, vzdělávání, lékařství či zpracování signálu.

Internet věcí (anglicky Internet of Things, IoT) je bezesporu jedním z technologických fenoménů dnešní doby. Představuje koncept, kde každé elektronické zařízení je součástí internetové sítě a je propojeno s dalšími elektronickými prvky v síti. Tato zařízení (dokonce se jim také říká „chytrá zařízení“) tak mohou mezi sebou komunikovat bez použití společného řídicího prvku, např. počítače.

Tato bakalářská práce se zabývá návrhem internetového rozhraní, implementovaného v hradlovém poli FPGA. Snaží se tak využít desku s hradlovým polem jako prvek, schopný komunikovat v rámci internetové sítě s dalšími zařízeními. Jako příklad je uveden digitální teploměr, který je v závěru práce propojen s internetovým rozhraním desky, a díky tomu je schopen komunikace po internetové síti.

Nejdříve se práce zabývá obvodem FPGA, pojednává o jeho vývoji a předchůdcích, dále se věnuje jeho stavbě a architektuře. Následující část obsahuje strukturu návrhu FPGA a popisuje hlavní body, které při návrhu obvodu hrají roli. Také ukazuje návrhové prostředí, ve kterém celý návrh probíhá, jeho části, funkce a k čemu slouží. Třetí kapitola pak obsahuje stručný nástin návrhového jazyka VHDL, uvádí jeho strukturu a syntaxi.

Čtvrtá kapitola se věnuje internetovému rozhraní, s důrazem na použité technologie. Popisuje technologii Ethernet, její využití a strukturu, a také uvádí funkce protokolů IP a UDP.

Praktická část se věnuje samotné realizaci projektu, návrhu internetového rozhraní a propojení digitálního teploměru se síťovým rozhraním. Práce je zakončena zhodnocením a závěrem, kde dojde k posouzení celého projektu a vyhodnocení dosažených cílů.

Kapitola 2
FPGA

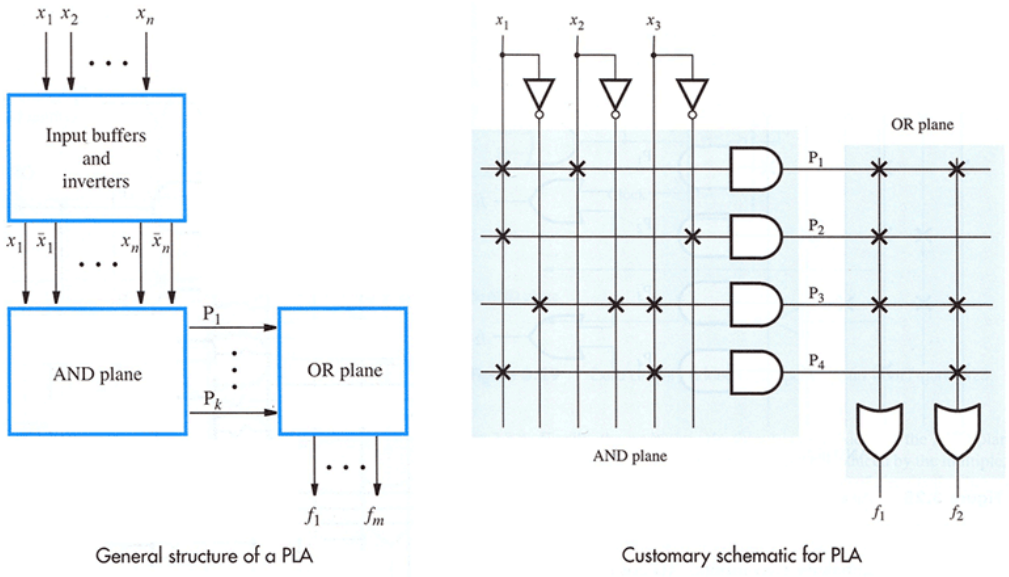
Obvod FPGA (Field-Programmable Gate Array, česky Programovatelné hradlové pole) je programovatelné zařízení (Programmable Logic Device, PLD), které je tvořeno pravidelnou sítí propojených logických bloků. Systém je uzpůsoben k tomu, aby měl uživatel možnost systém konfigurovat, a tím realizovat různé logické funkce.

Obvody FPGA slouží hlavně k jednoduchému a rychlému návrhu logických obvodů, a zároveň umožňují uživateli ověřit a dále upravovat svůj návrh bez nutnosti vyrábět nákladné prototypy návrhu.

2.1 Historie

2.1.1 PLA

Prvním pokusem o programovatelné logické obvody byl PLA (Programmable Logic Array) vyvinutý na začátku 70.let firmou Texas Instruments. [1] Systém vycházel z pamětí PROM (Programmable read-only memory), které umožňovaly uživateli zapsat do paměti informaci pomocí „přepálení“ vybraných propojek. Tento princip byl poté použit v obvodu PLA, jehož základ tvoří soustava křížově propojených logických hradel AND a OR. Přepálením vybraných spojů bylo možné implementovat do obvodu logickou funkci v její kanonické formě.



Obrázek 2.1. Struktura pole PLA, [2]

Koncem 70. let se začaly místo přepálitelných propojek používat tranzistory s plovoucím hradlem. Výhodou použití tranzistorů v programovatelném logickém poli byla možnost přeprogramování pole pod silným zdrojem UV záření, které umožnilo vybíjení

plovoucích hradel, a obvod byl připraven k zápisu nové funkce. Tato procedura však byla časově náročná, proces „mazání“ trval obvykle v řádu desítek minut až hodin.

2.1.2 PAL, GAL

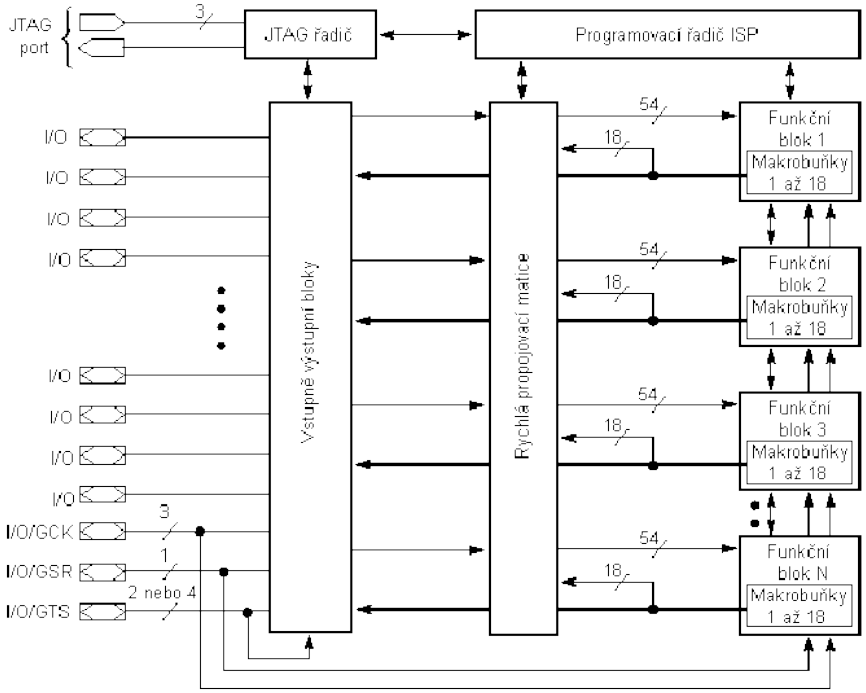
Koncem 70. let se také objevily pokusy o jednodušší implementaci původního systému PLA. [3] Výsledkem byl obvod PAL (Programmable Array Logic), který využíval statické propojení hradel OR a programovatelné propojení hradel AND. Došlo tak k výraznému zjednodušení struktury, snížení počtu spojů a to umožnilo konstrukci menších a levnějších obvodů. V některých odvětvích nacházejí obvody PAL stále uplatnění.

Další pokrok v oblasti programovatelných hradlových polí přišel v průběhu 80. let s nástupem obvodů GAL (Generic Array Logic), které výrazně zjednodušily proces mazání. Díky použití technologie EEPROM bylo možné smazat strukturu přímo elektrickou cestou, a znovu naprogramovat za použití programovací jednotky (programátoru).

2.1.3 CPLD

Se zvyšujícími se nároky na komplexnost zařízení rostla potřeba po stále větších a větších AND-OR sítích. V polovině 80. let byl uveden nový typ systému - CPLD (Complex Programmable Logic Device). Hlavní stavební jednotkou tohoto zařízení je makrobuňka, která každá obsahuje vlastní AND-OR síť. Bloky těchto makrobuněk jsou pak propojeny sítí vodičů, každý blok makrobuněk má také přidělený vlastní obvod, který obsluhuje vstupní a výstupní porty. [4]

Tento systém umožnil segmentaci návrhu do více funkcí, a implementaci návrhů, které by jinak v běžném PAL obvodu vyžadovaly enormní množství použitých hradel. Vzhledem ke zvyšujícímu počtu použitých portů již nebylo možné použít klasické programovací jednotky. Do čipu se tedy začalo implementovat speciální rozhraní JTAG, sloužící přímo k programování obvodu.



Obrázek 2.2. Struktura CPLD (obvod XC9500), [5]

2.1.4 FPGA

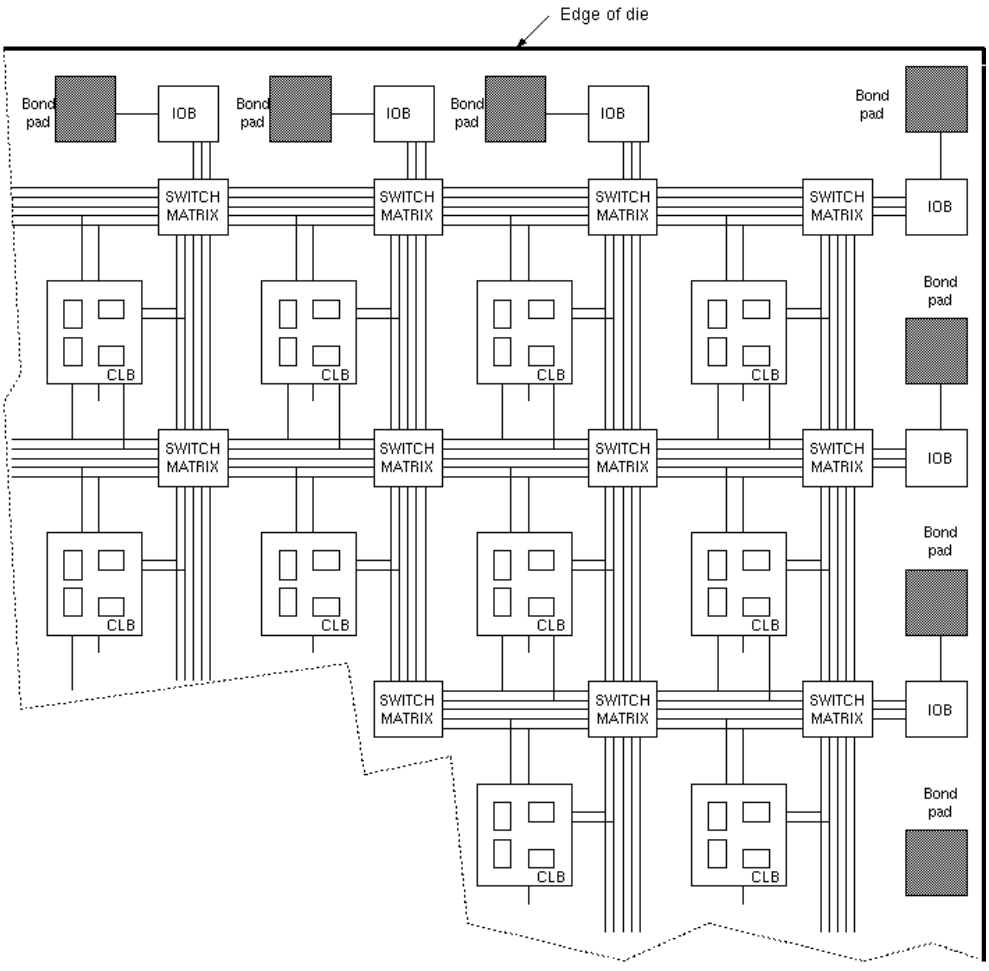
Za první FPGA se považuje obvod XC2064 vyrobený firmou Xilinx v roce 1985. Obsahoval 64 logických bloků (CLB), celkem obsahoval 800 logických hradel. [7]

Technologii FPGA přejali i další výrobci (Altera, Actel), a v 90. letech zažily obvody FPGA největší rozmach, koncem milénia dosahovaly kapacit 1 milionu hradel.

V posledních letech se objevují snahy o implementaci systému FPGA do klasického procesorového systému, a vytvoření jednotného systému, který by umožnil zpracování komplexních výpočetních operací. Příkladem můžou být softwarové procesory Microblaze/Picoblaze určené k implementaci do hradlového pole, nebo např. desky Zynq, obsahující jak procesor, tak obvod FPGA.

2.2 Struktura FPGA

Narozdíl od svých předchůdců FPGA nevyužívá síť AND-OR hradel - místo nich je hlavní stavební jednotkou *konfigurovatelný logický blok* (CLB). Dále obvod obsahuje síť propojovacích vodičů a spínačů sdružených do *sběrnic*. Sběrnice vedou v horizontálním a vertikálním směru, mezi nimi se nacházejí logické bloky. V oblasti křížení sběrnic je *spínačový blok*, který zajišťuje propojení jednotlivých sběrnic. Celý systém je pak propojen s vstupními/výstupními piny pomocí *I/O bloků*. [6] Celé schéma je na obrázku 2.3.



