VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

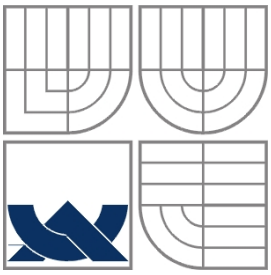
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

KOMUNIKACE TEPLOTNÍCH SENZORŮ S CPLD PŘES SBĚRNICI SMBUS

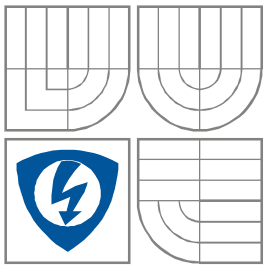
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE KAREL TONAR
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

KOMUNIKACE TELOPTNÍCH SENZORŮ S CPLD PŘES SBĚRNICI SMBUS

THERMAL SENZORS COMMUNICATIONS WITH CPLD USING SMBUS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Karel Tonar
AUTHOR

VEDOUCÍ PRÁCE — Ing. Michal Kováč
SUPERVISOR

BRNO, 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student: Tonar Karel ***ID:*** 77849
Ročník: 3 ***Akademický rok:*** 2007/2008

NÁZEV TÉMATU:

Komunikace teplotních senzorů s CPLD přes sběrnici SMBUS

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s dvoudrátovou komunikací na sběrnici SMBUS. Seznamte se se strukturou CPLD a programováním v jazyku VHDL. Najděte vhodné teplotní senzory, které komunikují se sběrnicí SMBUS, a prostudujte možnosti jejich řízení. Senzory budou komunikovat s programovatelným logickým obvodem CPLD na vývojové desce, a na LED displeji budou zobrazovat teplotu.

Navrhněte koncepci digitálního teploměru. Navhrněte vývojový diagram programu do cílového obvodu CPLD a vytvořte program ve VHDL pro digitální teploměr. Funkčnost Vytvořeného programu ověřte simulací.

Navrhněte a realizujte desku modulu s teplotním senzorem, který se připojí na vývojovou desku. Ověřte funkčnost teploměru. Rozšiřte teploměr o další senzory. Do programu připojte funkce maxima, minima a průměru teploty. K teploměru navhrněte komunikaci s PC pomocí rozhraní UART a měřené hodnoty zobrazujte a zaznamenávejte v PC.

DOPORUČENÁ LITERATURA:

[1] Maxim-IC. Sunnyvale, CA. Thermal Management, Sensors, and Sensor Conditioners. 2007. Firemní webové stránky. Dostupné na http://www.maxim-ic.com/Sensors.cfm, 2007.

[2] KOLOUCH, J. Programovatelné logické obvody a návrh jejich aplikací v jazycích ABEL a VHDL. Elektronické skriptum. Brno: FEKT VUT Brno, 2002.

Termín zadání: 11.2.2008 ***Termín odevzdání:*** 6.6.2008

Vedoucí práce: Ing. Michal Kováč

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Karel Tonar
Bytem:	Polní 337, Havlíčkova Borová, 582 23
Narozen/a (datum a místo):	24. července 1986 v Havlíčkově Brodě

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Dr. Ing. Zbyněk Raida, předseda rady oboru Elektronika a sdělovací technika
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

   ☐ disertační práce
   ☐ diplomová práce
   ☒ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

Název VŠKP:	Komunikace teplotních senzorů s CPLD přes sběrnici SMBUS
Vedoucí/ školitel VŠKP:	Ing. Michal Kováč
Ústav:	Ústav radioelektroniky
Datum obhajoby VŠKP:	_____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.

3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

* hodící se zaškrtněte

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti

   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy (z důvodu utajení v něm obsažených informací)

4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.

2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 6. června 2008

……..	……
Nabyvatel	Autor

Abstrakt

Cílem bakalářské práce je seznámení se s dvoudrátovou komunikací na sběrnici SMBUS, konkrétně se jedná o vzájemnou komunikaci obvodu FPGA s teplotními senzory.

Zpracování práce obsahuje výběr vhodného teplotního senzoru, návrh modulu pro měření teploty a návrh programu pro programovatelný logický obvod.

Program zpracovává data z modulů pro měření teploty a zobrazuje je na LED displeji. Program umožňuje zobrazení minimální, maximální a průměrné teploty za určitý čas. Všechny programy pro logický obvod jsou realizovány v jazyce VHDL.

Dále teploměr posílá data do PC pomocí rozhraní UART. Zde je teplota opět zobrazována a zaznamenávána.

Při práci byla použita vývojová deska SPARTAN-3 a návrhový systém Xilinx ISE 9.2i.

Klíčová slova

programovatelný logický obvod, digitální teplotní senzor, SMBUS, dvoudrátová komunikace, jazyk VHDL

Abstract

The goal of bachelor’s thesis is the introduction with two - wire circuit communication on the SMBUS, it is concretely related to alternate communication of FPGA with thermal sensors.

The bachelor’s thesis also contains the choice of optimal thermal sensor, the proposal of module for measuring the temperature and proposal of programme for Programmable Logical Device.

The programme processes data from the modules of temperature measurement and data are indicate on LED display. The programme makes it possible to display minimum, maximum and average temperature during certain time interval. All of programmes are implemented for logical device in VDHL language.

The thermometer sends data to Personal Computer over UART interface. There is temperature displayed and storaged once again.

During working on this bachelor’s thesis was used SPARTAN-3 starter board and design system Xilinx ISE 9.2i.

Keywords

Programmable Logical Device, digital thermal sensor, SMBUS, two - wire communication, VHDL language

Bibliografická citace projektu

TONAR, K. *Komunikace teplotních senzorů s CPLD přes sběrnici SMBUS*. Semestrální projekt BB2E. FEKT VUT Brno, 2008. 81 s., 13 příl.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Komunikace teplotních senzorů s CPLD přes sběrnici SMBUS jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 6. června 2008 .....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Michalu Kováčovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 6. června 2008 .....
podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Způsob měření teploty pomocí digitálních senzorů teploty je v dnešní moderní době stále více oblíbenější. Senzor obsahuje snímač teploty, který je integrovaný přímo na čipu nebo rozhraní pro připojení externího teplotně citlivého snímače pracujícího například na principu vlastností PN přechodu. Digitální teplotní senzory dokáží změnu teploty ihned zaregistrovat, zpracovat a vyhodnotit. Doba reakce na změnu parametru je téměř okamžitá. Senzory dokáží měřit teplotu s velkou přesností a rozlišitelností na několik desetinných míst. Výstupní data interpretuje ve formátu binárního signálu. Ten je ze snímače získáván pomocí jednovodičového nebo vícevodičového vedení, popř. sběrnice.

Pro současnou dobu mikrokontrolérů a logických obvodů je nejvíce využívaná dvoudrátová komunikace. Způsoby přenosu signálů se pro jednotlivé typy komunikace nepatrně liší. Každá komunikace přenáší informace pomocí pevně zadaných podmínek a formátů přenosu – tzv. rámce. Nejoblíbenější z nich se zdá být I_2C sběrnice, jejíž odnoží je i SMBUS, na níž může být připojeno i více než jedno zařízení schopné ovládat sběrnici. Toto zařízení se nazývá MASTER. Zařízení jemu podrobené je označováno jako SLAVE. Na zařízení typu MASTER jsou kladeny určité uživatelské požadavky např. řízení, správa a zpracování signálů, možnost připojení periferií, jejich ovládání, programovatelnost, komunikace s PC, univerzálnost atd. Tyto vlastnosti splňují mikrokontroléry, programovatelné logické obvody aj. SLAVE zařízení slouží ve většině případů pro měření či kontrolu. Mezi zmíněné patří např. snímače teploty, tlaku, vlhkosti, optické snímače, vysílače atd.

Programovatelné logické obvody (angl. Progammable Logical Device) zvláště CPLD se v dnešní době moderní elektrotechniky a mikroprocesorové techniky používají ke zminiaturizování programové a výpočetní části přístrojů. Obsahují programovatelné matice hradel AND následované hradlem OR a makrobuňkou. Výrobou logických obvodů se dnes zabývá mnoho světových polovodičových firem. Mezi největší výrobce patři firmy Xilinx, Altera, Lattice Semiconductor, Actel, Atmel, Cypress Semiconductor a QiuckLogic. Konkurence na poli logických obvodů je tedy vysoká a v budoucnosti stále poroste, což nám zajišťuje modernizaci obvodů a jejich neustálý vývoj. Použití logických obvodů se uplatňuje v mnoha různých odvětví např. telekomunikace, výpočetní technika atd. Mezi velké odběratele PLD obvodů patří např. Motorola, Alcatel, Siemens a další [2]. Významným návrhovým systémem logických obvodů se stal ABEL se stejným názvem jazyka vyšší úrovně. Pro návrh nových číslicových systémů založených na PLD obvodech se však doporučuje používat některý z novějších HDL jazyků, např. jazyk VHDL nebo jazyk Verilog [1].

2 Rozbor a analýza bakalářské práce

2.1 Výstupní práce projektu BB2E

Semestrální projekt BB2E prakticky navazoval na předchozí práci BB1E, jejíž cílem bylo seznámení se sběrnicí SMBUS, strukturou CPLD a programovacím jazykem VHDL. Dále pak průzkum a výběr vhodných teplotních senzorů a studie jejich řízení.

Z nabízených teplotních senzorů bylo vybráno několik následujících LM73, LM94, LM952xx, AD7415, MAXIM DS1631, z nichž nejvíce vhodným se zdál být senzor firmy ANALOG DEVICES typu AD7414ARTZ popř. AD7415ARTZ, jejichž teplotní rozsahy jsou dostačující vzhledem k zadání projektu bakalářské práce [6], [7], [8], [9], [10].

V semestrálním projektu BB2E byla navržena koncepce digitálního teploměru resp. základní blokové schéma programu, který byl v bakalářské práci implementován do logického obvodů CPLD (FPGA). Dalším bodem práce bylo vlastní sestrojení programu a jeho ověření funkčnosti pomocí simulace.

Program byl sestrojen z dílčích podprogramů (bloků programu), na kterých bylo provedeno ověření funkčnosti. Bloky byly následně pospojovány tak, aby konečný program fungoval dle zadání.

2.2 Zadání bakalářské práce

- návrh a realizace desky modulu s teplotním senzorem
- ověření funkčnosti teploměru
- rozšíření teploměru o další senzory
- rozšíření programu o funkce minima, maxima a průměru
- návrh komunikace s PC pomocí rozhraní UART

2.3 Výstup bakalářské práce

- VHDL kód pro programovatelný logický obvod CPLD (FPGA)
- moduly s teplotními senzory firmy ANALOG DEVICES typu AD7414ARTZ a AD7415ARTZ
- UART komunikace

2.4 Postupy a výběr řešení zadané problematiky

K práci na projektu byla vedoucím doporučena již zkonstruovaná vývojová deska se dvěma CPLD obvody s názvem XC2-XL. Doporučena byla z toho důvodu, že Ústav radioelektroniky tuto desku již vlastní a tudíž nebylo nutné složitě objednávat jinou. Samozřejmě, že komunikace s CPLD přes sběrnici SMBUS by mohla být odzkoušena na jiné víceúčelové desce s různými obvody CPLD, ale toto řešení bylo nejjednodušší.

XC2-XL je vývojová deska s platformou pro CPLD obvody. Obsahuje dva logické obvody CPLD - CoolRunnerII XC2C256 a XC 9572XL. Součástí vývojové desky je JTAG programovací rozhraní, které obsahuje napájení, zdroj hodinového signálu a základní vstupně/ výstupní zařízení. Programy tak mohou být implementovány ihned. XC2-XL deska je slučitelná se všemi verzemi Xilinx CAD

nástrojů, včetně volných WebPack nástrojů dostupných na internetových stránkách firmy Xilinx [11].