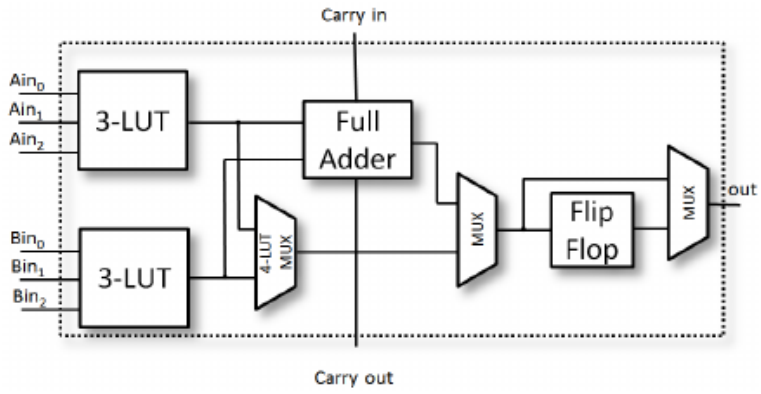
Obrázek 2.3. Struktura FPGA, [10]

2.2.1 Logické bloky

Logický blok je část obvodu, ve které jsou implementovány logické funkce. Každý logický blok se dále dělí na jednotky zvané *slice*, které jsou v bloku vzájemně nezávislé.

Slice je tvořen jednou nebo více *look-up tabulkami* (LUT), sčítačkou a výstupním registrem. Tabulka obsahuje konfigurovatelné buňky, ve kterých jsou uložené bitové hodnoty (1 nebo 0). Pro tabulku obsahující n vstupů bude potřeba 2^n buněk. Počet buněk tak odpovídá počtu řádků pravdivostní tabulky libovolné logické funkce o n proměnných.

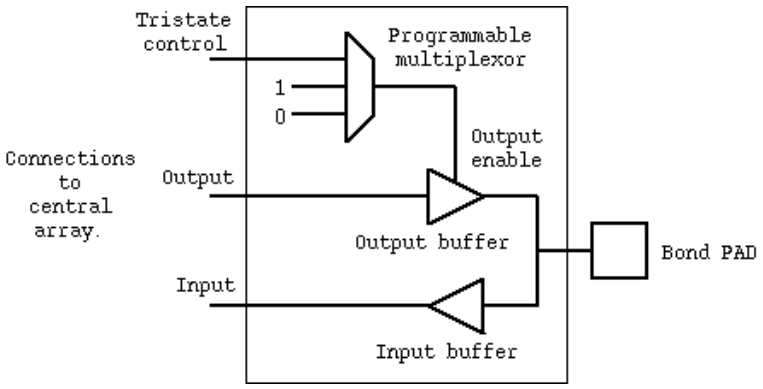
Výstupy z tabulek pak může být buďto sečteny, nebo mohou formovat vícebitovou LUT tabulku pomocí multiplexeru. Výstupní registr typu Flip-Flop se stará o sekvenční časování výstupu.



Obrázek 2.4. Struktura logického bloku FPGA - Slice, [9]

2.2.2 I/O bloky

Základní funkcí I/O bloku je propojení sběrnice se vstupními a výstupními piny. Základ bloku tvoří vstupní a výstupní registry, často typu flip-flop - umožňují tak přenést signál na pin bez zbytečného zpoždění. Ty jsou aktivovány Tri-state multiplexerem, který určuje, zda přiřazený pin bude přijímat či odesílat signál. Často také umožňuje určit rychlost náběžné hrany (*slew rate*), nebo aktivní oblast signálu (active low / active high). Zjednodušené schéma je na obrázku 2.5.



Obrázek 2.5. Zjednodušená struktura I/O bloku FPGA, [10]

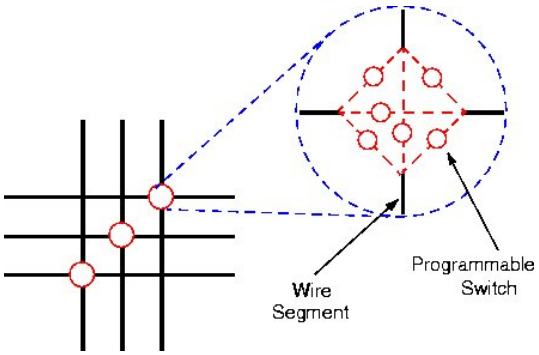
2.2.3 Propojovací prvky

Většinu plochy čipu zaujímají propojovací vodiče a spínače. Se vzrůstající délkou a počtem zapojených prvků roste odpor, kapacita a zpoždění signálu - je zde tedy snaha při

návrhu použít co nejkratší spojení bloků. Některé vodiče (typicky hodinový signál) však potřebují přistupovat k různě vzdáleným blokům stejně, typicky se proto propojovací vodiče dělí do více kategorií.

Globální vodiče poskytují univerzální propojení logických bloků napříč celým obvodem. Tyto vodiče jsou nejčastějším propojovacím prvkem, nevýhodou je větší zpoždění průchozího signálu. Lokální vodiče jsou určeny k propojení sousedních bloků, jsou krátké a mají malé zpoždění. Využívají se především k propojení více logických bloků do samostatného funkčního celku. Speciální vodiče zajišťují přístup hodinového, resetového a tri-state signálu do všech logických bloků, a jsou optimalizovány tak, aby poskytovaly co nejmenší zpoždění. Je jich však omezené množství, a nelze je použít k vedení běžných signálů.

Přepínačové matice pak slouží ke směrování signálu. Typický příklad propojení je na obrázku 2.6. Propojení je samozřejmě přístupné pouze mezi vodiči stejné kategorie.



Obrázek 2.6. Spínačová matice, [6]

2.2.4 Hardwarové prvky

Obvod FPGA může navíc také obsahovat samostatné jednoúčelové hardwarové bloky, které již nelze konfigurovat (či jen v omezené míře). Příkladem mohou být paměti RAM, aritmetické obvody, mikroprocesor, řadiče rozhraní, PLL (phase-locked loop) obvody pro správu hodinového signálu, a jiné. Výhodou jejich použití je úspora prostředků pro návrh ostatních částí obvodů a často poskytují větší výkonnost než jejich implementace pomocí logických bloků.

2.3 Spartan-3E FPGA

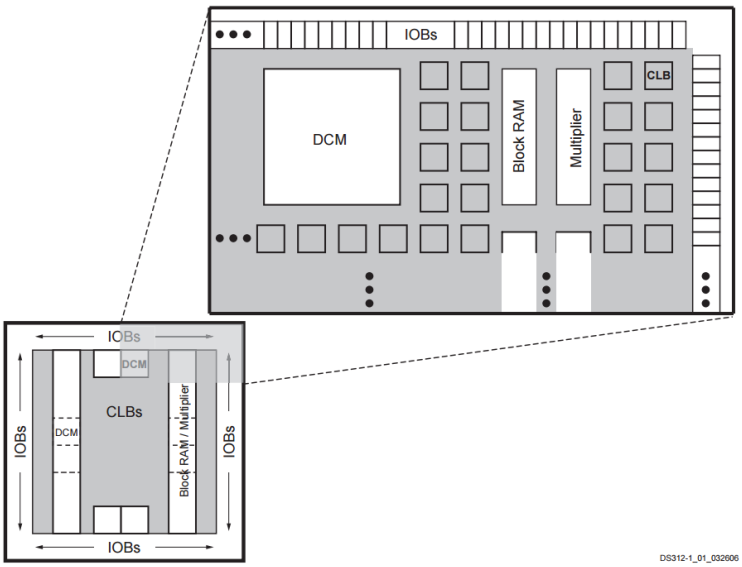
Spartan-3E Starter Kit Board je vývojová deska od firmy Xilinx vyráběná od roku 2006. [12] Srdcem desky je FPGA hradlové pole Spartan-3E XC3S500E, patřící do produktové řady Spartan-3. Hradlové pole je vyráběno 90nm technologií a obsahuje 500 000 hradel.

Základní architektura pole obsahuje 5 částí:

1) **Konfigurovatelné logické bloky (CLB)** obsahují LUT tabulky a registry. Jsou základní jednotkou pro implementaci logického obvodu.
2) **Input/output bloky (IOB)** poskytují přenos mezi interní logickou sítí a I/O piny. Umožňují třístavovou i obousměrnou komunikaci a umí komunikovat na vícero technologických standartech.
3) **Bloková RAM** umožňuje uchovávat programová data ve 18 Kbit blocích.
4) **Multiplikátory** poskytující násobení až 18-bitových čísel.
5) **Správce hodinových signálů (DCM)** zařizuje správné časování všech bloků a umožňuje dále pracovat s hodinovým signálem.

	Spartan-3E FPGA Family				
	XC3S100E	XC3S250E	XC3S500E	XC3S1200E	XC3S1600E
System Gates	100K	250K	500K	1,200K	1,600K
Logic Cells	2,160	5,508	10,476	19,512	33,192
Block RAM Bits	72K	216K	360K	504K	648K
Distributed RAM Bits	15K	38K	73K	136K	231K
DCMs	2	4	4	8	8
Multipliers	4	12	20	28	36
I/O Standards	18	18	18	18	18
Max Single Ended I/O	108	172	232	304	376
Max Differential I/O Pairs	40	68	92	124	156

Obrázek 2.7. Specifikace hradlových polí řady Spartan-3E, [12]



Obrázek 2.8. Architektura hradlového pole Spartan-3E, [12]

Vývojová deska dále kromě Spartan-3E FPGA obsahuje: [11]

- Konfigurační Flash paměť o kapacitě 4 MBit
- CPLD obvod Xilinx XC2C64A CoolRunner™
- 64MB DDR SDRAM, 100MHz
- 16MB StrataFlash paměť
- 16MBit serial Flash paměť
- LCD obrazovka
- PS/2 port
- VGA port
- 10/100 Ethernet PHY obvod LAN83C185
- 2x RS-232 port
- USB debug interface
- 50 MHz oscilátor
- 6-pinové expanzní konektory
- 8x LED
- Rotační spínač
- 4x přepínač
- 4x tlačítkový spínač

2.4 Metodika návrhu FPGA

Technika návrhu programovatelných logických obvodů se velmi podobá postupu návrhu klasických digitálních obvodů. Na začátku návrhu stojí **stanovení cíle**, specifikace požadavků a návrh základní architektury systému. V tomto kroku je také potřeba zvážit, jestli obvod, který máme k dispozici, má pro daný úkol dostatečné prostředky.

Dalším krokem je převod návrhu do **hardwarového programovacího jazyka**. V současnosti nejpoužívanějšími jazyky jsou VHDL a Verilog. Více informací o jazyce VHDL se nachází v kapitole 3. Alternativní cestou může být použití stavových diagramů či blokových schémat návrhu.

Následuje **simulace** systému, která je provedena přímo nad návrhem. Celou simulaci má na starosti soubor **Testbench**, ve kterém je stanovená metodika testování, testované komponenty, vektory vstupních proměnných, mohou obsahovat také procedury k vyhodnocování výsledků simulace. Testbench může být vytvořen přímo uživatelem, nebo vygenerován softwarovým nástrojem. Následně je provedena simulace, která umožní návrháři ověřit správnou funkčnost návrhu.

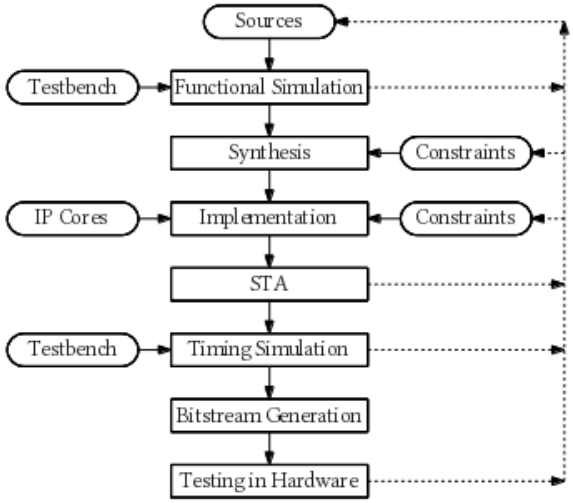
Následují kroky **syntézy** a **implementace**. Proces syntézy převede návrh do tzv. *netlistu*, který již popisuje návrh pomocí logických bloků, tj. na úrovni tabulek LUT, registrů atd. Implementace pak zajistí správné rozmístění netlistu do fyzických bloků. Proces probíhá za využití dodatečných návrhových pravidel, stanovených v „souboru podmínek“ (*Constraints file*). Tento soubor stanovuje rozmístění výstupních a vstupních pinů, konfiguraci jejich vlastností, vlastnosti hodinových signálů a jiná návrhová pravidla.

Statická časová analýza (Static Time Analysis, STA) a **časová simulace** (Timing Simulation) jsou procesy, které ověřují správnou funkčnost návrhu z hlediska časování. Pro simulaci je opět nutno vytvořit soubor vstupních stavů (testbench), případně jej vygenerovat softwarem.

V dalším kroku převede generátor bitstreamu odladěný návrh do souboru již použitelného pro naprogramování obvodu. **Bitstream** je soubor, který obsahuje konfigurační informace obvodu, a přesně určuje podobu návrhu, v jaké je implementován.

Posledním krokem je **přenos bitstreamu do konfigurátoru** (obvykle přes rozhraní JTAG) a nakonfigurování návrhu do obvodu. Poté je možné ověřit správnost návrhu přímo na testovaném hardwaru.

Celý návrhový proces lze vidět na obrázku 2.9.

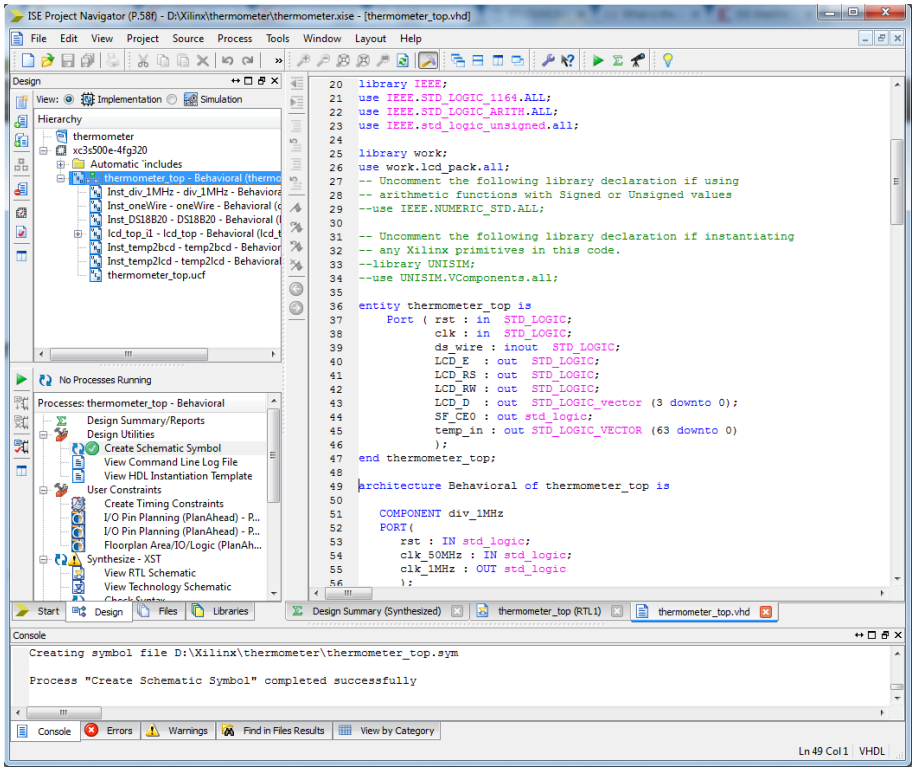


Obrázek 2.9. Návrhový proces programovatelného logického obvodu, [13]

2.5 Návrhový systém

2.5.1 Xilinx ISE

K návrhu projektu byl použit návrhový systém Xilinx ISE Design Suite ve verzi 14.5. Součástí systému jsou nástroje pro kompletní návrh obvodů CPLD a FPGA z portfolia firmy Xilinx. Obsahuje vývojové prostředí pro návrh v jazyce VHDL/Verilog, umožňuje testování, simulaci, syntézu i implementaci návrhu.



Obrázek 2.10. Návrhový systém Xilinx ISE - Project Navigator

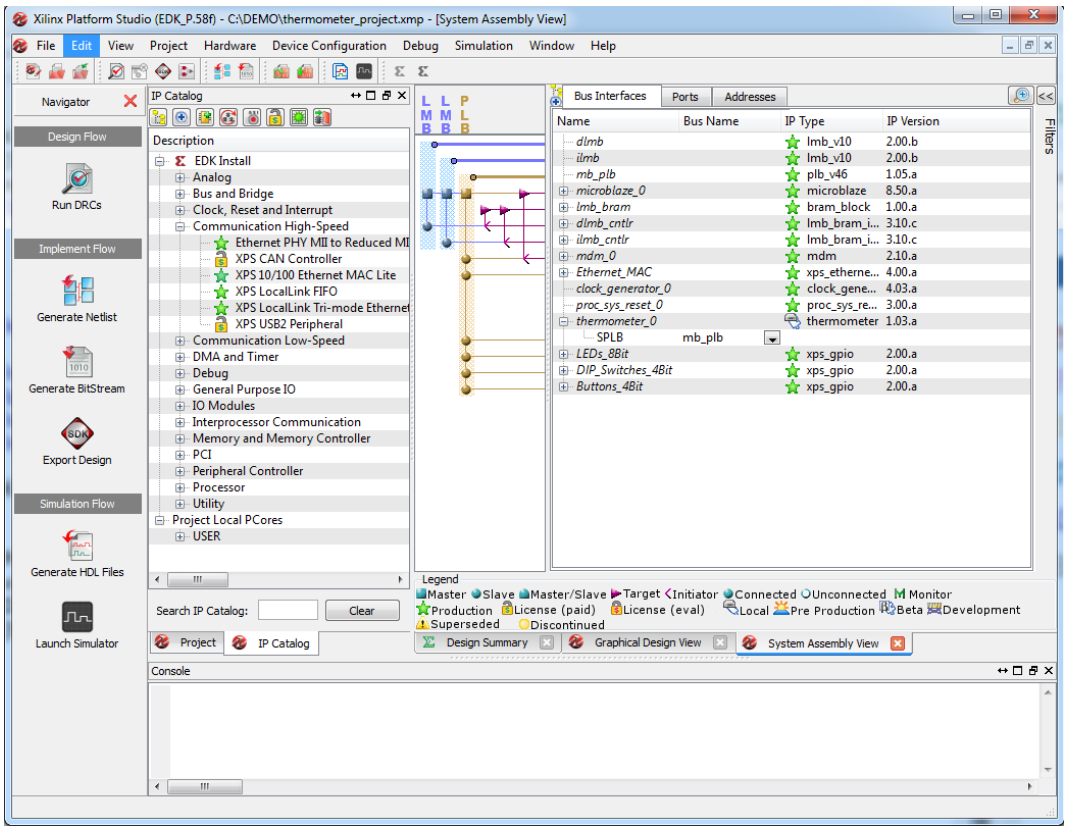
Součástí návrhového systému Xilinx ISE jsou dále nástroje pro vývoj vestavěných systémů (Embedded Development Kit, EDK). K těmto nástrojům patří programy Xilinx Platform Studio (XPS) a Software Development Kit (SDK).

2.5.2 Xilinx Platform Studio

Xilinx Platform Studio je program pro návrh vestavěného systému (embedded system). Základem systému je mikroprocesor, který může být buďto přímo na desce (hard-core), nebo může být navržen a implementován do hradlového pole (FPGA). Dále systém obsahuje sběrnici, je možné použít buďto sběrnici PLB (Processor Local Bus) nebo sběrnici AXI (Advanced Microcontroller Bus). Ke konfiguraci těchto základních nastavení komponent slouží průvodce BSB (Base System Builder). Program dále umožňuje navrhnutý systém upravovat, přiřazovat externí porty systému, měnit velikosti systémových pamětí a dále ladit systém.

Součástí EDK je také katalog IP[1] jader, což jsou již ozkoušené a odladěné návrhy, které je možno v systému použít. Pro použití jádra IP je nutné rezervovat dostatečně velkou systémovou paměť, přidělit systémovou adresu, připojit vstupy a výstupy IP jádra, a případně jej dále nakonfigurovat.

Po konfiguraci systému je návrh převeden do bitstreamu a připraven k implementaci na desku. Pro návrh systémových aplikací se systém ve formě bitstreamu importuje do vývojového prostředí SDK.



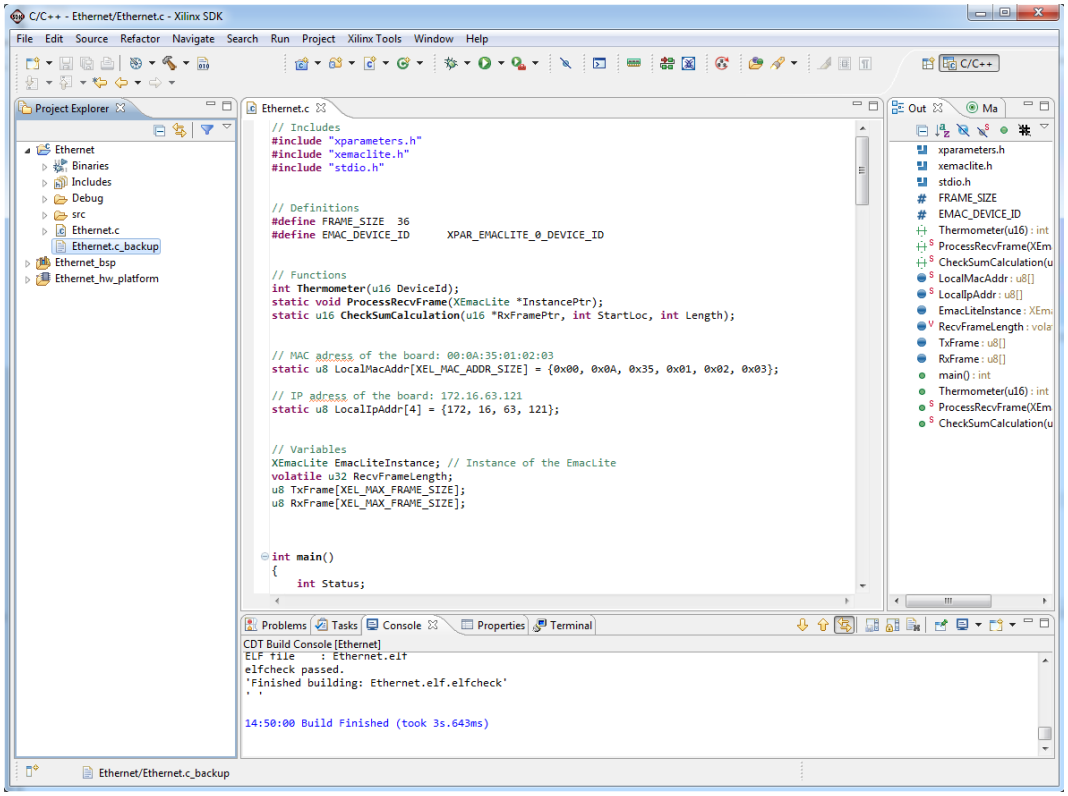
Obrázek 2.11. Hlavní okno programu Xilinx Platform Studio

2.5.3 Software Development Kit

[1] Nezaměňovat s IP - Internet Protocol

Software Development Kit je prostředí pro tvorbu aplikací pro vestavěné systémy. Obsahuje C/C++ kompilátor a debugger a poskytuje prostředí pro běh programů v přímém spojení s vývojovou deskou. Díky vestavěnému terminálu lze také propojit desku s PC sériovým kabelem UART a zaznamenávat výstup systému na desce.

Pro vyzkoušení běhu programu na desce je nejprve nutné na desku nahrát hardwarový návrh systému ve formě bitstreamu. Pro kontrolu se na desce rozsvítí oranžová LED (xc_done), když je přenos dokončen. Poté je možné přenést na desku aplikaci a spustit ji. Aplikace je ve formě spustitelného souboru s koncovkou *.elf*, který vznikne kompilací programového kódu v návrhovém prostředí.



Obrázek 2.12. Návrhové prostředí Software Development Kit

Kapitola 3
VHDL

Jazyk VHDL patří do skupiny programovacích jazyků HDL - Hardware Description Language. Jsou to jazyky používané k popisu hardwarového návrhu, a použitelné k jeho implementaci. Jazyk VHDL (Very-high-speed integrated circuit Hardware Description Language) byl původně vyvinut pro vojenské účely k návrhu číslicových systémů, později se stal standardem IEEE (IEEE 1087 z roku 1987). Výhodou tohoto jazyku je jeho univerzálnost a možnost navrhout stejný systém pro různé technologie. Jazyk je tak nezávislý na cílové platformě, záleží až na kompilátoru kódu, jak bude návrh implementován.

Návrh v jazyce VHDL probíhá zápisem do souborů s koncovkou *.vhd*. Každý VHDL soubor tvoří samostatný modul, který lze použít v dalších modulech.

Na začátku souboru jsou uvedeny použité knihovny a moduly. Soubor dále obsahuje dvě základní komponenty: **Entity** a **Architecture**.

3.1 Entity

Komponenta Entity definuje rozhraní popisovaného modulu, tj. vstupní/výstupní porty, vlastnosti a parametry portů, šířka sběrnic a jiné. V sekci **Port** jsou definovány názvy signálů, jejich mód a typ signálu. Název signálu musí být jednoslovný (bez mezer, lze použít podtržítko). Mód může nabývat hodnot:

- IN - vstupní port, nelze do něj zapisovat
- OUT - výstupní port
- INOUT - obousměrný port
- BUFFER - lze jej použít jako vstupní i výstupní port
- LINKAGE - signál s neznámým směrem toku

Typ signálu může být:

- bit - hodnoty 0 nebo 1
- bit_vector - bitový vektor, např. 8-bitový *bit_vector (0 to 7)*
- std_logic, std_logic_vector - nejpoužívanější typ, může nabývat těchto hodnot:
  - U - neinicializováno
  - X - zesílená neznámá
  - W - slabá neznámá
  - 0 - zesílená 0
  - 1 - zesílená 1
  - Z - vysoká impedance
  - L - slabá 0
  - H - slabá 1
  - „-“ - neurčená hodnota
- boolean - hodnoty True/False

- integer - celočíselná hodnota v rozsahu $-2^{32}, 2^{32}$
- real - reálné číslo
- character - znak ASCII
- string - vektor znaků ASCII

V sekci **Generic** jsou deklarovány parametry, které přímo nepředstavují signál. Může se jednat například o hodnoty zpoždění nebo o šířku sběrnice. Sekce obsahuje název parametru, jeho typ (viz Typ signálu v předchozí sekci), může mu také v této části přiřadit hodnotu.