Právě kvůli slučitelnosti XC2-XL bylo pro vlastní programování vybráno prostředí Xilinx ISE . Jedná se o nejnovější verzi tohoto programu (Xilinx ISE 9.2i) volně dostupnou na webových stránkách firmy Xilinx. Toto prostředí dovoluje i následné simulace programu. Program může být psán přímo ve formě zdrojového textu nebo může být vytvořen pomocí tzv. schematic [12].

Během práce na bakalářské práci bylo zjištěno, že zvolená vývojová deska XC2-XL nevyhovuje požadavkům zadání. Deska neobsahovala sériový port RS-232 používaný pro zprostředkování komunikace UART. Problém byl vyřešen tak, že stávající vývojová deska byla nahrazena jinou, taktéž dostupnou na Ústavu radioelektroniky. Nynější deska od výrobce Xilinx nese název SPARTAN-3. Deska je téměř shodná s deskou XC2-XL, obsahuje stejné porty, ovládání, napájení, JTAG programovací rozhraní, zobrazovací a indikační součásti. Mezi deskami jsou však i rozdíly. Největším je použitý logický obvod. Základním prvkem vývojové desky SPARTAN-3 je programovatelný obvod FPGA (konkrétně typ XC3S200). FPGA obvody mají jinou strukturu a obsahují více hradel, než obvody CPLD. Další rozdíl tvoří použitý krystalový oscilátor, který do desky dodává obdélníkový signál s kmitočtem f = 50 MHz oproti oscilátoru na desce XC2-XL. Jeho kmitočet byl 1.8432 MHz. Poslední důležitou změnou je použitý sériový port RS-232. Deska je jako předchozí plně slučitelná se všemi verzemi Xilinx CAD nástrojů, včetně volných WebPack nástrojů [13].

K měření teploty byly vybrány senzory firmy ANALOG DEVICES typu AD7414ARTZ a AD7415ARTZ. AD7414 (AD7415) je samostatný jednočipový snímač teploty, který umožňuje přesné určení okolní teploty. Snímač obsahuje 10-ti bitový analogově/ číslicový převodník, který měřenou teplotu interpretuje ve formátu dvojkových doplňků. Zařízení je napájeno 2,5 V až 5,5 V a pracuje v rozsahu měřených teplot - 40 ℃ až + 125 ℃ s maximální volitelnou rozlišovací schopností 0,25 ℃ [9]. Konstrukce modulu s teplotním senzorem může být provedena několika způsoby. Příkladem může být např. zapojení součástek na nepájivém kontaktním poli nebo napájení zařízení na desku plošných spojů. Zapojení součástek spolu s teplotním senzorem bylo převzato z datasheetu používaného senzoru [9]. Toto zapojení zajišťuje určitou záruku výrobce, že modul pro měření teploty bude bezproblémově pracovat. Bylo rozhodnuto, že výsledný modul bude umístěn na desce plošných spojů. Pro návrh této desky je možno použít nespočet návrhových programů. Jako nejpřijatelnější je však návrhový program Eagle Layout Editor. Byla použita verze 4.16r2 volně přístupná na stránkách fakulty. Tento program je na VUT Brno využíván i jako výukový a pro podobné návrhy je zcela optimální.

3 Teplotní senzory

Teplota je a jistě dále bude patřit k nejvíce měřeným neelektrickým veličinám v oblasti průmyslové regulace i spotřební a výpočetní elektroniky. Na její hodnotě obvykle záleží řada výrobních procesů a regulací, ať již přímo nebo nepřímo. Samotné měření teploty lze rozdělit na dva sériově spojené bloky:

- blok převodu teploty na elektricky zpracovatelný signál
- blok zpracování tohoto signálu.

První blok obsahuje tzv. senzory teploty - převodníky teploty na elektricky měřitelnou veličinu (odpor, napětí), které mohou pracovat na nejrůznějších principech. Vždy se však využívá vlastností materiálů, které v jiných aplikacích obvykle považujeme za parazitní a nežádoucí.

Druhý blok obsahuje elektrický obvod pro úpravu signálu a pro následný přenos. Tento blok může buď jen přizpůsobovat a zesilovat signál, nebo již unifikovat analogový elektrický signál (napěťový 0 – 10 V, proudový 0 – 20 mA, 4 - 20mA - analogová proudová smyčka - atd.) ze senzoru s případnými korekcemi nežádoucích vlastností, jakými jsou například nelinearita, odstup signál - šum apod. Také však může provádět přímý převod analogového signálu na digitální, korekci a přizpůsobit přenos dat například do obvyklého standardu sériového přenosu (RS-232, RS-485 apod.). V případě, že oba bloky jsou součástí jednoho kompaktního provedení, často se takové čidlo označuje jako smart (inteligentní). Základní princip měření teploty lze rozdělit do dvou hlavních skupin:

a) *Dotykové měření* – senzor musí být připevněn (dotýkat se) objektu či látky, jejíž teplotu má měřit. Využívá se zde přímého přenosu tepla mezi dvěma objekty. Lze ho využít všude tam, kde je snadný přístup k měřenému objektu nebo okolnímu prostředí.

b) *Bezdotykové měření* – senzor se nachází v určité vzdálenosti od měřeného objektu, a tím nedochází k vzájemnému ovlivňování. Využívá se zde jevu, kdy každý objekt o určité teplotě vyzařuje určitou vlnovou délku infračerveného záření – pyrometrie [14].

3.1 Dotykové senzory

Ty se dále dělí na:

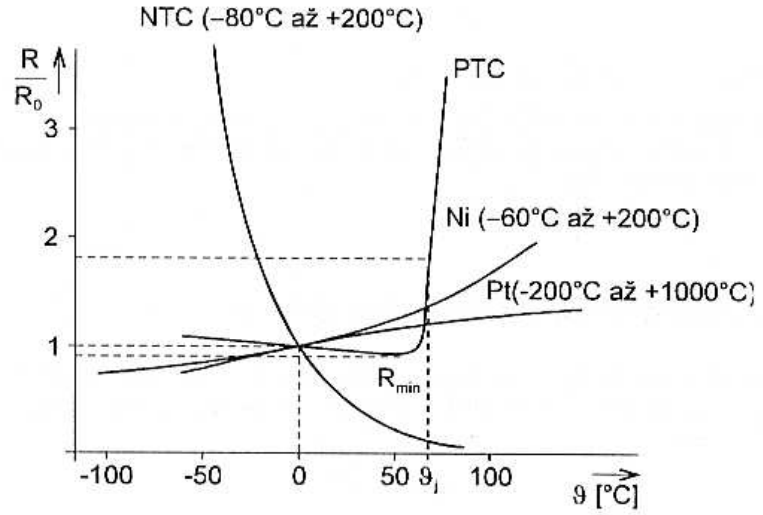
- *elektrické*
- *dilatační*
- *speciální (jiné).*

Mezi dotykové senzory patří např.:

- *Odporové pasivní senzory* – využívají převodu změny teploty na změnu odporu (kovové, polovodičové senzory).
- *Napěťové aktivní senzory* – využívají převodu změny teploty na změnu napětí (senzory s PN přechodem, termočlánky).
- *Krystalové senzory* – využívají převodu změny teploty na změnu frekvence.
- *Optické senzory* – využívají závislosti odrazu, útlumu a disperze světla na teplotě.

3.1.1 Odporové pasivní senzory

Využívají se především v oblastech s velkým rozsahem měřených teplot (až přes 1000 ℃), jako jsou oblasti regulace topných systémů, zpracování materiálů, potravinářství atd. Jeden z hlavních sledovaným parametrů pro výběr senzoru je převodní charakteristika, její linearita a strmost [14]. Ta je různá v závislosti na typu odporového senzoru - kovové (převážně Pt a Ni), polovodičové - termistory (NTC a PTC) a monokrystalické senzory. Zvláště kovové senzory se vyznačují velkou linearitou, malou chybou měření a dlouhodobou stabilitou. Termistory se vyznačují velkou citlivostí.



Obr. 3.1: Porovnání převodních charakteristik odporových senzorů

3.1.2 Napěťové aktivní senzory

Výstup těchto senzorů je již přímo napěťový signál. Zvláště polovodičové křemíkové senzory s PN přechodem mají výhodu v snadné integraci spolu se zpracovávajícími obvody, logikou a AD převodníkem přímo na jenom čipu. Další výhodou je velká citlivost – 2 mV/ ℃ a linearita.

Jejich nevýhodou je pak nízký měřitelný rozsah teplot daný materiálem (do 125 ℃). Právě tento způsob je téměř vždy využíván pro senzory teploty s diskrétními či číslicovými sériovými výstupy. Termočlánky se naopak vyznačují velkým rozsahem teplot (až 2000 ℃) a miniaturními rozměry, ale nízkou citlivostí.

3.2 Bezdotykové senzory

Tohoto principu se tedy dá využít pro stálé měření teploty předmětů v nebezpečných nebo velmi nepřístupných prostorách, pohyblivých částí, stejně tak jako rychlého měření příručními přístroji při technických kontrolách, údržbářských pracích apod. Výhodou je i měření velmi vysokých teplot (až 3000 ℃), nebo naopak velmi nízkých (od – 200 ℃). Záleží na zvoleném fyzikálním principu funkce senzoru a zvolených materiálech [14]. Nevýhodou mohou být chyby měření způsobené nejistotou stanovení emisivity měřeného objektu, prostupnosti prostředí, odraženým zářením z okolního prostředí apod. Bezdotykové senzory lze dle fyzikálního principu rozdělit na:

a) tepelné senzory infračerveného záření – infračervené termočlánky bolometry, pyroelektrické senzory

b) kvantové senzory – polovodičové infračervené fotodiody a fotovodivostní detektory.

První uvedená skupina využívá ohřátí citlivé části senzoru vlivem infračerveného záření a převodu teploty na elektrický signál. Druhá část senzoru využívá přímé interakce fotonů záření s polovodičovým materiálem detektoru.

3.3 Zpracování a přenos el. signálu senzorů

Jak již bylo uvedeno, v průmyslu musí být zpravidla přímý signál z čidla předzpracován na signál vhodný pro následující přenos na vzdálenosti desítek až stovek metrů z důvodu odolnosti proti rušení. I v dnešní době se stále ještě využívá plného analogového předzpracování signálu a unifikace například na signál 0 – 10 V nebo 4 - 20 mA pro proudovou smyčku. Převážně však pro odporové senzory. Obvykle se využívá principu odporového můstku nebo proudových zdrojů. Stále více se však využívá přenosu po digitální sériové lince, která umožňuje pomocí vedení přenos informace na velkou vzdálenost [14].

3.4 Inteligentní senzory teploty

Na trhu je nepřeberné množství různých senzorů teploty pracující na různých fyzikálních principech. S postupující digitalizací se objevily tzv. inteligentní senzory teploty (smart temperature sensors). Ty již obsahují snímač teploty integrovaný přímo na čipu nebo přímé rozhraní pro připojení externího teplotně citlivého snímače (např. PN přechod). Senzor pak dále provádí zpracování signálu (linearizace, korekce, redukce šumu apod.) a hlavně téměř vždy A/D převod. Často se vyskytují i přídavné funkce jako jsou různé režimy snížení spotřeby energie (shutdown mód), nastavení rozsahu s indikací jeho překročení [14].

3.5 Použití inteligentních teplotních senzorů

- Počítačová technika, servery, pracovní stanice
- Měření teploty v mobilních aplikacích
- Regulace topení a vytápění
- Měření teploty vnitřních i venkovních prostorů
- Kontrola a detekce přehřátí či podchlazení
- Testování zařízení

Pro účely bakalářské práce bylo vybráno několik teplotních senzorů viz. datasheety [6], [7], [8], [9]. Všechny obvody splňují požadavky pro návrh a konstrukci modulu teplotního čidla.

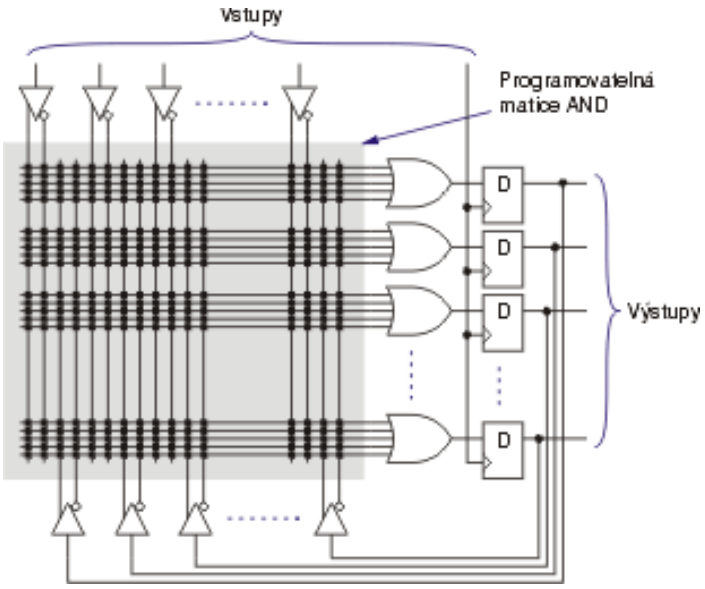
4 Programovatelné logické obvody

Programovatelné součástky a zejména hradlová pole jsou velmi důležitými prvky dnešní elektroniky. Díky nim si kdokoli může vytvořit „vlastní“ integrovaný obvod přizpůsobený konkrétní aplikaci s minimálními náklady. Když ale člověk sleduje české odborné časopisy (jak papírové tak i elektronické), vypadá to jako by nejpokročilejší používané číslicové obvody byly osmibitové mikrokontroléry. Přitom využití programovatelných logických obvodů by bylo často mnohem výhodnější a efektivnější.

Všechny číslicové programovatelné součástky se souhrnně označují PLD (Programmable Logical Device). Programovatelné součástky je možné podle vnitřní struktury rozdělit do tří skupin. První skupinu jsou klasické PLD, druhou komplexní PLD a do třetí skupiny patří obvody typu FPGA [15].

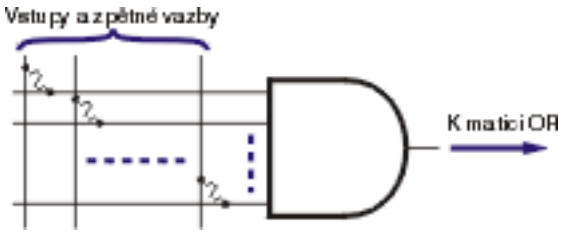
4.1 Klasické PLD

Obvody této kategorie jsou charakteristické vnitřní strukturou podle obr 4.1.



Obr. 4.1: Struktura PLD

Každá vodorovná čára v programovatelné matici AND představuje vždy jedno součinové hradlo. Na vstupy každého hradla lze připojit "libovolnou" kombinaci vstupních signálů, zpětných vazeb a jejich negací. Počet vstupů každého součinového hradla je však omezen. Zapojení jednoho součinového hradla je znázorněno viz. obr. 4.2.



Obr. 4.2: Detail struktury PLD

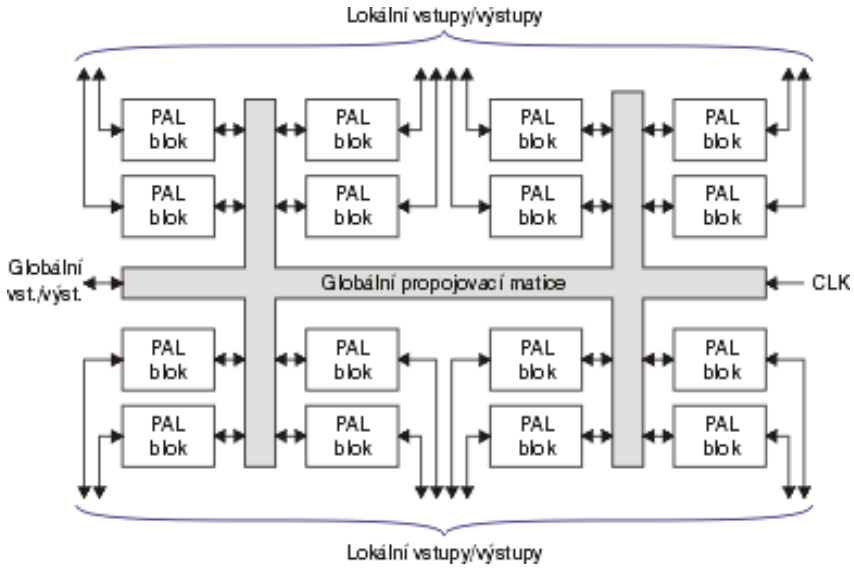
Vlnovky na tomto obrázku představují programovatelné spínače. Jejich realizace závisí na výrobní technologii obvodu. Například u bipolárních obvodů se jednalo o jakousi pojistku, která se při programování obvodu "přepálila" proudovým impulsem. V technologii CMOS jsou spínače realizovány stejnými principy jako u pamětí PROM, EPROM nebo EEPROM. Složitější obvody z kategorie FPGA mívají často spínače řízeny statickou pamětí RAM [15]. Do kategorie klasických PLD je možné zařadit i obvody následujících typů:

a) *PAL* – obvody typu PAL (Programmable Array Logic) mají strukturu podle výše uvedených obrázků. Některé starší typy neměly například výstupní registry, takže byly vhodné spíše pro kombinační logiku. Zástupci této kategorie jsou obvody PAL, GAL atd.

b) *PLA* - obvody typu PLA (Programmable Logic Array) mají obecnější strukturu než PAL na horním obrázku. Mají totiž programovatelnou nejenom matici logických součinů, ale i následující matici logických součtů.

4.2 Komplexní PLD

Klasické obvody PLD mají velmi omezené prostředky, takže umožňují realizovat pouze jednodušší funkce. Proto výrobci začali sdružovat více takovýchto obvodů na jednom čipu spolu s nutnými prostředky pro propojení. Takovéto obvody se většinou označují jako CPLD (Complex Programmable Logical Devices), česky komplexní programovatelné logické obvody. Typická struktura CPLD je znázorněna na obrázku obr. 4.3.

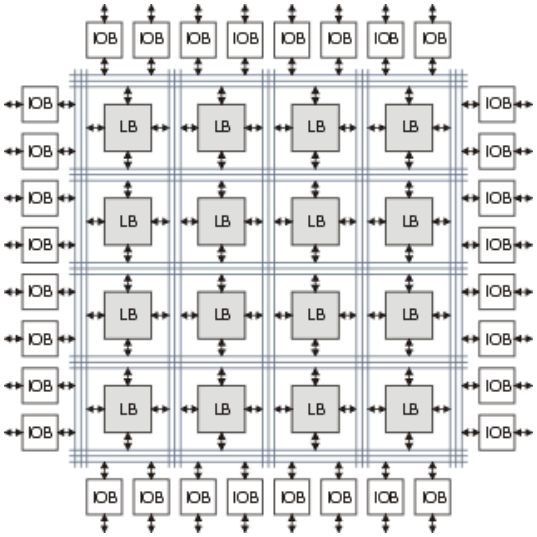


Obr. 4.3: Blokové schéma CPLD

Každý výrobce CPLD používá trochu jinou interní strukturu obvodů, ale většinou vychází z tohoto schématu [1], [15]. CPLD od různých výrobců se obvykle liší v provedení bloků vlastní programovatelné logiky, i když většinou vychází z klasické struktury PAL.

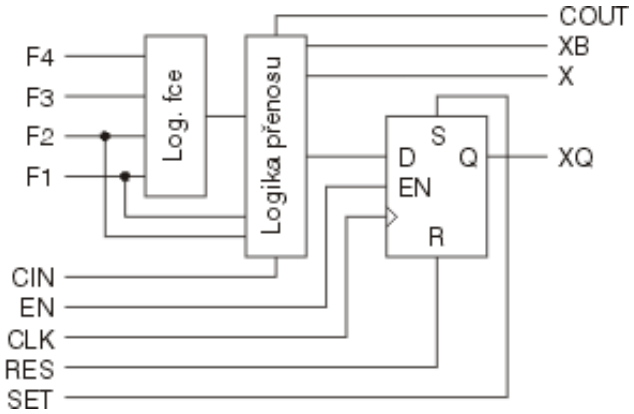
4.3 Obvody FPGA

Obvody typu FPGA (Field Programmable Gate Array) mají z programovatelných obvodů nejobecnější strukturu a obsahují nejvíce logických hradel [15]. Současné největší obvody FPGA obsahují až 6 milionů ekvivalentních hradel. Používají se klasická dvouvstupová hradla NAND. Typickou strukturu obvodu FPGA znázorňuje obr. 4:4.



Obr. 4.4: Blokové schéma FPGA

Bloky označené IOB (Input/ Output Block) představují vstupně/ výstupní obvody pro každý vstupně/ výstupní pin FPGA. Tyto bloky obvykle obsahují registr, budič, multiplexer a ochranné obvody. Bloky LB (Logic Block) představují vlastní programovatelné logické bloky. Všechny bloky mohou být různě propojeny globální propojovací maticí. Nejpoužívanější struktura konfigurovatelného logického bloku je znázorněna na následujícím obrázku.



Obr. 4.5: Schéma zapojení bloku IOB

FPGA obvykle umožňují propojit některé signály logických bloků přímo se sousedními bez nutnosti využívat globální propojovací matici. Takovéto spoje mají mnohem menší zpoždění a umožňují tak realizovat například rychlé obvody pro řízení přenosu, což je nezbytné pro sčítačky nebo násobičky.

Kromě bloků znázorněných na předchozích obrázcích integrují výrobci do FPGA další prvky. Většina moderních FPGA obsahuje několik bloků rychlé

synchronní statické paměti RAM. Velmi často obvody FPGA obsahují PLL (Phase Locked Loop) nebo DLL (Delay Locked Loop) pro obnovení charakteristik hodinového signálu, případně pro násobení nebo dělení jeho frekvence. Nejnovější hradlová pole často obsahují bloky vhodné pro vytváření složitých systémů pro číslicové zpracování signálů jako jsou například hardwarové násobičky nebo dokonce mikroprocesory.