Příklad deklarace 8-bitového multiplexeru 4 na 1:

```
entity mux4_to_1 is
    port (I0,I1,I2,I3: in std_logic_vector(7 downto 0);
        SEL: in std_logic_vector (1 downto 0);
        OUT1: out std_logic_vector(7 downto 0));
    generic (DATA_WIDTH : integer := 8);
end mux4_to_1;
```

3.2 Architecture

Komponenta Architecture popisuje vnitřní stavbu samotného bloku (modulu). Existují tři typy popisu architektury:

- Data Flow - popisuje funkci na úrovni logických operátorů
- Strukturální - popisuje zapojení základních logických bloků, např. hradel nebo flip-flopů
- Behavioral - popisuje algoritmicky chování modulu

První část komponenty Architecture je deklarační - deklarují se zde signály, konstanty a typy použité v těle architektury. Struktura je podobná jako v deklaraci portů a konstant komponenty Entity.

```
Architecture architecture_type of entity_name is
   --Deklarační část
begin
   --Příkazová část
end architecture_type;
```

Příkaz *begin* určuje konec deklarační části a začátek části příkazové. V příkazové části se nachází samotný popis systému.

3.2.1 Data flow

Architektura data-flow popisuje systém pomocí logických operátorů. Operátory VHDL mají vlastní hierarchii, která určuje jejich prioritu. Jejich přehled je uveden v tabulce 3.1. [1]

3.2.2 Structural

Ve strukturálním popisu je celý modul popsán na úrovni entit. Dané entity se pak vyvolávají a jejich porty se přiřazují pomocí příkazu *port map*. Syntaxe je následující:

[1] Popis funkcí jednotlivých operátorů je nad rámec této práce, lze jej nalézt např. na adrese `http://www.vhdl.renerta.com/source/vhd00047.htm`

Priorita	Název skupiny	Operátory
1 (nejvyšší)	Speciální operátory	** , abs , not
2	Násobící operátory	* , / , mod , rem
3	Sčítací operátory	+ , - , &
4	Posuvné operátory	sll , srl , sla , sra , rol , ror
5	Relační operátory	= , /= , ¡ , ¡= , ¿ , ¿=
6	Logické operátory	AND , OR , NAND , NOR , XOR , XNOR

Tabulka 3.1. Operátory VHDL dle priority

```
název_instance: název_entity    port map    (
                            port1 => signal1,
                            port2 => signal2,
                            …
                            portn => signaln);
```

Příklad návrhu hradla NAND pomocí strukturálního popisu:

```
architecture structural of NAND is
   signal ab_and : stdl_logic_vector(7 downto 0);
begin
   and_i: entity work.AND
      port map ( Input0=>A, Input1=>B, Output=>ab_and);
   not_i: entity work.NOT
      port map ( Input0=>ab_and, Output=>Y);
end structural;
```

3.2.3 Behavioral

Behaviorální návrh popisuje systém pomocí jednoho nebo více procesů. Chování systému je tak popisováno pomocí klasických algoritmických příkazů, avšak nepopisuje zapojení signálů a výběr logických bloků.

Proces je uvozen libovolným návěštím (labelem) procesu, po němž následuje příkaz *process*. Následuje seznam citlivých proměnných (sensitivity list), při jejichž změně má dojít ke spuštění procesu. V deklarační části jsou definovány použité konstanty a proměnné, které jsou uvnitř procesu použity. Příkaz *begin* určuje konec deklarační části a začátek části příkazové, která obstahuje jednotlivé algoritmické příkazy. Syntax procesu vypadá následovně:

```
name: process (sensitivity list)
  --declarations
begin
  --sequential statements
end process name;
```

Mezi základní algoritmické příkazy řadíme:

1) Podmíněné příkazy (případně s alternativou)

```
IF <condition> THEN <statements> [ELSIF <statements>] END IF;
```

2) Výběr více příkazů

```
CASE <condition> IS WHEN <value> => <statements>
   [WHEN <value> => <statements>]
   [WHEN OTHERS => <statements>]
END CASE;
```

3) Cyklus WHILE

```
WHILE <condition> LOOP <statements> END LOOP;
```

4) Cyklus FOR

```
FOR <range> LOOP <statements> END LOOP;
```

5) Příkaz WAIT

```
WAIT (ON <signals>) (UNTIL <statements>) (FOR <time>);
```

6) Příkazy přerušení

```
NEXT when <condition>; --restartuje proces
END when <condition>; --ukončí proces
```

Kapitola 4
Ethernet a sítě

4.1 Ethernet

Ethernet je technologie pro tvorbu počítačových sítí. [16] Operuje na úrovni linkové vrstvy, tj. 2. vrstvy RM-OSI modelu. Ethernet byl původně vyvinut v 70. letech v laboratořích Xerox PARC. Vývoj pokračoval v konsorciu tří společností DEC, Intel a Xerox. Ke standardizaci došlo roku 1983, kdy koncept převzal institut IEEE (The Institute of Electrical and Electronics Engineers, Inc.) a byl vydán pod standardem IEEE 802.3.

První standardizovaná verze poskytovala rychlost 10 MBit/s a využívala koaxiálních kabelů. Později byla také definována pro kroucenou dvojlinku a optická vlákna. Ethernet se stával stále populárnější technologií, a v 90. letech se stal nejrozšířenějším standardem pro lokální počítačové sítě. Došlo také k navýšení přenosové rychlosti nejdříve na 100 MBit/s a poté na 1 GBit/s. Jednogigabitové síťové Ethernet prvky jsou dnes běžným používaným standardem. Nejnovější standardy podporují rychlosti do 100 GBit/s a dodnes se vyvíjejí nové technologie.

4.1.1 Topologie

Prvním přenosovým médiem, které Ethernet využívalo, byl koaxiální kabel. Kabel fungoval jako sdílené médium, ke kterému se připojovaly počítače pomocí rozbočovačů ve tvaru T. Na koncích kabelu byly zapojené koncovky zvané terminátory. Jelikož po kabelu bylo v jednu chvíli možné uskutečňovat pouze jedno spojení, klíčová byla implementace protokolu předcházejícího kolizím mezi spojeními - CSMA/CD (viz sekce 4.1.2).

Koaxiální kabel umožňoval základní topologii typu Bus. Celková délka segmentu sítě však byla technologicky omezena na několik stovek metrů - pro vytvoření větších sítí tak bylo potřeba směrovacích prvků:

- *Repeater* přímo propojuje 2 různé síťové segmenty.
- *Hub* přímo propojuje více síťových segmentů.
- *Bridge* propojuje 2 různé síťové segmenty, avšak směruje pouze data z různých segmentů. Komunikaci v rámci jednoho segmentu dál nesměruje, může tak docházet k simultánním přenosům v rámci obou segmentů.
- *Switch* propojuje více síťových segmentů, avšak směruje data pouze do cílového segmentu.
- *Router* umožní propojit sítě pracující na různém standardu.

Nejběžnějším kabelem pro ethernetovou komunikaci je kroucená dvojlinka s koncovkou RJ-45. Tento kabel je výhradně dvojbodový - umožňuje propojit pouze dvě zařízení mezi sebou. Pro realizaci propojení třech a více zařízení je tudíž potřeba větvících směrovacích prvků - topologie Star, Tree. Dříve byly často používány huby, protože jsou stavebně jednodušší a tudíž nabízely nižší cenu. Narozdíl od switche/routeru totiž nemusí obsahovat žádné buffery na vstupech/výstupech. V současnosti tvoří buffery jen

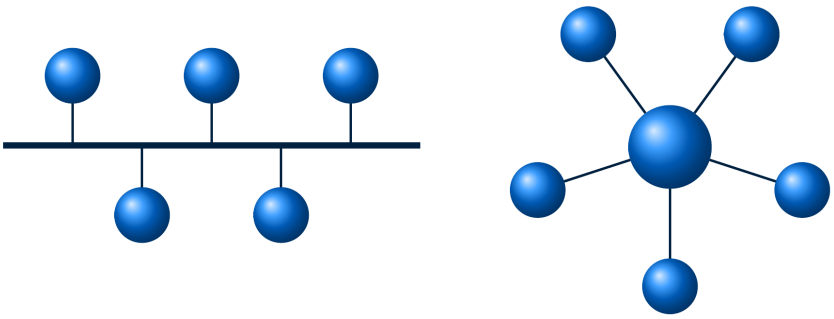
zlomovou část výrobní ceny, proto se využívají už téměř výhradně pouze switche, které nabízejí cílené směrování dat a oddělení přenosů z různých segmentů.

Klasická kroucená dvojlinka je tvořena pouze dvěma vodiči ve dvojité šroubovici. Vedený signál je diferenciální - je tvořen rozdílem napěťových úrovní linek. Nulový rozdíl úrovní značí nulu, nenulový rozdíl značí jednotku. Tento systém eliminuje chyby detekce z důvodu změny napěťové úrovně či z důvodu zašumění signálu.

Použitím dvou vodičů můžeme vést signál pouze jedním směrem, pokud chceme vést signál i směrem druhým, musíme počkat na dokončení prvního přenosu. Tento mód se nazývá *Half-duplex*, a pro správnou funkčnost je potřeba implementace antikolizního protokolu CSMA/CD.

Máme-li k dispozici alespoň 2 páry vodičů, můžeme je využít pro obousměrný provoz, kdy každému směru je přiřazen samostatný pár vodičů. Tento mód se nazývá *Full-duplex*, nevyžaduje použití antikolizního protokolu a logicky poskytuje vyšší propustnost než mód half-duplex.

Posledním používaným přenosovým médiem je optický kabel. Jeho výhodou je vysoká odolnost vůči elektromagnetickému rušení, nevýhodou je nutnost použít převodníky na elektrický signál. V optickém kabelu lze realizovat full-duplex mod pomocí dvou samostatných vláken, nebo použitím různých vlnových délek.



Obrázek 4.1. Topologie typu Bus (nalevo), Star (napravo)

4.1.2 CSMA/CD

Protokol CSMA/CD (Carrier-sense with Multiple Access and Collission Detection) [16] je protokol zajišťující bezkolizní prostředí pro přenos informací mezi síťovými stanicemi. Protokol slouží zejména pro přenos v módu half-duplex, nebo obecně pro přenos ve sdíleném médiu (např. koaxiální Ethernet kabel).

Síťové stanice se řídí těmito pravidly:

1. Každá stanice sleduje stav kanálu, jestli je aktuálně využíván k přenosu dat.
2. Pokud má stanice data k odeslání a kanál je volný, začne odesílat data. Pokud má stanice data k odeslání a kanál není volný, stanice čeká na uvolnění kanálu a pak ihned začne odesílat data.
3. Pokud stanice při odesílání dat zaregistruje kolizní signál, ihned přeruší odesílání, počká náhodnou dobu (back-off perioda) a vrátí se k bodu 2.

Protokol je také možné alterovat v bodu 2, kdy neumožníme stanici začít vysílat těsně poté, co byl dokončen předchozí přenos. Z velké části tím eliminujeme chybu, způsobenou současným začátkem přenosu více stanic, které mají data k odeslání a souběžně detekovaly konec přenosu. Řešením je umožnit stanici sledovat stav kanálu jen v určených časových okamžicích:

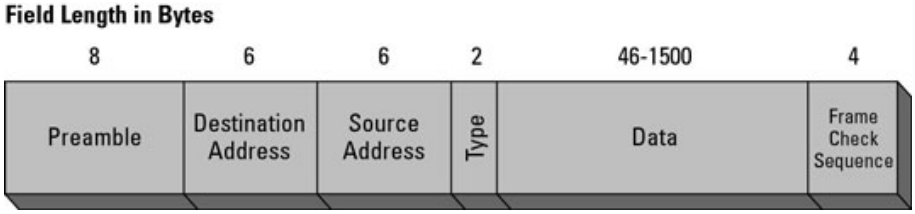
- Non-persistent CSMA sleduje kanál pouze v okamžicích oddělených náhodnou časovou periodou
- 1-persistent CSMA sleduje kanál neustále, a odesílá, jakmile je kanál uvolněný
- P-persistent CSMA sleduje kanál neustále, a jakmile je kanál uvolněný, odesílá s pravděpodobností P. Pokud data neodešle, počká náhodnou dobu a opět se pokouší o odeslání.

4.1.3 Struktura rámce

Struktura rámce je definována ve standardu IEEE 802.3: [17]

- Preambule: uvozující bitová sekvence, skládající se ze sedmi bloků (10101010) a bloku (10101011), celkem 64 bitů.
- MAC adresa destinace: adresa koncového zařízení, měla by být jedinečná, nezávisle na síti, 48 bitů.
- MAC adresa zdroje: adresa zdrojového zařízení, měla by být jedinečná, nezávisle na síti, 48 bitů.
- Délka/typ: 16 bitové číslo, určující délku rámce v bytech (oktetech). Maximální délka ethernetového rámce je 1500 bytů, větší hodnoty tohoto pole značí identifikátor protokolu.
- Data (Payload), maximálně 1500 bytů.
- Kontrolní součet (CRC, Cyclic Redundancy Check), 32 bitů.

Celkové schéma je uvedeno na obrázku 4.2.



Obrázek 4.2. Struktura ethernetového rámce, [18]

4.2 Internet Protocol

Internet Protocol (IP) je protokol obsluhující síťovou vrstvu, tj. 3. vrstvu modelu RM-OSI. Vytváří prostředí pro přenos datagramů (bloků uživatelských dat) přes různé sítě. Má na starosti směrování dat, je to nejnižší vrstva, která obsahuje kompletní trasu datagramu.

IP protokol byl vyvinut v 70. letech, v současnosti nejpoužívanější verze IPv4 byla specifikována roku 1981. Přenos datagramů je na principu *best effort*, tj. přenos probíhá s maximálním úsilím, ale bez záruky doručení. Datagramy se mohou ztrácet, nebo být doručeny v jiném pořadí, či jinak pozměněny. Pro kontrolu správnosti doručení a nenarušení dat obsahuje protokol CRC (cyklický kontrolní součet).

4.2.1 IP adresa

V současnosti se používají především dvě verze protokolu: IPv4 a IPv6. Největší rozdíl je ve velikosti adresového prostoru: IPv4 adresa je 32 bitová, nabízí tudíž 2^{32} adres, IPv6 adresa je 128 bitová, nabízí tak mnohem větší adresový prostor.