5 Vývojové prostředky

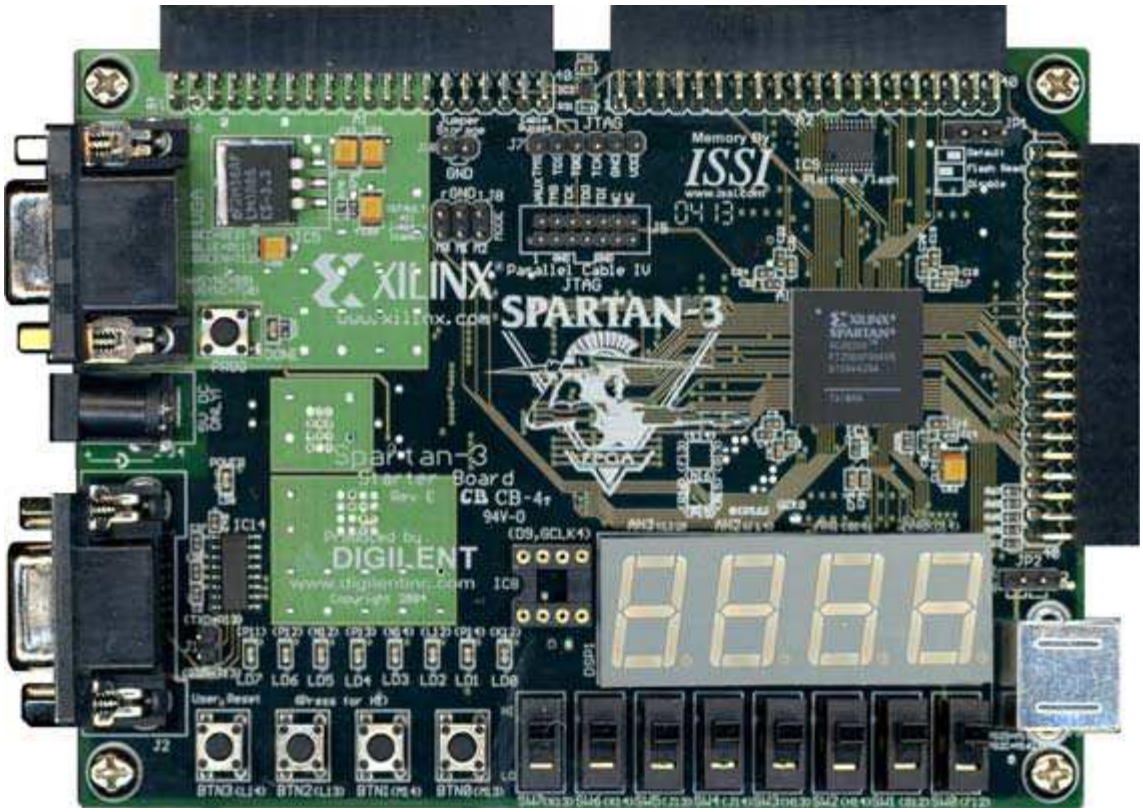
Pro vývoj aplikací s programovatelnými logickými obvody existuje několik návrhových systémů. Pro vývoj je nutné použít minimálně dvou nástrojů. Prvním je nástroj pro syntézu, který převede textový popis návrhu v některém HDL jazyce na netlist využívající obecné logické bloky. Druhý nástroj zajistí konverzi obecného netlistu na netlist využívající prostředky konkrétního logického obvodu a zajistí jejich "optimální" rozmístění a propojení. Nástroje pro rozmístění a propojení obvykle nabízejí výrobci programovatelných hradlových polí [15]. Prostředky pro syntézu však nabízejí i jiné firmy. Firma Xilinx nabízí pro FPGA s menší hustotou logiky vývojový systém ISE WebPACK, který je zadarmo. Tento vývojový systém pro FPGA firmy Xilinx je omezenou verzí jejich kompletního systému. Omezení se však týká pouze velikosti hradlových polí pro které je možno prostředí použít. Systém WebPACK umožňuje zadání návrhu v některém z HDL jazyků Verilog nebo VHDL, případně i pomocí schématu. Prostředí podporuje např. následující obvody Xilinx:

- FPGA Spartan-II
- FPGA Spartan-IIE (max XC2S300E)
- FPGA Spartan-3 (XC3S200)
- FPGA Virtex-E (max XCV300E)
- FPGA Virtex-II (max XC2V250)
- FPGA Virtex-II Pro (max XC2VP2)
- CPLD CoolRunner2
- CPLD XC9500/ XL/ XV
- CPLD CoolRunner XPLA3

5.1 Vývojová deska SPARTAN-3

Jak již bylo zmíněno v rozboru řešení bakalářské práce, původně používaná vývojová deska X2C-XL s logickými obvody Xilinx CoolRunner II XC2C256 CPLD a Xilinx XC9572XL CPLD byla nahrazena platformou SPARTAN-3 nesoucí obvod FPGA XC3S200FT256. Důvodem byla ta skutečnost, že X2C-XL neobsahovala sériový port rozhraní RS-232.

Spartan-3 poskytuje silnou, celistvou vývojovou platformu pro návrhy do FPGA obvodů firmy Xilinx. Vyznačuje se počtem 200000 ekvivalentních hradel umístěných na přístrojové desce a dvěma velkými paměťovými moduly. Je plně programovatelná pomocí standardu JTAG, programovatelná paměť ROM je energeticky nezávislá. Vývojová deska je slučitelná s veškerými verzemi Xilinx ISE nástrojů, včetně volných WebPacků [17].



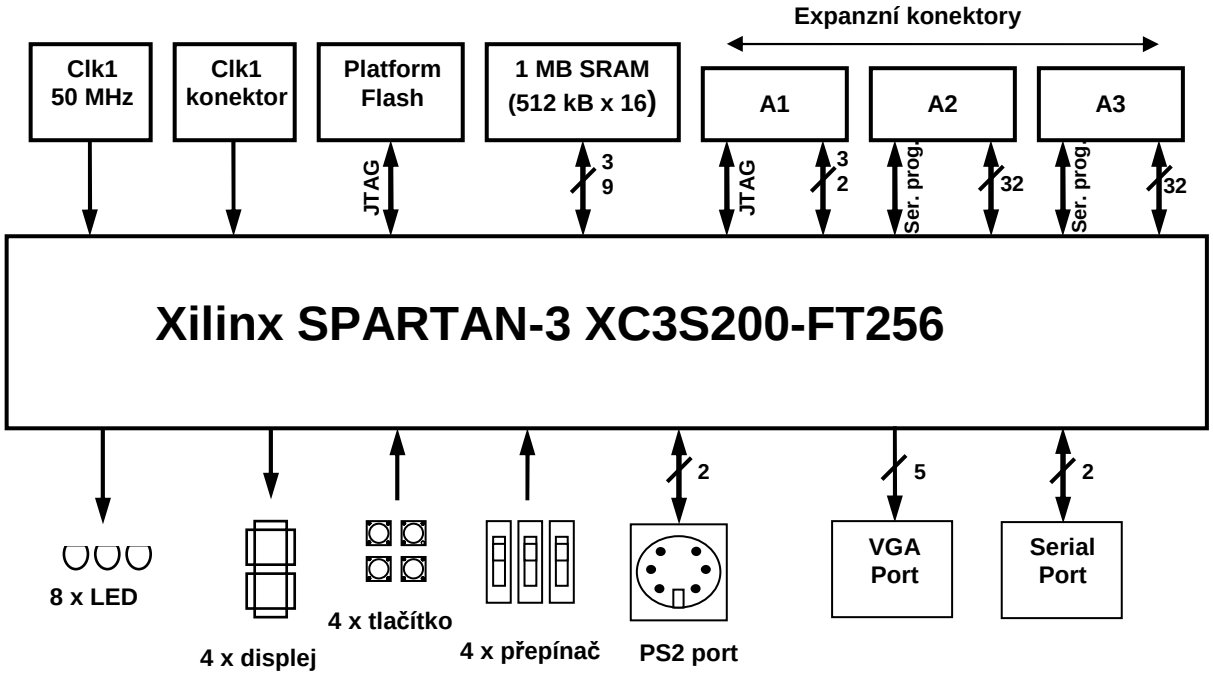
Obr. 5.1: Ilustrační foto vývojové desky SPARTAN-3

5.1.1 Vlastnosti

- 200000 ekvivalentních hradel implementovaných v FPGA spolu s 18-ti bitovými násobiteli, 216 kbit blokové paměti RAM, pracuje do kmitočtu až 500 MHz, to vše v označení XC3S200FT256
- 2 Mbit Flash paměť na samotné desce
- 8 posuvných přepínačů, 4 mikro-tlačítka, 8 LED diod, 4 číslice sedmisegmentového displeje
- tři 40-ti pinové vstupně/ výstupní konektory
- tři regulátory napětí (3,3 V, 2,5 V a 1,2 V)
- vstup pro JTAG3 programovací kabel slučitelný s paralelním P4 kabelem a nástrojem MultiPRO firmy Xilinx

Další vlastnosti a součásti vývojové desky

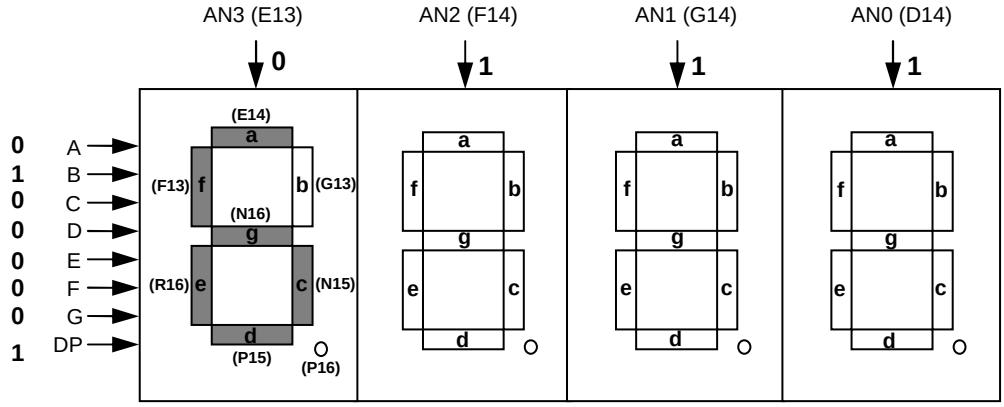
- až 173 vstupně/ výstupních signálů definovaných uživatelem
- 3 bitový, 8-mi barevný VGA port pro zobrazovací zařízení
- 9-ti pinový sériový port RS-232
- PS/2 port pro připojení klávesnice nebo myši
- krystalový oscilátor 50 MHz
- konektor pro pomocný externí krystal
- jumper propojku pro nastavení konfiguračního módu FPGA
- vstupní konektor pro pevné AC napájecí napětí 5 V
- výkonová indikační LED pro indikaci připojení k síti



Obr. 5.2: Blokové schéma SPARTAN-3

5.1.2 Čtyř místný sedmi-segmentový displej

Displej dokáže zobrazit až 4 různé znaky a je řízen vstupně/ výstupními piny FPGA zařízení. Každý znak (v tomto případě číslice) se skládá z osmi segmentů, které lze rozsvěcovat pomocí signálů k nim přiřazeným. Jednotlivé znaky jsou připojeny na vstupní řídící anodu [18]. Detailní zobrazení displeje je viz. obr. 5.3. V závorkách je uvedeno označení příslušných pinů, které propojují displej s FPGA. Pro osvětlení segmentu musí být jeho příslušný řídící signál v nízké úrovni stejně tak, jako přidružený signál řídící anody znaku, ve kterém je signál rozsvěcován.



Obr. 5.3: Příklad nastavení signálů pro rozsvícení znaku

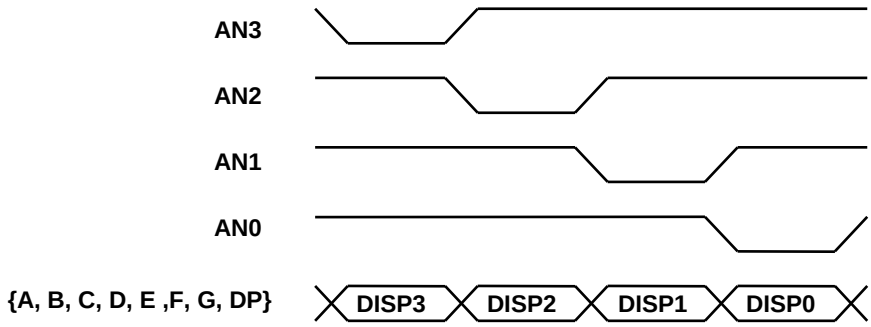
	Označení	FPGA pin
Segment	A	E14
	B	G13
	C	N15
	D	P15
	E	R16
	F	F13
	G	N16
	DP	P16
Řídící anoda	AN3	E13
	AN2	F14
	AN1	G14
	AN0	D14

Tab. 5.1: Propojení FPGA pinů se sedmisegmentovým displejem

	Vstupní signály displeje						
Znak	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
b	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
d	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

Tab. 5.2: Charakteristické úrovně signálů pro určité znaky

Řídící signály pro znaky LED displeje jsou časově multiplexovány, aby byla zobrazena celková řada znaků (v našem případě celková hodnota) složená z jednotlivých znaků na displeji jak je ukázáno na obr. 5.4. Tyto znaky jsou postupně rozsvěcovány s takovou frekvencí, že je lidský mozek vnímá tak, jako by byly rozsvíceny současně.