Každá IP adresa je abstraktní a nemá přímou vazbu na konkrétní zařízení. Pro překlad adresy je tak potřeba speciální protokol ARP (address resolution protocol), překlad je pak uchováván v tabulce směrovače (routeru).

IP adresa se dělí na síťovou část a relativní část adresy. Síťová část identifikuje síť a slouží k rozpoznání příslušnosti zařízení k síti. Velikost síťové části určuje množství propojitelných sítí. Relativní část adresy identifikuje zařízení v dané síti. Její velikost určuje kapacitu dané sítě (počet individuálních adres).

Původní koncept IPv4 představoval rozdělení adres do tříd:

- třída A: nejvyšší bit je 0, určena pro největší sítě, 7 bitová síťová část + 24 bitová relativní část
- třída B: nejvyšší bity jsou 1 a 0, 14 bitová síťová část + 16 bitová relativní část
- třída C: nejvyšší bity 1, 1, 0, určena pro lokální sítě, 21 bitová síťová část + 8 bitová relativní část

S rozšiřováním protokolu IP docházelo již v 90. letech k masivnímu úbytku IP adres, byla nutná revize systému rozdělení. Rozdělení do tříd způsobilo, že spousta adres zůstala nevyužitých. Pro řešení problému byl zaveden systém CIDR (1993), který zrušil použití tříd a umožnil volbu velikosti bloku. Tato vlastnost je zapsána v tzv. masce podsítě, která odpovídá velikosti síťové části adresy - počet bitů síťové části odpovídá počtu jedniček v masce podsítě, zbytek masky je vyplněn nulami. Rozměr masky je shodný s rozměrem adresy (pro IPv4 32 bitů). Alternativně se maska vyjadřuje číselnou hodnotou za lomítkem za adresou.

V současnosti jsou již všechny IPv4 adresy rozdělené a nepřiřazují se.

IP adresa se nejčastěji zapisuje ve čtyřech osmibitových číslech v dekadickém tvaru oddělených tečkou. Za lomítkem je pak velikost síťové masky. Příklad IP adresy:

```
192.168.0.1/8
```

Speciální adresy:

- 0.0.0.0/8 - toto zařízení
- 255.255.255.255/32 - všechna zařízení (broadcast)
- 10.0.0.0/8 - privátní adresy (lze použít v lokálních sítích)
- 127.0.0.0/8 - loopback
- 172.16.0.0/12 - privátní adresy
- 192.168.0.0/16 - privátní adresy

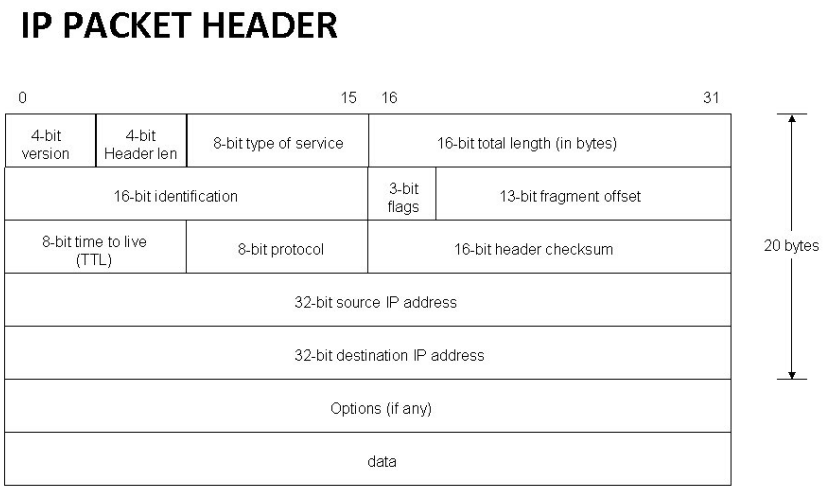
4.2.2 IPv4 rámec

Rámec IP datagramu musí obsahovat následující položky:

- Verze IP protokolu: 4 bitová hodnota, pro IPv4 je vždy hodnota 4.
- Velikost IP hlavičky: 4 bitová hodnota, typická délka je 20 bytů. Hodnota vyjadřuje počet čtyřbytových položek hlavičky - typicky se jedná o hodnotu 5.
- Typ služby: 8 bitová hodnota, určuje služby zajišťující kvalitu přenosu.
- Délka: 16 bitová hodnota, označuje délku datagramu v bytech.
- Identifikátor: 16 bitů, identifikuje datagram, unikátní číslo v paketu v každém spojení.
- Fragmentace: 3+13 bitů, řídí fragmentační proces.
- Time To Live: 8 bitů, určuje počet povolených průchodů směrovačem, chrání před cyklickým přeposíláním datagramu.

- Protokol: 8 bitů, určuje nadřazený protokol transportní vrstvy. Čísla protokolů obsahuje dokument RFC 1700 [20]
- Kontrolní součet hlavičky (CRC): 16 bitů, slouží k detekci narušení datagramu
- IP adresa odesílatele: 32 bitů
- IP adresa přijímače: 32 bitů

Na konci hlavičky IP lze uvést další volitelné položky. Celkovou strukturu popisuje obrázek 4.3.



Obrázek 4.3. Struktura IP rámce, [21]

4.3 UDP

UDP je protokol operující na 4. úrovni modelu RM-OSI, transportní vrstvě. Spolu s protokolem TCP patří do skupiny protokolů TCP/IP. Narozdíl od protokolu TCP poskytuje nespojovaný přenos, tj. komunikace probíhá bez sekvencí k navázání a ukončení spojení. Vyžaduje menší režii, není nutné potvrzovat přijetí paketu a neumožňuje sledovat stav přenosu.

Protokol UDP je tak využíván hlavně v případech, kdy nehraje roli posloupnost paketů nebo jejich návaznost. Výhodou jsou menší nároky na síť, tento protokol však neumí zaručit kvalitu služby.

4.3.1 Porty

Porty slouží k adresaci na transportní vrstvě. Jedná se o 16 bitové číslo určující bránu v zařízení, přes kterou mohou aplikace komunikovat. K jednomu portu může být připojena jedna nebo více aplikací, zpravidla však jsou přiřazeny porty aplikacím jednoho typu. Port je obousměrná brána, umožňuje jak posílat, tak přijímat data.

Porty se přiřazují v rozsahu 0 až 65535 a jejich rozdělení spravuje organizace IANA [22]. Obecně se dělí do 3 kategorií:

1) Systémové porty (0 až 1023) jsou rezervovány pro běžně používané síťové služby.

2) Registrované porty (1024 až 49151) jsou porty pro ostatní registrované služby.
3) Uživatelské porty (49152 až 65535) se nepřidělují, lze je použít pro vlastní aplikace.

Číslo portu se většinou uvádí v dekadickém tvaru, řadí se za IP adresu a je oddělené dvojtečkou.

4.3.2 UDP rámec

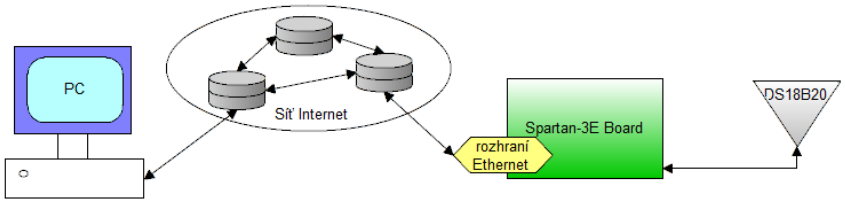
Rámec UDP datagramu musí obsahovat následující položky:

- Zdrojový port: 16 bitů,
- Cílový port: 16 bitů,
- Délka: 16 bitů, určuje počet bytů dat + UDP hlavičky,
- Kontrolní součet: 16 bitů, v datagramu UDP není nutný, v tom případě je tato položka nulová.

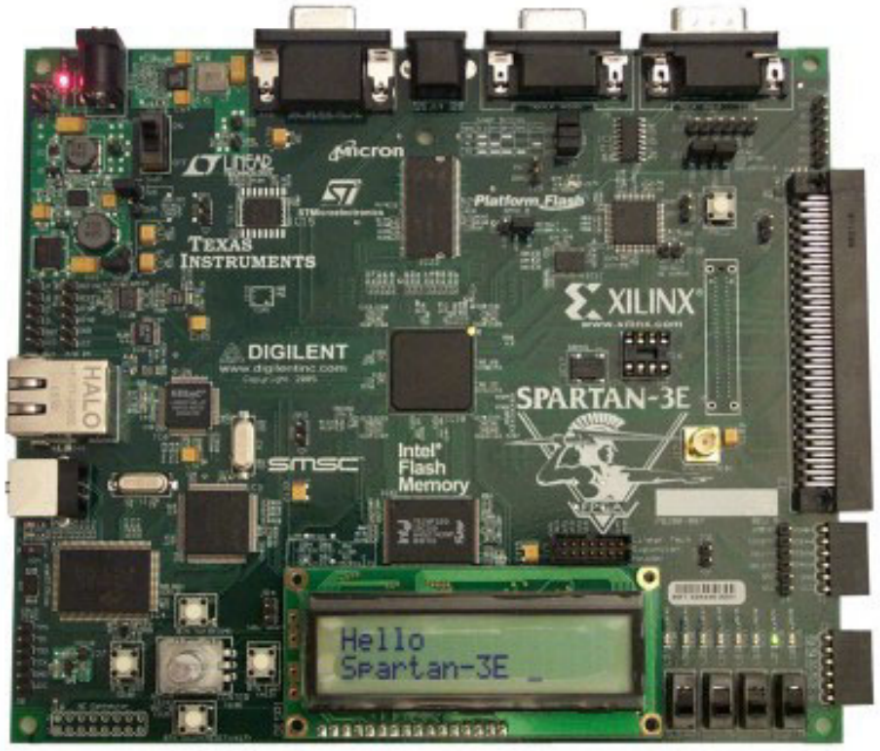
Kapitola 5
Návrh a realizace projektu

Cílem této práce bylo navrhnout a realizovat internetové rozhraní hradlového pole FPGA, a umožnit tak hradlovému poli komunikaci s počítačem, vzdáleným serverem nebo jiným zařízením v síti.

Projekt byl realizován na desce Spartan-3E FPGA Starter Kit Board s FPGA obvodem Spartan XC3S500E. Bližší popis použité desky je v kapitole 2.3. K propojení desky se sítí byl použit ovladač 10/100 Ethernet PHY na čipu LAN83C185. Jako zdrojová data jsou použity údaje z digitálního teploměru DS18B20, připojeného k desce pomocí expanzního portu J1.



Obrázek 5.1. Diagram celkového zapojení



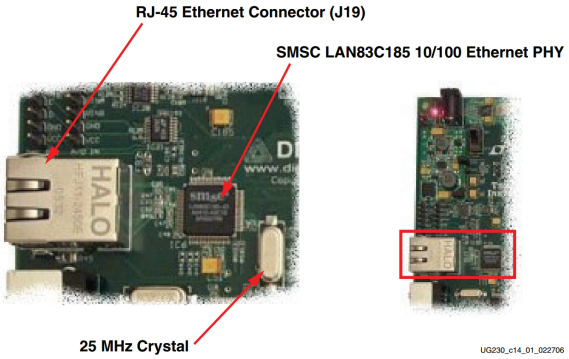
Obrázek 5.2. Deska Spartan-3E, [11]

5.1 Mikroprocesorový systém

5.1.1 Návrh systému

Hlavním úkolem nyní bylo zajistit přeposílání dat z FPGA desky do sítě přes ethernetový port. Na desce Spartan-3E se již nachází čip LAN83C185, který obhospodařuje pouze fyzickou vrstvu (PHY) ethernetového rozhraní. Veškeré další systémy zajišťující komunikaci na vyšších vrstvách tak musí být implementovány v FPGA.

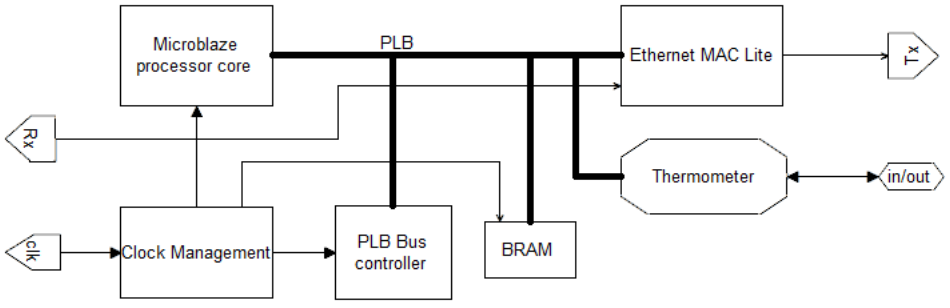
Vzhledem k poměrné složitosti návrhu vlastního ethernetového ovladače jsem se rozhodl použít IP jádro *Ethernet MAC Lite* [25], který se nachází v katalogu IP jader návrhového systému EDK.



Obrázek 5.3. Ethernetový PHY čip se síťovým konektorem RJ-45, [11]

Jádro IP však nelze použít pouze samostatně - je uzpůsobeno k systémovému řešení a poskytuje pouze připojení na sběrnici. Je tedy nutné navrhnout procesorový systém, k tomuto účelu byl použit program Xilinx Platform Studio. Pomocí průvodce BSB (Base System Builder) byl navržen systém na sběrnici PLB (Processor Local Bus) s těmito komponentami:

- Soft-core 32-bitový mikroprocesor MicroBlaze
- XPS 10/100 Ethernet MAC Lite
- Generátor hodinového signálu
- BRAM controller
- Thermometer - periferie teploměru
- ostatní komponenty zajišťující funkci sběrnice PLB



Obrázek 5.4. Základní schéma mikroprocesorového systému [1]

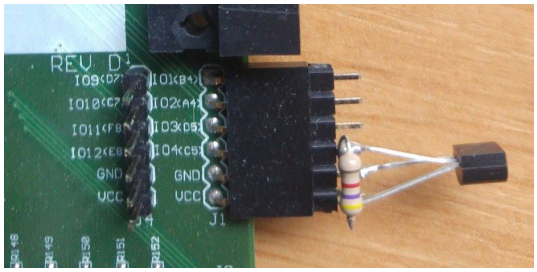
[1] Podrobné blokové schéma se nepodařilo z programu exportovat, a technologické schéma není kvůli rozměrům možné přiložit do dokumentu. Schéma lze zobrazit v projektu XPS, přiloženém na CD.

V následujících dvou sekcích bych se rád zmínil o digitálním teploměru DS18B20, a úpravě jeho návrhu do podoby systémové periferie.

5.1.2 Digitální teploměr DS18B20

Jako zdroj dat byl využit digitální teploměr DS18B20 [23]. Projekt využívá návrh zapojení teploměru a desky Spartan-3E, prezentovaný v rámci semestrální práce *Digitální teploměr s DS18B20* od Petra Jaška [24]. Archiv s touto prací je přiložen na CD.

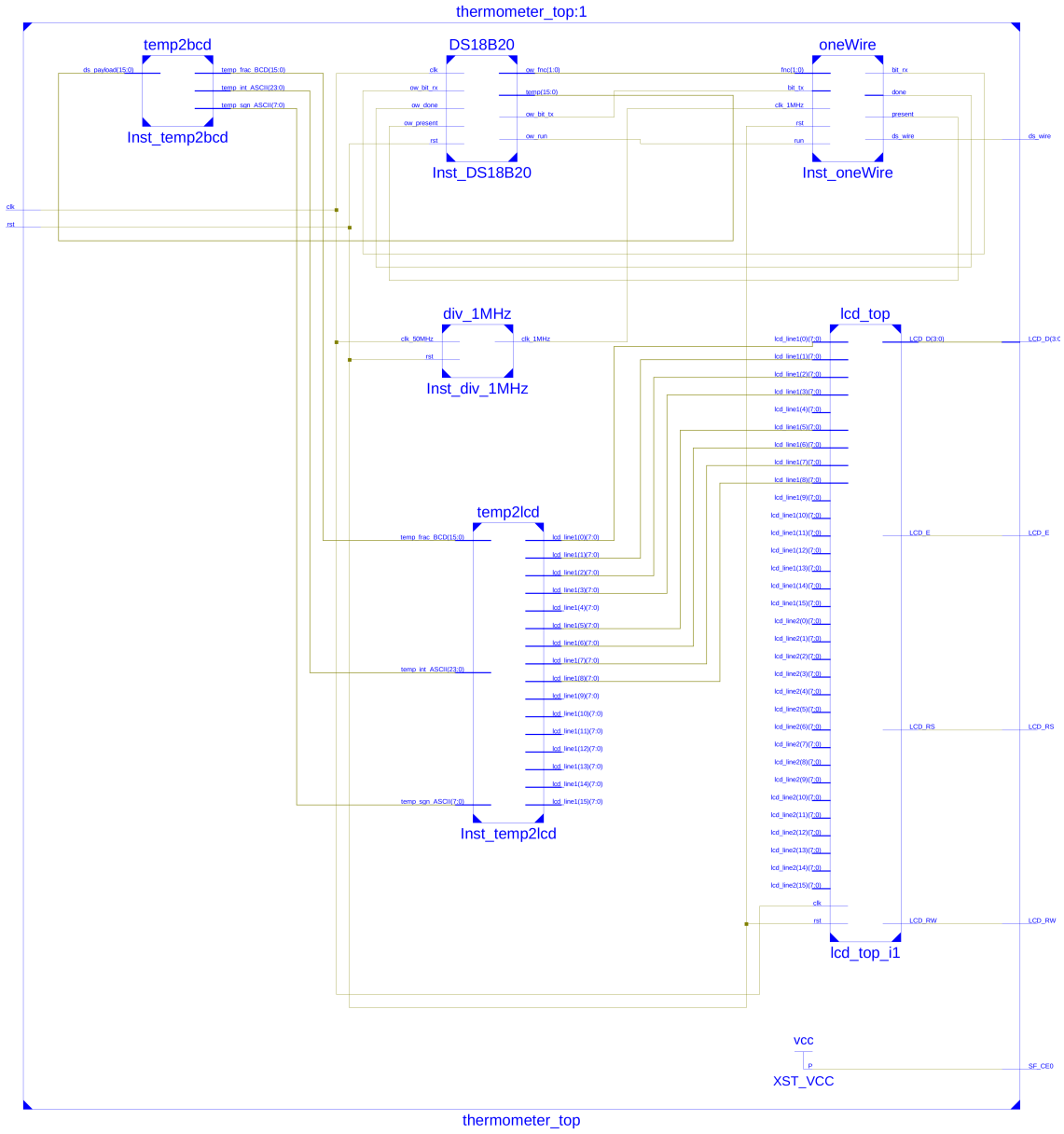
Teplotní senzor komunikuje s deskou přes sběrnici One-Wire a podporuje externí a „parazitní“ napájení. Shodně s uvedenou semestrální prací bylo zvoleno externí napájení senzoru, umožňující čtení stavu měření.



Obrázek 5.5. Zapojení teplotního senzoru, [24]

Původní projekt obsahuje výstup dat v podobě teploty zobrazované na LCD displeji desky. Se senzorem komunikuje přes samostatný One-Wire IN-OUT pin, jenž je obsluhován soustavou dvou stavových automatů *oneWire* a *DS18B20*. Bitový výstup je pak dále převáděn na dekadický formát a poté do ASCII znaků. Pro výstup na displej slouží moduly *temp2lcd* a *lcd_top*, které ASCII znaky řadí a zobrazují na displeji. O časování systému se stará modul *div_1MHz*, který vstupní hodinový signál o kmitočtu 50 MHz převádí na 1 MHz signál.

RTL (Register Transfer Level) schéma je zobrazeno na obrázku 5.6.



Obrázek 5.6. RTL schéma návrhu zapojení teploměru DS18B20, [24]

5.1.3 Návrh periferie

Dalším krokem bylo vytvoření systémové periferie, která do sběrnice posílá data z teploměru. Nová komponenta byla vytvořena přes nástroj Peripheral Wizard v programu XPS, její strukturu tvoří soubory:

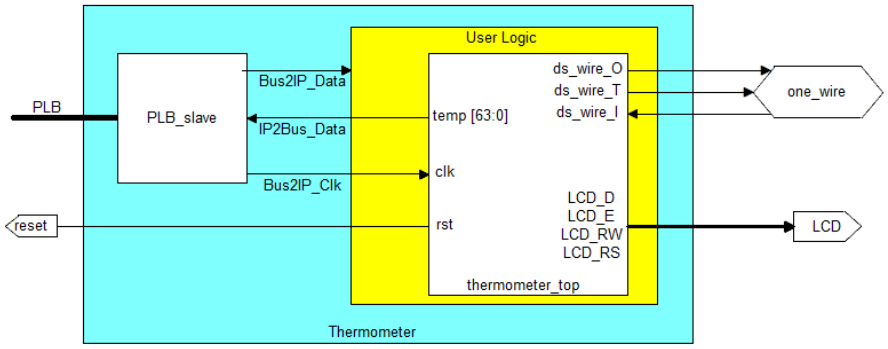
- Microprocessor Peripheral Description (.mpd) určující vlastnosti periferie a definující vstupní a výstupní porty
- Peripheral Analysis Order (.pao) obsahující seznam použitých zdrojových souborů
- Zdrojové soubory VHDL:
  - Top-level soubor VHDL
  - Soubor VHDL s uživatelským návrhem

Pro použití návrhu digitálního teploměru došlo k mírné úpravě původního návrhu. K bloku *temp2lcd* byl přidán 64-bitový výstup *temp[63:0]*, který obsahuje údaj o teplotě

v ASCII formátu. Tento výstup pak je veden do dvou 32-bitových registrů, ze kterých posléze může číst sběrnice PLB.

Dále musel být přepracován vstupně-výstupní One-Wire port. Podle poznámky v manuálu EDK [26] na straně 72 je použití signálu typu INOUT povoleno pouze v top-level bloku. Bylo proto nutné převést obousměrný signál na trojvodičové vedení (vstup, výstup, přepínač). Převod na INOUT signál probíhá teprve ve výstupním I/O bloku daného pinu.

Po navržení a vytvoření periferie ji bylo možno přidat do mikroprocesorového systému. Hodinový signál pro řízení teploměru je přiveden ze přímo ze sběrnice, ponechány byly výstupy na LCD displej. Pro indikaci změny teploty je vyveden signál na LED diodu. Reset teploměru je vyveden na samostatné tlačítko na desce (btn_east), reset procesorového systému je vyveden na tlačítko (btn_south).



Obrázek 5.7. Základní schéma periferie Thermometer

Kompletní diagram periferie je přiložen v příloze B.1.

5.1.4 Export návrhu

Následně byla přidána nově vytvořená periferie do návrhu systému, proběhlo připojení externích pinů a vygenerování systémové adresy periferie: 0xC4C00000.

Výsledná systémová konfigurace je uložena v souboru *system.mhs*, a obsahuje popis portů, rozhraní sběrnic a parametrů komponent. Součástí návrhu je také soubor *system.ucf* popisující propojení externích pinů s porty systému.

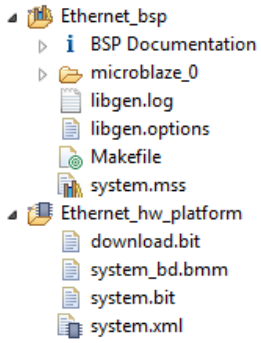
Posledním krokem byla syntéza návrhu, generace bitstreamu a export návrhu do prostředí SDK.

5.2 Software

5.2.1 Příprava

Po exportu návrhu do prostředí SDK byl vygenerován soubor *system.bit* a dokumentační soubor *system.xml*. Tyto soubory popisují hardwarovou platformu, na které je návrh postaven.

Pro tvorbu vlastní aplikace na mikroprocesorovém systému bylo nutné vytvořit tzv. Board Support Package [27] (BSP), který zastává funkci operačního systému. Poskytuje propojení funkcí mikroprocesoru a sběrnice, případně může obsahovat ovladače periferních zařízení.



Obrázek 5.8. Soubory BSP a HW platformy

5.2.2 Aplikace

Vlastní aplikace byla psaná v programovacím jazyce C, s použitím příkladů kódu k jádru Ethernet MAC Lite [28]. Aplikaci tvoří spustitelný soubor *Ethernet.elf*, případně zdrojový soubor *Ethernet.c*. Hlavní funkce aplikace je čtení příchozích UDP paketů, pokud soubor obsahuje v datovém poli kontrolní příkaz „send“ v ASCII kódu, pak odpoví na adresu odesílatele UDP paketem s údajem o teplotě.

Aplikace využívá funkcí pro zápis a čtení systémových registrů (knihovna *stdio.h*), funkce pro příjem/odesílání ethernetových paketů (knihovna *xemaclite.h*) a přístup k systémovým parametrům (knihovna *xparameters.h*).

Adresace zařízení:

- IP adresa zařízení je pevně daná: **172.16.63.121**.
- MAC adresa zařízení je: **00:0A:35:01:02:03**.
- UDP port aplikace: **65296**

Popis funkcí:

- Thermometer:
  inicializuje ovladač EmacLite a připraví platformu. Poté se rozběhne cyklus while, který odchytává příchozí pakety a posílá je ke zpracování procesu ProcessRecvFrame.
  Parametry:

  DeviceID - číslo instance EmacLite

- ProcessRecvFrame:
  nejdříve kontroluje příchozí paket, jestli se jedná o protokol UDP/IP. Dále kontroluje, jestli se v datovém rámci nachází kontrolní sekvence „send“ v ASCII kódu. Pokud ano, pak sestaví a odešle UDP paket na adresu odesílatele, port 65296. V datovém rámci je obsažen 8 bytový údaj o teplotě ve formátu:

  - první 4 byty: teplota v °C
  - poslední 4 byty: teplota v °C za desetinnou čárkou

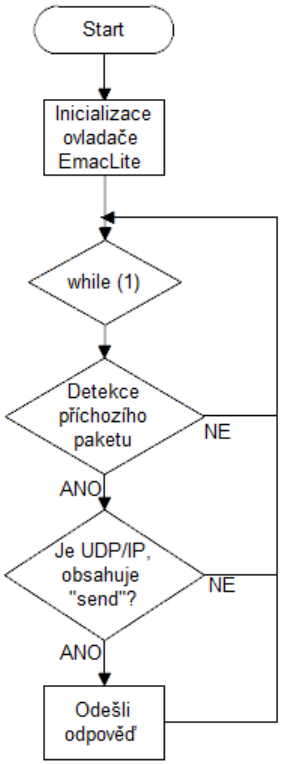
  Parametry:

  InstancePtr - ukazatel na konfiguraci instance EmacLite
  Packet_ID - identifikační číslo paketu

- CheckSumCalculation:
  slouží k výpočtu kontrolního součtu (CRC)
  Parametry:

  RxFramePtr - ukazatel na registr přijímače

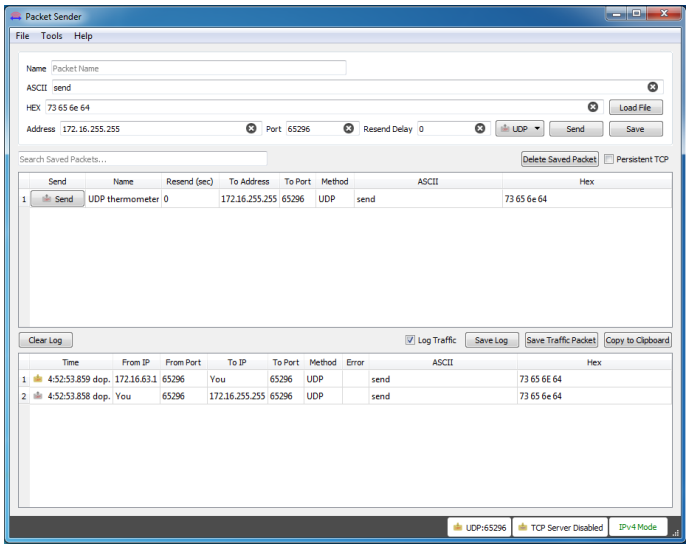
StartLoc - počáteční pozice výpočtu CRC
Lenght - délka řetězce určeného k součtu



Obrázek 5.9. Vývojový diagram aplikace Ethernet.elf

5.2.3 PC rozhraní

Ke komunikaci s deskou přes internetové rozhraní jsem použil programy Wireshark [29] a Packet Sender [30]. Tyto programy umožňují sledování síťové aktivity, přijímaní, sestavování a posílání paketů.