Obr. 5.4: Přepínání mezi segmentovými displeji

Celkem je tedy potřeba k ovládání displeje 12 řídících signálů. Čtyři pro aktivaci jednotlivých segmentových displejů a 8 pro ovládání segmentů.

5.1.3 Posuvné přepínače

Vývojová deska jich obsahuje celkem 8 (označení SW0 až SW7). Vypínače jsou připojeny k FPGA pinům, jak ukazuje tab. 5.3.

Přepínač	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
FPGA pin	K13	K14	J13	J14	H13	H14	G12	F12

Tab. 5.3: Propojení FPGA pinů s posuvnými přepínači

Pokud je vypínač v horní poloze (zapnuto), je k němu příslušný pin připojen k napětí. To simuluje vysokou úroveň signálu. Když je vypínač v dolní poloze (vypnuto), je pin uzemněn. Vypínač po změně polohy ještě asi 2 ms vykazuje mechanické kmity, které nejsou žádané a ve spoustě návrhů mohou způsobit problémy [18].

5.1.4 Tlakové mikrospínače

Deska vlastní 4 tyto součástky. Spínače jsou označené BTN0 až BTN3 a jsou opět přiřazeny k jednotlivým pinům FPGA zařízení.

Mikrospínač	BTN3	BTN2	BTN1	BTN0
FPGA pin	L14	L13	M14	M13

Tab. 5.4: Propojení FPGA pinů s tlačítkovými mikrospínači

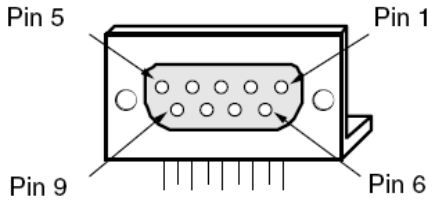
Stlačením mikrospínače je na daném pinu generována vysoká úroveň. Po uvolnění součástky je definována nízká úroveň. Spínač BTN3 je na desce používán také jako resetovací spínač. Pokud tento spínač není uživatelem definován jinak, funguje právě jako reset vývojové desky SPARTAN-3 [18].

5.1.5 Port RS-232

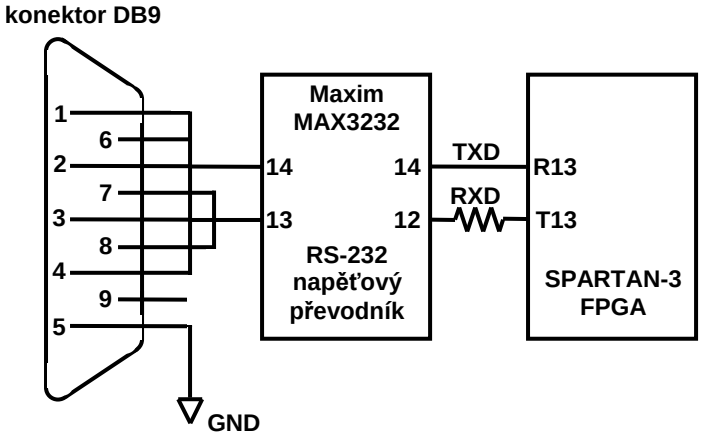
Je na desce prezentován jako zásuvka konektoru DB9. Port odpovídá rozhraní, které je dostupné na většině počítačů. Pro spojení SPARTAN-3 vývojové desky s počítačem se využívá standardní přímý sériový kabel.

Na desce je přítomný i konvertor napětí firmy Maxim MAX3232, který převádí logické úrovně přicházející od FPGA zařízení na napěťové úrovně vhodné pro

zmíněný port. Podobně toto zařízení přeměňuje logické úrovně v opačném směru. Odpor zapojený mezi výstupní pin MAX3232 a pin zařízení FPGA (RXD) chrání proti náhodným přeskokům logických úrovní. Konektor nepočítá s použitím signálů DCD, DTR, DSR, RTS a CTS. Konektory pro tyto signály jsou navzájem propojeny [18].



Obr. 5.5: Konektor sériového portu DB9



Obr. 5.6: Blokové schéma sériového portu

Signál	FPGA pin
RXD	T13
TXD	R13
RXD - A	N10
TXD - A	T14

Tab. 5.5: Propojení FPGA pinů se sériovým rozhraním

5.1.6 Zdroj hodinového signálu

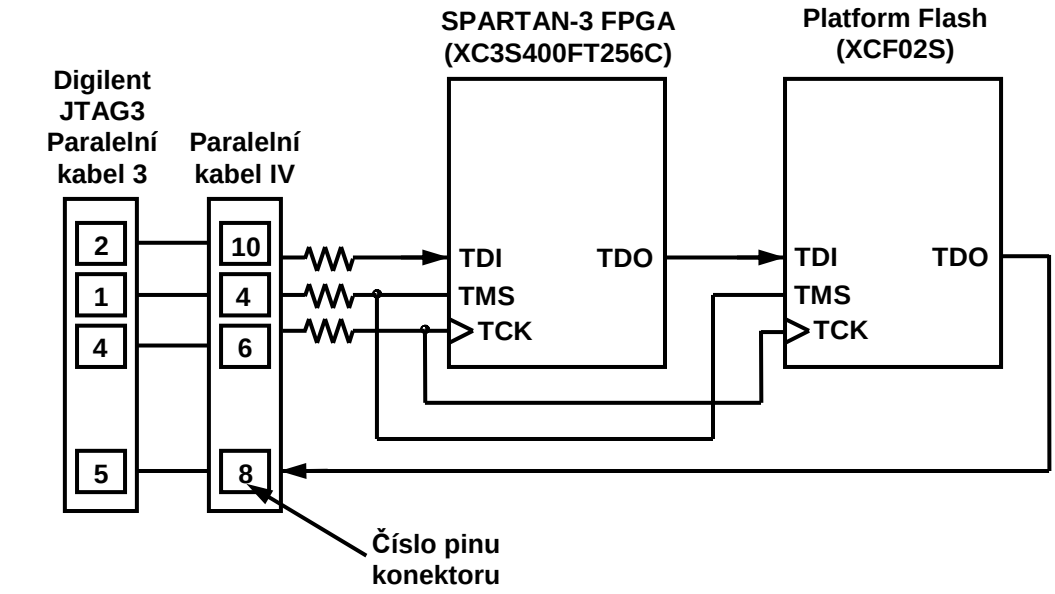
Deska obsahuje vlastní oscilátor Epson SG 8002JF, který dodává signál o frekvenci 50 MHz. Dále také vlastní konektor pro připojení externího oscilátoru [18].

Použitý oscilátor	FPGA pin
integrovaný 50 MHz	T9
konektor pro extení	D9

Tab. 5.6: Propojení FPGA pinů s oscilátorem

5.1.7 JTAG rozhraní

SPARTAN-3 vývojová deska je programovatelná pomocí JTAG rozhraní obsahující Flash zařízení jako součást JTAG řetězce. Na desce jsou přítomny i již zmíněné konektory pro podporované JTAG kabely.

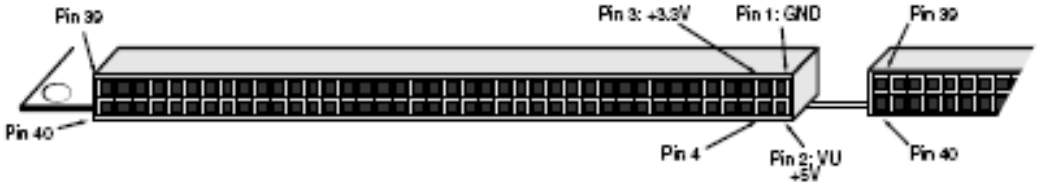


Obr. 5.7: JTAG programovací rozhraní

5.1.8 Expanzní konektory

Vývojová deska obsahuje tři 40-ti pinové expanzní konektory označené A1, A2 a B1. Konektor A1 umožňuje připojení periferního zařízení s maximálně 32 vstupně/ výstupními piny. Konektory A2 a B1 mají ještě o dva piny více. Některé piny jsou sdíleny s dalšími funkcemi, které deska obsahuje.

Piny konektoru jsou očíslovány dle obr. 5.8. Některé piny jsou od výrobce pevně nastavené. Jedná se o piny 1, 2 a 3 každého konektoru. Pin 1 je připojen na zem. Na pinu 2 je vždy stejnosměrné napětí 5 V. Poslední definovaný pin 3 je připojen na stejnosměrné napájecí napětí 3,3 V [18].



Obr. 5.8: Expanzní konektor

Příklad přiřazení pinů FPGA zařízení pro konektor A1 ukazuje tab.5.7.

Název	FPGA pin	Pin konektoru		FPGA pin	Název
GND		**1**	**2**	*VU (5 V)*	
V_{CCO} *(3,3 V)*	V_{CCO}	**3**	**4**	N8	ADR0
DB0	N7	**5**	**6**	L5	ADR1
DB1	T8	**7**	**8**	N3	ADR2
DB2	R6	**9**	**10**	M4	ADR3
DB3	T5	**11**	**12**	M3	ADR4
DB4	R5	**13**	**14**	L4	ADR5
DB5	C2	**15**	**16**	G3	WE
DB6	C1	**17**	**18**	K4	OE
DB7	B1	**19**	**20**	P9	CSA
LSBCLK	M7	**21**	**22**	M10	MA1-DB0
MA1-DB1	F3	**23**	**24**	G4	MA1-DB2
MA1-DB3	E3	**25**	**26**	F4	MA1-DB4
MA1-DB5	G5	**27**	**28**	E4	MA1-DB6
MA1-DB7	H4	**29**	**30**	H3	MA1-ASTB
MA1-DSTB	J3	**31**	**32**	J4	MA1-WRITE
MA1-WAIT	K5	**33**	**34**	K3	MA1-RESET
MA1-INT	L3	**35**	**36**	*JTAG izolace*	
TMS	C13	**37**	**38**	*FPGA JTAG TCK*	*TCK*
TDO-ROM	*JTAG DTO*	**39**	**40**	*konektor J7, pin 3*	*TDO-A*

Tab. 5.7: Propojení pinů expanzního konektoru s FPGA

6 Jazyk VHDL

VHDL je jazyk vysoké úrovně určený pro popis hardware. Jazyk představuje akronym - VHSIC Hardware Description Language. VHDL jazyk byl vytvořen pro účely návrhu, modelování a simulace velmi rozsáhlých číslicových obvodů a systémů. Obrovskou výhodou tohoto jazyka je velká nezávislost číslicového systému popsaného jazykem VHDL na jeho realizaci a variabilitě vyjádření. V současnosti je nejvíce používaný standart syntaxe jazyka VHDL standart IEEE Std 1076 – 1993.