Obrázek 5.10. Program Packet Sender

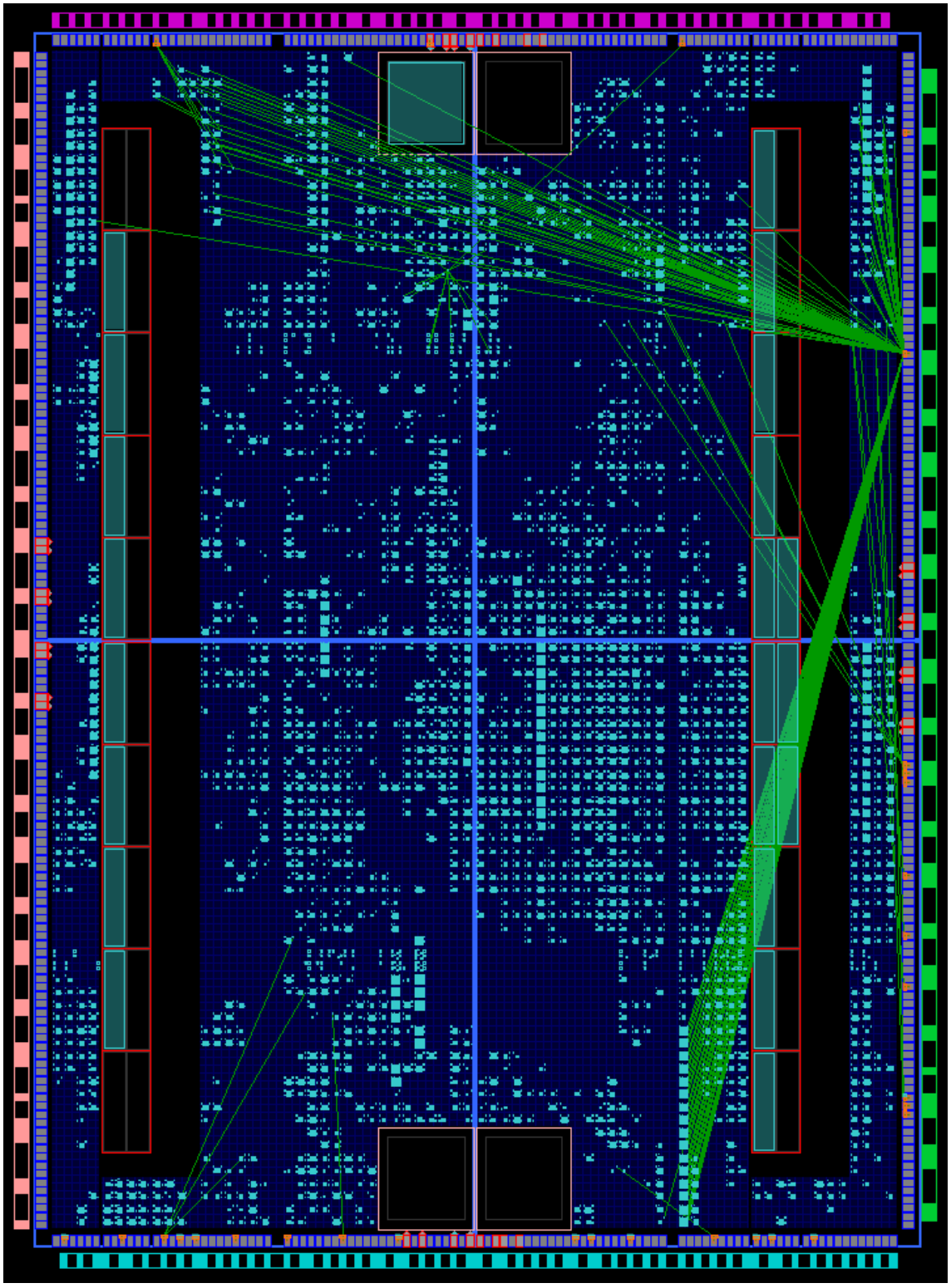
Kapitola 6
Výsledky

Na platformě Spartan-3E Starter Kit Board byl úspěšně navržen mikroprocesorový systém operující na sběrnici PLB. Součástí systému je soft-core mikroprocesor Microblaze, komunikující s jádrem Ethernet MAC Lite a s vlastním návrhem jádra Thermometer pro sběr dat z digitálního teploměru DS18B20. Na obrázku 6.1 je tabulka s využitými prostředky hradlového pole Spartan-3E.

Device Utilization Summary (actual values)				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	2,009	9,312	21%	
Number of 4 input LUTs	3,458	9,312	37%	
Number of occupied Slices	2,457	4,656	52%	
Number of Slices containing only related logic	2,457	2,457	100%	
Number of Slices containing unrelated logic	0	2,457	0%	
Total Number of 4 input LUTs	3,632	9,312	39%	
Number used as logic	3,065			
Number used as a route-thru	174			
Number used as 16x1 RAMs	4			
Number used for Dual Port RAMs	278			
Number used as Shift registers	111			
Number of bonded IOBs	29	232	12%	
IOB Flip Flops	16			
Number of RAMB16s	18	20	90%	
Number of BUFGMUXs	3	24	12%	
Number of DCMs	1	4	25%	
Number of BSCANs	1	1	100%	
Number of MULT18X18SIOs	3	20	15%	
Average Fanout of Non-Clock Nets	3.63			

Obrázek 6.1. Využití prostředků desky Spartan-3E

Na obrázku 6.2 je rozložení (layout) systémových bloků.



Obrázek 6.2. Floorplan layout návrhu na desce Spartan-3E

Z přehledu vidíme, že nejvíce vyčerpaným prostředkem jsou bloky BRAM. Je to pravděpodobně tím, že HW platformě byla pro běh aplikací přiřazena poměrně velká paměť (32KByte). Vzhledem k tomu, že paměti jsou tvořeny 18Kb bloky [12], využití 18 z 20 bloků je celkem očekávatelná hodnota.

V následující sekci byla zkoumán běh aplikace na FPGA desce, na obrázcích 6.3 a 6.4 jsou výsledky ze sledování síťové komunikace pomocí programů Wireshark a Packet Sender. Měření bylo provedeno přes jednoduché propojení kabelem LAN bez routeru.

No.	Time	Source	Destination	Protocol	Length	Info
67	24.570781	172.16.63.1	172.16.255.255	UDP	46	65296 → 65296 Len=4
68	24.570920	172.16.63.121	172.16.63.1	UDP	60	65296 → 65296 Len=8
69	25.551197	172.16.63.1	172.16.255.255	UDP	46	65296 → 65296 Len=4
70	25.551388	172.16.63.121	172.16.63.1	UDP	60	65296 → 65296 Len=8
71	26.532925	172.16.63.1	172.16.255.255	UDP	46	65296 → 65296 Len=4
72	26.533082	172.16.63.121	172.16.63.1	UDP	60	65296 → 65296 Len=8
73	27.516359	172.16.63.1	172.16.255.255	UDP	46	65296 → 65296 Len=4
74	27.516530	172.16.63.121	172.16.63.1	UDP	60	65296 → 65296 Len=8
75	28.498844	172.16.63.1	172.16.255.255	UDP	46	65296 → 65296 Len=4
76	28.499009	172.16.63.121	172.16.63.1	UDP	60	65296 → 65296 Len=8
77	29.485783	172.16.63.1	172.16.255.255	UDP	46	65296 → 65296 Len=4
78	29.485953	172.16.63.121	172.16.63.1	UDP	60	65296 → 65296 Len=8
79	30.464832	172.16.63.1	172.16.255.255	UDP	46	65296 → 65296 Len=4
80	30.464996	172.16.63.121	172.16.63.1	UDP	60	65296 → 65296 Len=8
81	31.446708	172.16.63.1	172.16.255.255	UDP	46	65296 → 65296 Len=4

Obrázek 6.3. Síťová aktivita zachycená programem Wireshark

	Time	From IP	From Port	To IP	To Port	Method	Error	ASCII	Hex
1	6:58:49.636 dop.	172.16.63.121	65296	You	65296	UDP		256875	20 20 32 35 36 38 37 35
2	6:58:49.636 dop.	172.16.63.1	65296	You	65296	UDP		send	73 65 6E 64
3	6:58:49.635 dop.	You	65296	172.16.255.255	65296	UDP		send	73 65 6e 64
4	6:58:48.115 dop.	172.16.63.1	65296	You	65296	UDP		send	73 65 6E 64
5	6:58:48.114 dop.	You	65296	172.16.255.255	65296	UDP		send	73 65 6e 64
6	6:58:46.893 dop.	172.16.63.121	65296	You	65296	UDP		255000	20 20 32 35 35 30 30 30
7	6:58:46.892 dop.	You	65296	172.16.255.255	65296	UDP		send	73 65 6e 64
8	6:58:46.892 dop.	172.16.63.1	65296	You	65296	UDP		send	73 65 6E 64
9	6:58:45.556 dop.	You	65296	172.16.255.255	65296	UDP		send	73 65 6e 64
10	6:58:45.556 dop.	172.16.63.1	65296	You	65296	UDP		send	73 65 6E 64
11	6:58:43.348 dop.	172.16.63.1	65296	You	65296	UDP		send	73 65 6E 64
12	6:58:43.348 dop.	172.16.63.121	65296	You	65296	UDP		251875	20 20 32 35 31 38 37 35
13	6:58:43.347 dop.	You	65296	172.16.255.255	65296	UDP		send	73 65 6e 64

Obrázek 6.4. Síťová aktivita zachycená programem Packet Sender

Vidíme, že program Wireshark detekuje každý paket vyslaný aplikací na desce, lze z toho usoudit, že aplikace i HW platforma fungují správně.

Program Packet Sender zachytává přibližně každý druhý paket vyslaný aplikací na desce, bližší důvody by musely být předmětem dalšího výzkumu.

Kapitola 7
Závěr

Během tvorby této práce se mi podařilo osvojit si návrh digitálních systémů v návrhových systémech firmy Xilinx. Jelikož mne velmi zajímá historie digitální techniky, velkou část této práce jsem nejdříve věnoval prvním logickým obvodům a předchůdcům FPGA. Trocha pozornosti byla také věnovaná současnému směru vývoje této technologie.

Následně jsem popsal, jak většinou probíhá návrh logického obvodu, od počátečního stanovení cíle až po konečnou implementaci do fyzického obvodu. Kdybych to měl porovnat s návrhem obecného tištěného digitálního obvodu, je zde vidět velký rozdíl v náročnosti, ve prospěch programovatelného hradlového pole. Myslím si, že právě to je důvod, proč je FPGA tak využívané, a pravděpodobně tomu bude tak i nadále. Díky neustálému technologickému vývoji bude nabízet stále více funkcí a nacházet nová použití. Na druhou stranu, rychle se vyvíjejících technologií je dnes velmi mnoho, a jen určitý zlomek z nich obstojí v dnešním světě.

Menší část byla věnovaná stručnému úvodu do jazyka VHDL. Jazyk ve své struktuře rozhodně nepatří mezi nejjednodušší, a jeho ucelený popis by vydal na celou diplomovou práci.

Čtvrtá část s tématem Ethernet a sítě je shrnutím informací o nejběžněji používaných síťových protokolech, speciálně protokolech IP a UDP. Do rodiny těchto protokolů patří také protokol TCP, avšak ve svém návrhu jsem jej nepoužil, proto je uveden pouze v porovnání s protokolem UDP. Pozornost byla také věnována technologii Ethernet, a to jak z hlediska fyzické stránky přenosu, tak z hlediska protokolu.

V návrhové části se podařilo úspěšně navrhnout systém pro komunikaci s deskou přes internetové rozhraní. Návrh internetového rozhraní dokáže efektivně komunikovat se vzdáleným zařízením pomocí UDP paketů. Na vyslaný kontrolní příkaz umí rozhraní odpovědět a předat užitečné informace.

Tato schopnost byla demonstrována na desce Spartan-3E s hradlovým polem XC3S500E a připojeným digitálním teploměrem DS18B20. Na desce byl implementován systém se soft-core mikroprocesorem MicroBlaze, sběrnicí PLB a periferními jádry pro čtení údajů z teploměru a pro komunikaci přes Ethernet rozhraní.

V úvodu jsem se zmínil o technologii Internet věcí (Internet of Things), která byla jakýmsi cílovým horizontem, kam by tato práce mohla v ideálním případě dospět. Je jasné, že možnosti komunikace navrženého systému nesplňují požadavky, které jsou současnými technologiemi kladeny. Systém dokáže odpovídat na přesně určené dotazy, ale musí být v konfigurované síti a nedokáže si sám svou konfiguraci upravit. K tomu, aby se systém stal adaptivním a „chytrým“, vede ještě dlouhá cesta.