Při programování pomocí jazyka VHDL musíme brát v potaz, že popisujeme číslicový systém, který později budeme realizovat jako hardware v logickém obvodu. To znamená, že vytvořenou posloupnost příkazů implementujeme do hradel a klopných obvodů syntézou tohoto kódu. S výjimkou testovacího programu nebo programů určených k simulacím v simulátoru musí být vytvořené posloupnosti příkazů syntetizovatelné. Testovací program (angl. testbench) je pak program, kterým zadáváme vstupní signály pro námi vytvořený program a dále pak jím ověřujeme jeho správnou funkčnost. Funkčnost programu můžeme sledovat i graficky pomocí simulace testovacího programu spolu se zdrojovým kódem. Zdrojový kód testbenche pak není nutné syntetizovat, neboť je používán z převážné části pouze pro účely simulace [2].

Vývoj jazyka byl zahájen v roce 1981 v rámci výzkumného projektu ministerstva obrany Spojených států amerických. Od té doby byl jazyk postoupen firmám jako IBM, Intermetrics atd. Tyto firmy měly za úkol vyvíjet a standardizovat jazyk. Ten do dnešní doby prošel zatím třemi revizemi a jedním rozšířením.

1981: zahájení vývoje jazyka, projekt VHSIC U.S. DoD
1983 – 1985: vývoj základů jazyka firmami (IBM, Intermetrics, TI)
1986: postoupení jazyka organizaci IEEE (angl. Institute of Electrical and Electronics Engineers)
1987: publikování standartu IEE Std 1076 – 1987 (VHDL – 87)
1994: publikování standartu IEE Std 1076 – 1993 (VHDL – 93)
1999: publikování standartu IEE Std 1076.1 – 1999 (VHDL – AMS)
2000: publikování standartu IEE Std 1076 – 2000 (VHDL – 2000)
2002: publikování standartu IEE Std 1076 – 2002 (VHDL – 2002)

6.1 Základní vlastnosti VHDL jazyka.

- Jedná se o otevřený standart. Není k jeho použití potřebná licence.
- Variabilita umístění. Zkonstruovaný program může být umístěn v různých logických obvodech např. PLD, CPLD, FPGA.
- Možnost konstrukce programu, aniž bychom věděli, do kterého logického obvodu bude program zakomponován.
- Přenositelnost kódu. Tentýž zdrojový kód může být použit v různých cílových obvodech, může být použit při simulaci navržených obvodů i při jejich syntéze.
- Zdlouhavost vytvořeného kódu. Jazyk VHDL není stručný. Často se určité bloky v programu opakují

6.2 Způsob sestavení modelu

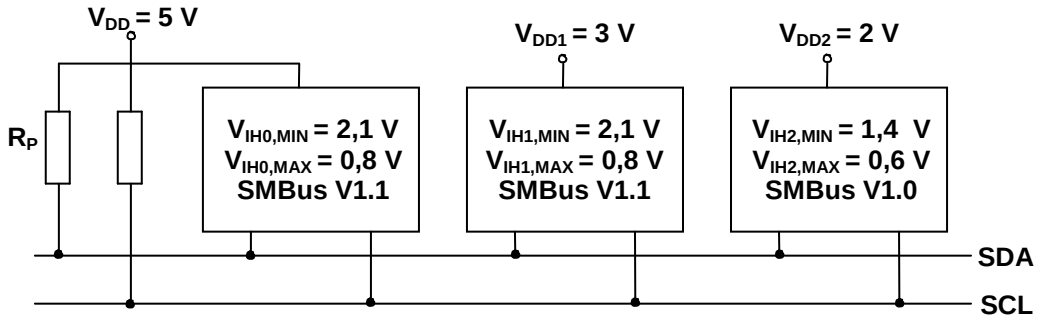
- **zdola nahoru** (bottom – up): nejdříve se vytvoří jednotlivé bloky programu (tzv. podprogramy) a ty se pak skládají do sebe, konstrukce je hierarchická
- **shora dolů** (top – down): nejdříve se definuje program jako celek a v něm se pak vytvoří bloky podprogramů s jednoduchými zdrojovými kódy programů, konstrukce je taktéž hierarchická
- **model plochého typu** (float): neobsahuje členění na podprogramy, představuje jeden blok příkazů

Struktura programu v jazyce VHDL je rozdělena na dvě části. První je tzv. deklarace entity, která definuje vstupy a výstupy napsaného programu. Druhá část se nazývá tělo architektury a obsahuje vlastní zdrojový kód programu zapsaného v jazyce VHDL [1], [2].

7 SMBus

SMBus je dvoudrátová sběrnice. To znamená, že na ní může být připojeno více než jedno zařízení schopné jejího ovládání “master“ [16]. Toto zařízení zavádí přenos dat a poskytuje signály hodin. Slave zařízení může přijmout data poskytovaná masterem nebo mu může data poskytnout. Jelikož se může více než jedno zařízení připojené ke sběrnici pokusit převzít kontrolu nad sběrnicí “jako master“, SMBus poskytuje mechanismus arbitráže, které vychází z I_2C a spoléhá se na uzlový součin všech zařízení připojených ke sběrnici. Jestliže dva nebo více masterů se pokusí umístit informaci na sběrnici, pak první produkuje "1". Ostatní produkují "0", upouští arbitráž a musí uvolnit sběrnici. Signály hodin během arbitráže jsou uzlovým součinem kombinací všech hodinových signálů poskytovaných zařízeními, které ovládají sběrnici. Na sběrnici mohou být umístěna zařízení různých rychlostí. SMBus verze 1.1 může být napájena napětím 3 V – 5 V +/- 10 %. Zařízení mohou být napájena sběrnicí V_{DD} nebo svým vlastním zdrojem energie.

Následující diagram ukazuje implementaci příkladu 5 V SMBus se zařízeními napájeným sběrnicí (V_{DD}) a interními zařízeními napájeným jejich vlastním zdrojem.



Obr. 7.1: SMBus topologie

Specifickým příkladem je sběrnice se zařízeními napájenými V_{DD1} = 3 V. Jedná se o SMBus verzi 1.1. Sběrnice se zařízeními napájenými V_{DD2} = 2 V je SMBus verze 1.0. Je-li V_{DD} sběrnice 3 – 5 V +/- 10 % , pak může být SMBus zařízení napájené přímo ze sběrnice [16]. Obě SCL a SDA linky jsou připojeny na kladné napájecí napětí přes rezistor, zdroj proudu nebo jiné podobné obvody. Když je sběrnice volná, obě linky jsou ve vysoké úrovni.

7.1 Platnost dat

SMBus používá stanovené napěťové úrovně 0,8 V a 2,1 V na definovaní logické "0" a logické "1" na příslušné sběrnici.

Data na SDA (datové lince) musí být stabilní během vysoké úrovně periody hodinového signálu. Data mohou měnit stav jen když SCL (linka hodinových signálů) je na nízké úrovni.

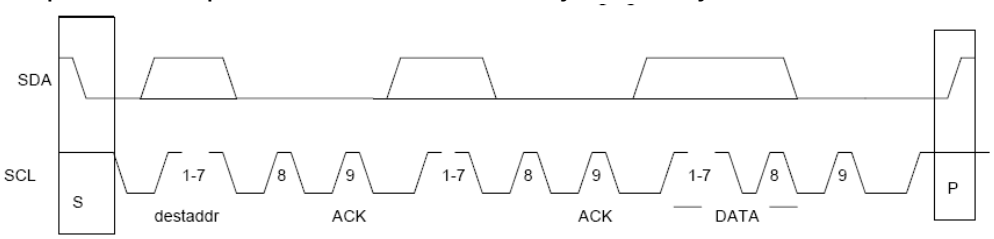
7.2 Start a stop podmínka

Stejně jako u I_2C, dvě jedinečné sběrnicové situace definují podmínky START a STOP. Jestliže vysoká úroveň přechází na nízkou v SDA lince, když SCL je na vysoké úrovni, dochází ke START podmínce. Pokud se v SDA lince změní nízká úroveň na vysokou a pokud je SCL na vysoké, dochází ke STOP podmínce. START

a STOP podmínky jsou vždy generovány mastrem sběrnice. Po START podmínce je sběrnice považována za zaneprázdněnou. Sběrnice se stane znovu volná po STOP podmínce nebo zůstanou-li obě linky SCL a SDA na vysoké úrovni po dobu delší než 50 μs viz. [16].

7.3 Přenosy dat na SMBus

Data přenášená po sběrnici SMBus udržují následující tvar viz. obr. 7.2.



Obr. 7.2: SMBus formát dat

Po START podmínce (S) master umístí sedmi-bitovou adresu na slave zařízení, to pak čeká na adresu ze sběrnice. Adresa je dlouhá 7 bitů. Za ní následuje 8. bit, který indikuje řízení přenosu dat (R / _W); „0“ indikuje přenos (WRITE) a „1“ indikuje žádost dat (READ). Následují data. Každý byte dat se skládá z 8 bitů. Všechny byty přenesené sběrnicí musí být následovány potvrzovacím bitem. Byty jsou přenášeny s nejvýznamnějším bitem (MSB) napřed. Slave zařízení nemusí potvrdit byte dat v následujících situacích viz. [16]:

- Zařízení slave je zaneprázdněné prováděním úkolu nebo žádaná data jsou nedostupná. Master na detekci NACK podmínky musí generovat STOP podmínku k ukončení přenosu. Alternativa: zařízení slave může prodloužit nízkou úroveň periody hodin uvnitř limitu této specifikace k tomu, aby dokončil své úkoly a pokračoval v přenosu.
- Zařízení slave detekuje chybný příkaz nebo chybná data. V tomto případě zařízení slave nesmí potvrdit přijatý byte. Master na detekci této podmínky musí vygenerovat STOP podmínku a opakovat transakci.
- Jestliže master - přijímač byl zapojen do přenosu, musí signalizovat konec dat k slave - vysílači a nesmí generovat potvrzení za posledním bytem odeslaným slave zařízením. Slave - vysílač musí uvolnit datové linky a dovolit masterovi vytvořit STOP podmínku.