Literatura

1. *MOS/LSI Circuits.*, Texas Instruments, 1971.
2. *Logic Devices*, IVC, http://beta.ivc.no/blog/2011/03/30/logic-devices/, 2011
3. *PLD: A brief history*, http://www.island.net/~kdbrown/pld_hist.php, 2011
4. *CPLD*, Xilinx, https://www.xilinx.com/cpld/, 2017
5. Procházka, Pavel. *Architektura CPLD XC9500* http://www.prochazka.profitux.cz/index.php?a=xilinx/architektura_cpld_xc9500
6. Betz, Vaughn. *FPGA Architecture*, http://www.eecg.toronto.edu/~vaughn/challenge/fpga_arch.html, 2012
7. Roelandts, Wim. *Xcell Issue 32*, https://www.xilinx.com/publications/archives/xcell/Xcell32.pdf, 1999
8. Daněk, Martin. *Programovatelná hradlová pole – FPGA*, http://automa.cz/cz/casopis-clanky/programovatelna-hradlova-pole-fpga-2006_02_30930_672/, 2006
9. Hawick, K.A. *Field Programmable Gate Arrays for Computational Acceleration of Lattice-Oriented Simulation Models* , http://bit.ly/2qq3FbP
10. Graves, David. *FPGA - Field Programmable Gate Array* , https://www.cl.cam.ac.uk/teaching/1011/P35/obj2.2/zhp7e8c2a2a6.html
11. *Spartan-3 Generation FPGA User Guide*, Xilinx, https://www.xilinx.com/support/documentation/user_guides/ug331.pdf, 2011
12. *Spartan-3E FPGA Family Data Sheet*, Xilinx, https://www.xilinx.com/support/documentation/data_sheets/ds312.pdf, 2013
13. *Digital Design Blog*, AbcLinuxu, http://www.abclinuxu.cz/blog/digital_design
14. Sedláček, R. *Stručný popis jazyku VHDL*, http://measure.feld.cvut.cz/system/files/files/cs/vyuka/predmety/A0B38APH/VHDL.pdf, 2016
15. *VHDL Online Help*, http://www.vhdl.renerta.com/
16. *Ethernet Technologies*, Cisco systems , http://docwiki.cisco.com/wiki/Ethernet_Technologies
17. *IEEE Standard for Ethernet*, ieee.org , http://standards.ieee.org/getieee802/download/802.3-2012_section1.pdf
18. Tetz, Edward. *Cisco Networking: Data Framing* , http://www.dummies.com/programming/networking/cisco/cisco-networking-data-framing/
19. RFC 791, *Internet Protocol*, IEFT , https://tools.ietf.org/html/rfc791
20. RFC 1700, *Assigned Numbers*, IEFT , https://tools.ietf.org/html/rfc1700
21. Loh, Mike. *Understanding and Reading packets* , https://www.edgis-security.org/infosec/understanding-and-reading-packets/, 2013

22. *Service Name and Transport Protocol Port Number Registry*, IANA, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, 2017
23. *Programmable Resolution 1-Wire Digital Thermometer*, https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
24. Jašek, Petr. *Digitální teploměr s DS18B20*, semestrální práce, 2017
25. *LogiCORE IP XPS Ethernet Lite Media Access Controller*, https://www.xilinx.com/support/documentation/ip_documentation/xps_ethernetlite.pdf, 2010
26. *Platform Specification Format Reference Manual*, https://www.xilinx.com/support/documentation/sw_manuals/edk10_psf_rm.pdf, 2008
27. *Standalone Board Support Package (BSP)*, http://www.wiki.xilinx.com/Standalone+Board+Support+Package+%28BSP%29, 2015
28. *Ethernet MAC Lite Examples*, https://github.com/Xilinx/embeddedsw/tree/master/XilinxProcessorIPLib/drivers/emaclite/examples, 2015
29. WireShark, https://www.wireshark.org/
30. Packet Sender, https://packetsender.com/

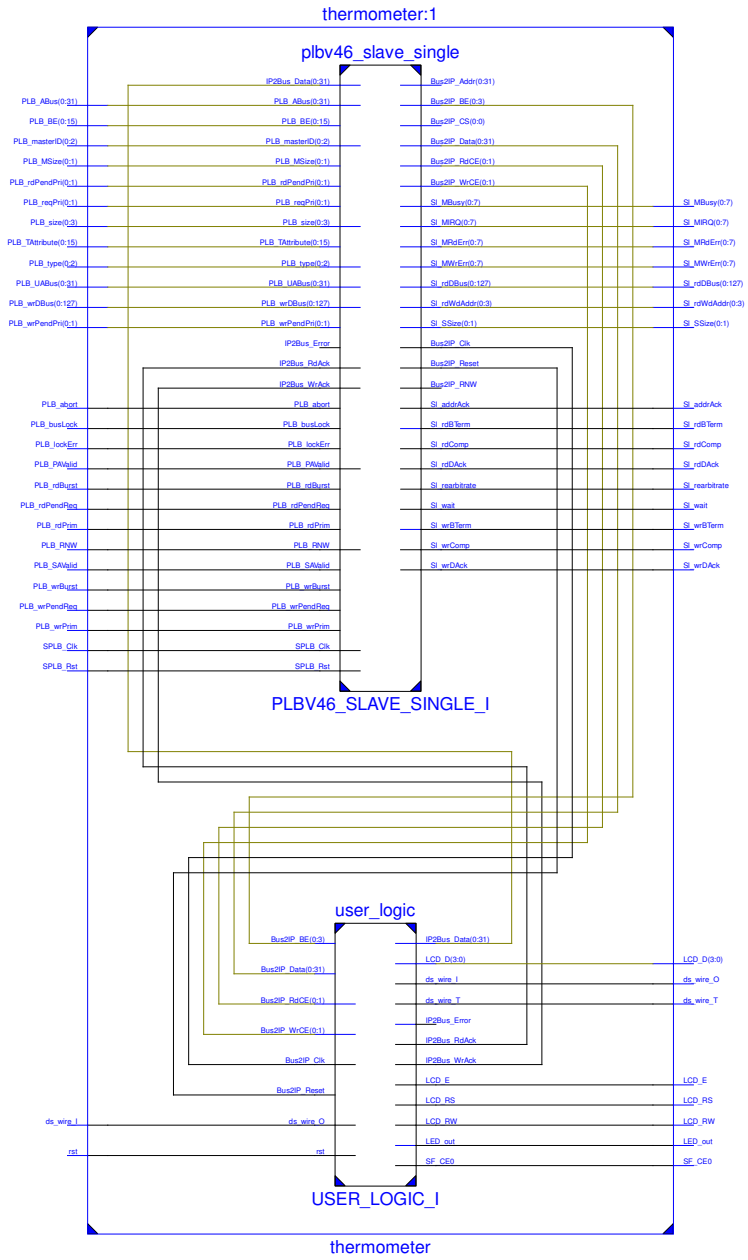
Příloha A
Seznam použitých zkratek

ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
AXI	Advanced Microcontroller Bus
BSB	Base System Builder
BSP	Board Support Package
CIDR	Classless Inter-Domain Routing
CLB	Configurable Logic Block
CPLD	Complex Programmable Logic Device
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier-sense multiple access with collision detection
ČVUT	České vysoké učení technické v Praze
DCM	Digital Clock Manager
DDR SDRAM	Double data rate synchronous dynamic random-access memory
EDK	Embedded Development Kit
EEPROM	Electrically Erasable Programmable Read-Only Memory
FEL	Fakulta elektrotechnická CVUT
FPGA	Field-Progammable Gate Array
GAL	Generic Array Logic
HDL	Hardware Description Language
I/O	Input/Output
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical and Electronics Engineers
IOB	Input/Output Blocks
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISE	Integrated Synthesis Environment
JTAG	Joint Test Action Group
LCD	Liquid-crystal display
LED	Light-emitting diode
LUT	Look-Up Table
MAC	Media Access Control
PAL	Programmable Array Logic
PC	Personal computer
PHY	Physical Layer
PLA	Programmable Logic Array
PLB	Processor Local Bus
PLD	Programmable Logic Device
PLL	Phase-Locked Loop
PROM	Programmable Read-Only Memory

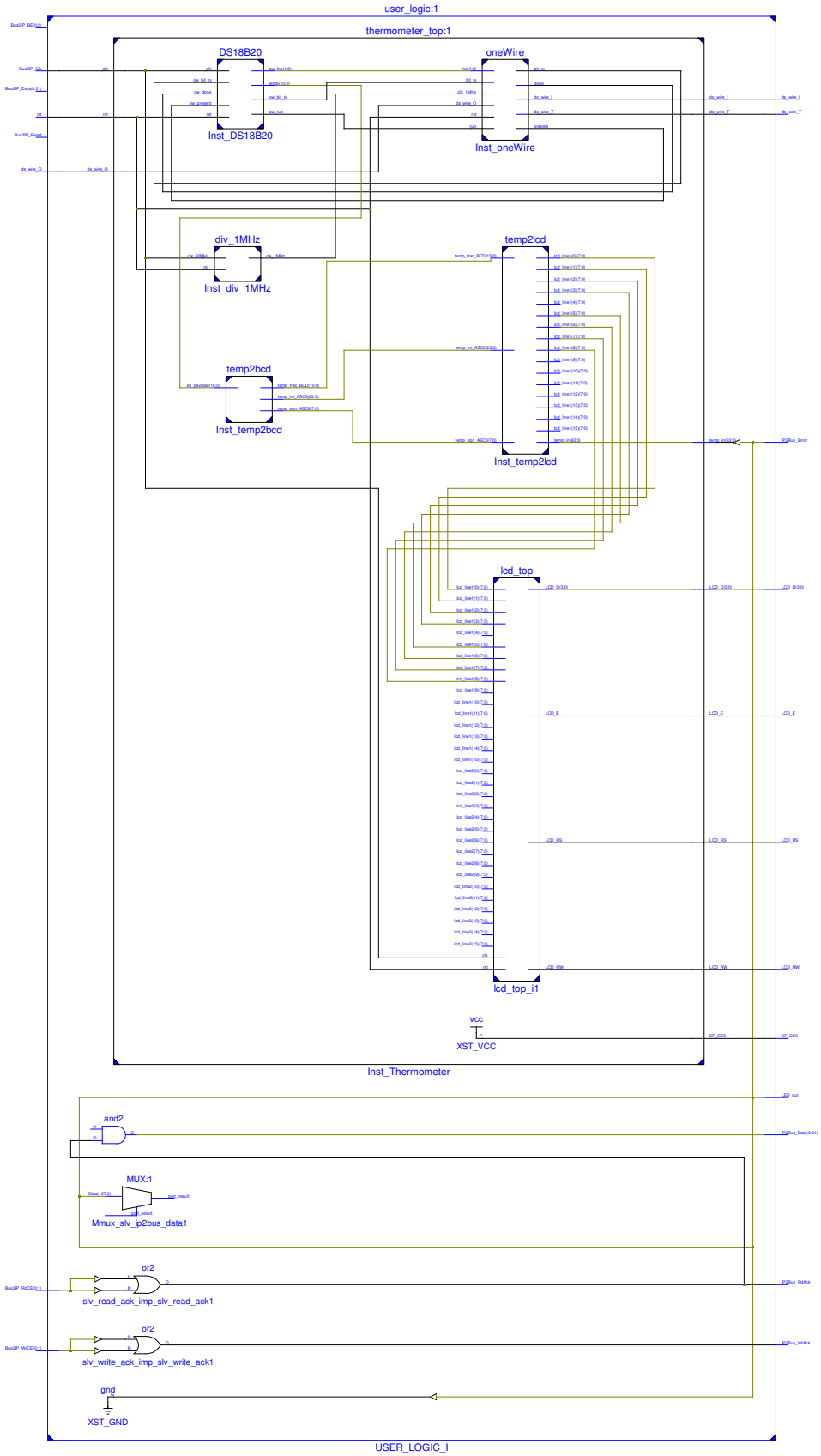
RM-OSI	Reference Model - Open Systems Interconnection
RTL	Register Transfer Level
SDK	Software Development Kit
STA	Static Time Analysis
TCP	Transmission Control Protocol
TTL	Time-To-Live
UART	Universal asynchronous receiver/transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VGA	Video Graphics Array
VHDL	Very high speed integrated circuit Hardware Description Language
XPS	Xilinx Platform Studio

Příloha B
Diagramy

B.1 Periferie Thermometer



Obrázek B.1. Blokové schéma periferie Thermometer



Obrázek B.2. Schéma bloku *user_logic* periferie Thermometer

Příloha C
Instrukce pro použití

V této sekci je vysvětlen postup, jak je možné návrh použít.

1. Desku Spartan-3E připojíme ke zdroji napětí a zapneme.
2. Připojíme digitální teploměr DS18B20 k expanznímu portu J1 na desce.
3. Připojíme konfigurační kabel USB k desce a k počítači.
4. Připojíme desku i počítač na jednu síť.
5. Nakonfigurujeme síť na privátní adresy 172.16.0.0/12.
6. Pomocí programu Xilinx SDK nebo iMPACT nahrajeme bitstreamový soubor *system.bit* na desku.
7. V programu Xilinx SDK spustíme aplikaci *Ethernet.elf* a zvolíme možnost Launch on Hardware.
8. Pomocí programu Packet Sender vyšleme na adresu 172.16.63.121, port 65296 UDP paket s textem „send“. Obratem by nám měl přijít UDP paket, obsahující v těle zprávy aktuální údaj o naměřené teplotě. V případě, že odpověď nepřichází, zkusíme vyslat UDP paket na broadcast adresu 172.16.63.255, port 65296 s textem „send“.