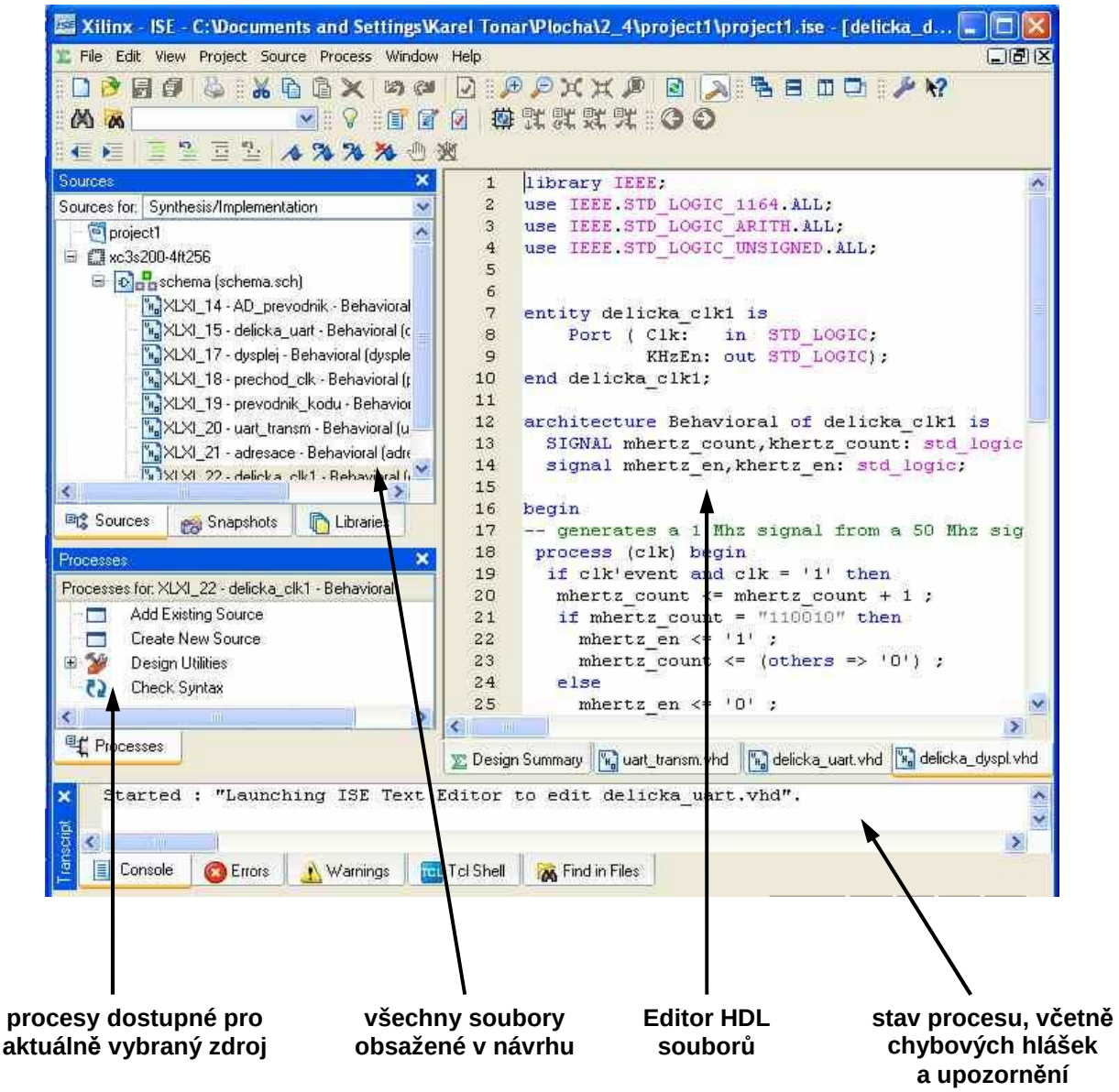
Přenos dat je vždy ukončen STOP podmínkou (P) generovanou masterem. SMBus realizuje několik komunikačních formátů, které jsou podmnožinou formátů pro I_2C.

8 Návrhový systém XILINX ISE 9.2i

Xilinx ISE je integrované vývojové prostředí pro návrh digitálních logických obvodů pro programovatelné součástky FPGA a CPLD firmy Xilinx. Skládá se z mnoha programů pro návrh, syntézu, implementaci, analýzu a simulaci obvodů.

8.1 Project Navigator

Hlavní částí je tzv. Project Navigator, který poskytuje jednotné rozhraní ke všem ostatním programům, a také jednoduchý způsob pro správu souborů projektu [19].



procesy dostupné pro aktuálně vybraný zdroj

všechny soubory obsažené v návrhu

Editor HDL souborů

stav procesu, včetně chybových hlášek a upozornění

Obr. 8.1: Xilinx Project Navigator

Plocha Navigatoru je dělena na viz. obr. 8.1:

- *Sources* – obsahuje všechny prvky návrhu, umožňuje přidávat a odebírat nové soubory projektu a měnit jejich nastavení.

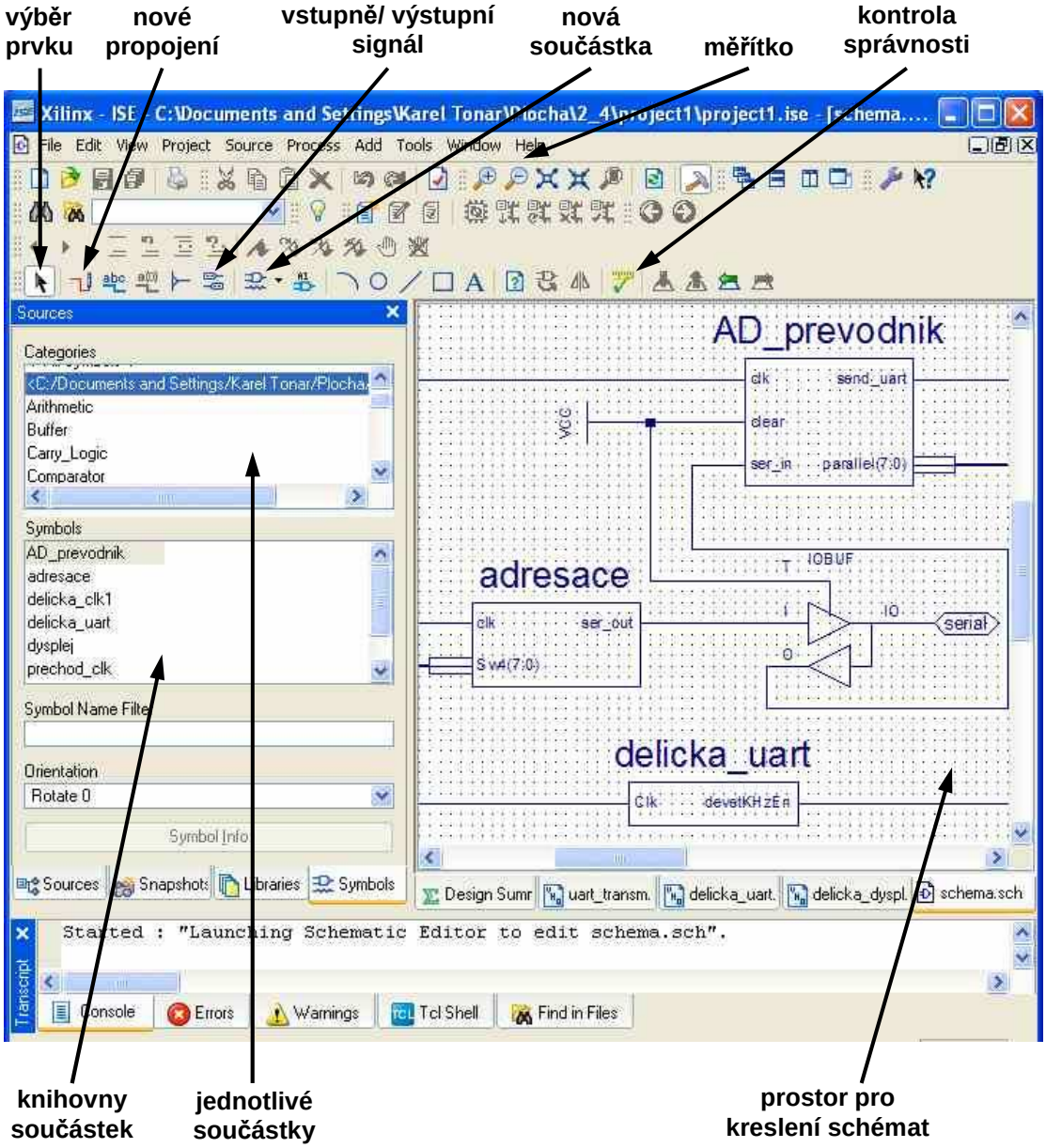
- *Processes* – seznam akcí, které je možno s vybraným souborem v Sources provádět. Pro různé soubory jsou na výběr různé akce.
- *Console* – ukazuje stav probíhajícího procesu, statistiky a chybové zprávy.
- *HDL Editor* – textový editor HDL souborů.

V Sources zakládáme projekty, ve kterých je návrh sloučen, a vkládáme do nich nové soubory. Pro programování obvodu FPGA použitého v bakalářské práci byly použity následující typy souborů:

- Schematic – obvod zadaný schématem z hradel, klopných obvodů a dalších prvků. Od interní reprezentace ve VHDL je uživatel odstíněn.
- Test Bench Waveform – testování podle stimulů.
- Soubor ucf – určuje přiřazení vstupně/ výstupních signálů k jednotlivým pinům vývojové desky.

V části Processes spouštíme jednotlivé úlohy pro vložené soubory, jako je návrh činnosti obvodu, syntéza, implementace, simulace a vlastní programování.

8.2 Editor schémat



Obr. 8.2: Xilinx ECS

Editor schémat má podobu plochy, na kterou umísťujeme součástky a propojujeme je pomocí vodičů. Pro práci na bakalářské práci potřebujeme jen několik málo akcí:

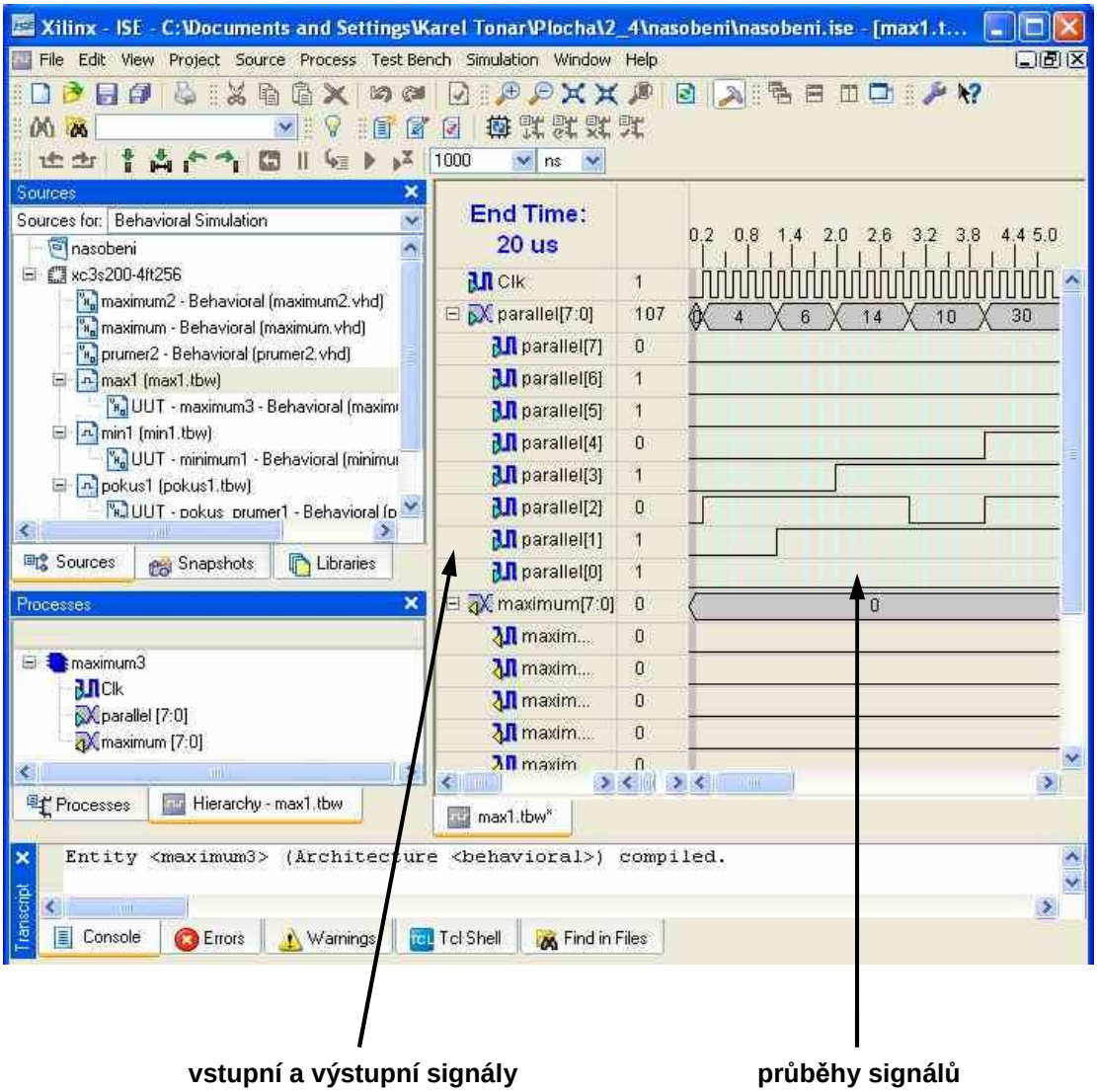
- *Symboly* – vkládáme součástky z knihovny.
- *Propojení* – propojení kreslíme přímo do schématu od jedné součástky ke druhé.
- *Vstupy a výstupy* – ke vstupům a výstupům jsou přiřazeny příslušné I/O značky.

Program je po registraci možné získat v omezené verzi zdarma z internetových stránek výrobce. Omezení se vztahují především na typy podporovaných obvodů, ovšem pro návrh FPGA je instalace plně dostačující.

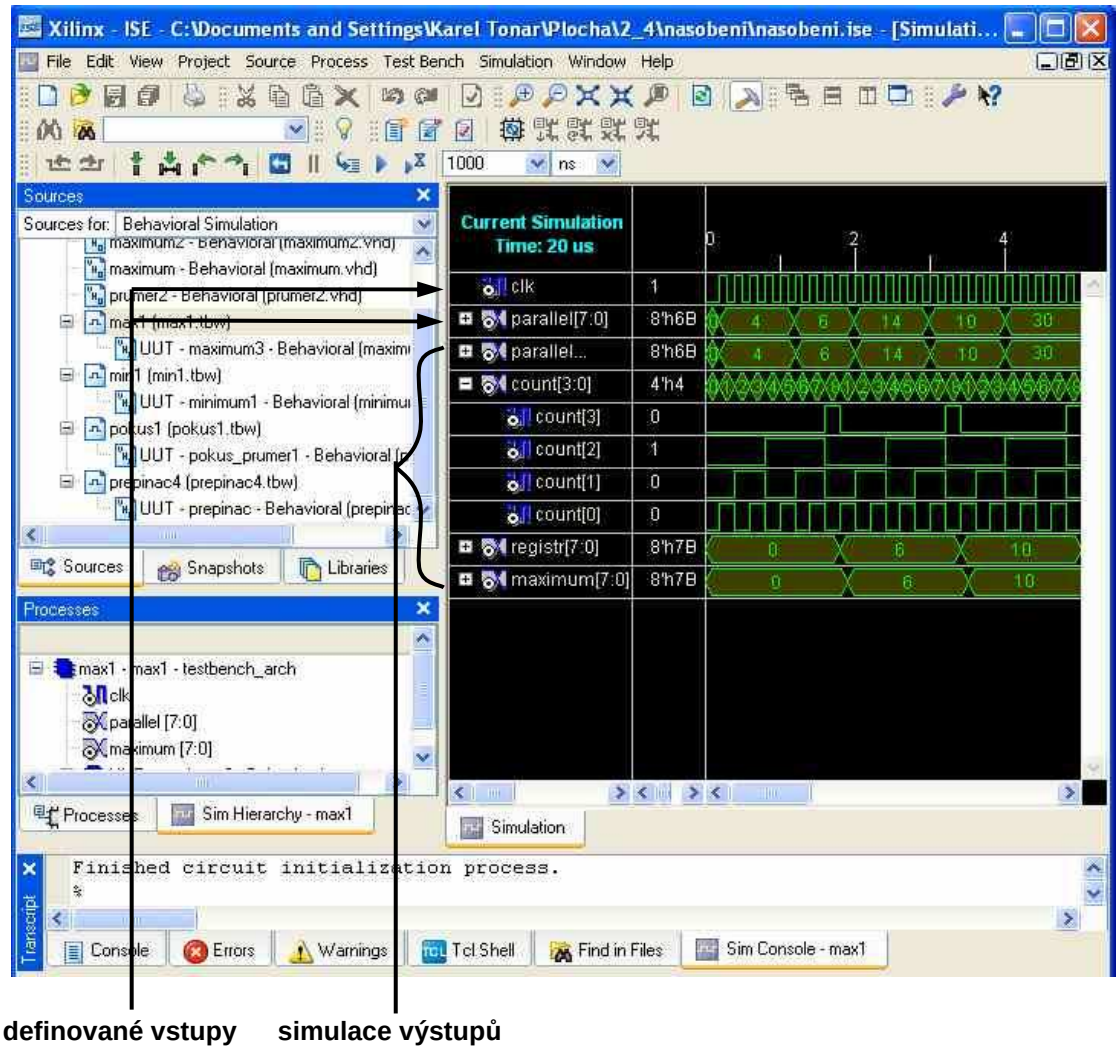
8.3 ISE Simulator

Simulator je integrované prostředí ISE Webpack pro simulaci obvodů popsaných ve VHDL nebo Verilogu. Tato editace obsahuje data o všech obvodech dostupných v ECS a umožňuje jejich bezproblémovou simulaci [19]. Program je integrován do ISE a odstiňuje uživatele od jeho ovládání pomocí příkazů a všech rutinních postupů jako jsou: vytváření projektů, vkládání souborů a knihoven, kompilaci a spouštění simulace.

K simulaci jsou potřeba vstupní stimuly obvodu. Vstup lze určit z vytvořeného zdroje Test Bench Waveform vloženém do projektu. Graficky se v něm definuje průběh vstupních signálů, případně i očekávané výstupní hodnoty pro srovnání se simulací. Je možno zadat jednorázovou nebo pravidelnou změnu signálů, např. pro signál hodin. Výstupní simulace je následně vložena do modulu zvaného VHDL Test Bench.



Obr. 8.3: Xilinx Test Bench Waveform



Obr. 8.4: VHDL Test Bench

ISE podporuje ještě další simulátory jako např. ModelSim, Cadence NC-VHDL a Synopsys Scirocco. ModelSim je možné získat bezplatně po registraci na webových stránkách firmy Xilinx.

9 Analog Device AD7414/ AD7415

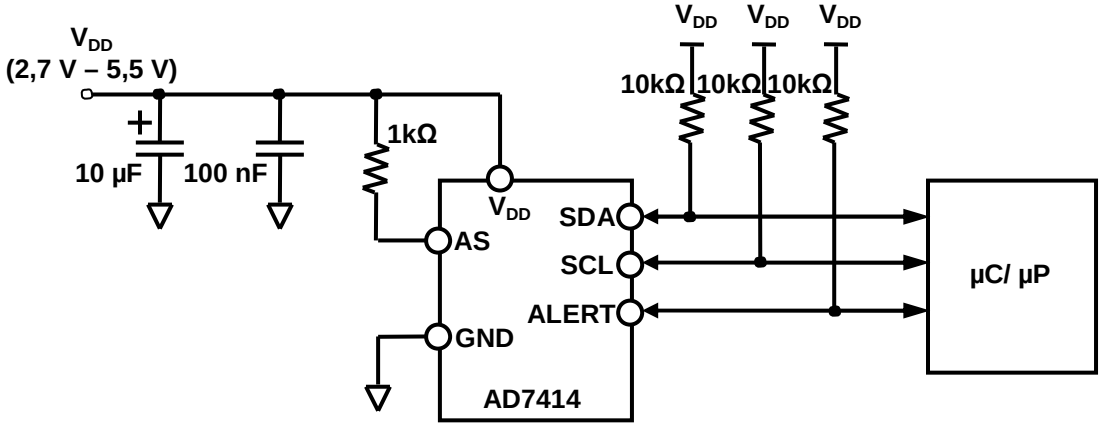
Jedná se o digitální snímače teploty. Čip teplotního snímače dovoluje přesné měření okolní teploty. Analogová hodnota této veličiny je pomocí analogového 10-ti bitového převodníku reprezentována ve formátu dvojkového doplňku. Sériové rozhraní je kompatibilní s I_2C a SMBus. Snímač požaduje napájení v rozmezí 2,5 V až 5,5 V. Rozsah měřených teplot je -40 ℃ až +125 ℃.

9.1 Popis funkce

Měření teploty může být zahájeno dvěma způsoby:

1) První způsob využívá odečítání vnitřních hodin po dobu 800 ms a provedení konverze. Vnitřní oscilátor je jediný obvod, který je napájen mezi dvěma konverzemi a jednou za 800 ms je budící signál odeslaný do zbytku systému. Vnitřní monostabilní klopný obvod je aktivovaný na začátku budícího signálu proto, aby zajistil dostatečný čas na start procesu. Mezi dvěma změřenými hodnotami teploty je MKO typicky 4 µs v tzv. klidovém stavu. Dokončení změny teploty pak obvykle trvá 25 µs. Nová hodnota teploty je nakonec načtena do teplotního hodnotového registru, kde je připravena pro čtení na I_2C rozhraní.
2) Druhý způsob měření teploty je zahájen, když je použit jednorázový příkaz pro snímač. Tento příkaz spočívá v přepsání příslušného bitu v konfiguračním registru zařízení. Konverze teploty pak nastává 4 µs po operaci zápisu bitu, což odpovídá době klidového stavu MKO v prvním případě. Po uplynutí 25 µs je teplotní hodnotový registr naplněn novou hodnotou.

Naměřené teploty jsou porovnávány s hodnotou uloženou v 8-mi bitovém read/ write registru. Toto porovnávání funguje jako termostat a využívá ALERT výstup snímače. Zmíněnou funkci je možno použit pouze pro AD7414, jelikož AD7415 nemá ALERT výstup. Kdyby měření přesáhlo hodnotu uloženou v read/ write registru, ALERT výstup se aktivuje, pokud je tedy povolen v příslušném registru [9].



Obr. 9.1: Typické zapojení snímače

Běžné měření teploty využívá záporného teplotního koeficientu diody, nebo napětí báze - emitor tranzistoru v režimu konstantního proudu. Bohužel, tento postup vyžaduje kalibrování k tomu, aby byl anulován efekt absolutní hodnoty napětí U_{BE},

která je pro každou součástku rozdílná. Způsob užívaný v AD7414/ AD7415 spočívá v měření změn U_{BE} v době, kdy je zařízení ovládáno dvěma různými proudy

$$\Delta V_{BE} = \frac{K \cdot T}{q} \cdot \ln(N). \quad (1)$$

K…… Boltzmannova konstanta
q……. náboj elektronu ($1{,}6 \cdot 10^{-19}$ C)
T…….absolutní teplota ve stupních Kelvina
N……. proudový poměr.

Měření ΔU_{BE} spočívá v přepínání mezi dvěma operačními proudy. Výsledný časový průběh signálu prochází stabilizovaným operačním zesilovačem, který vykonává funkci zesílení a korekce časového průběhu signálu a vytvoření stejnosměrného napětí úměrného U_{BE}. Toto elektrické napětí je měřené analogově digitálním převodníkem. Teplota je odesílaná na výstup ve formátu 10-ti bitového dvojkového doplňku [9].

9.2 Formát výstupních dat

Rozlišovací schopnost A/D převodníku je 0,25 ℃. Tato hodnota odpovídá bitu s nejnižší vahou LSB. Snímač může teoreticky měřit rozsah teplot až do 255 ℃, nejnižší měřitelná teplota je -40 ℃. Proto je teplotní rozsah omezen na -40 ℃ až +125 ℃, jak ukazuje viz. Tab. 7:1.

teplota [℃]	digitální formát dvojkového doplňku [DB9…DB0]
-55	1 100 100 100
-50	1 100 111 000
-25	1 110 011 100
-0,25	1 111 111 111
0	0 000 000 000
0,25	0 000 000 001
10	0 000 101 000
25	0 001 100 100
50	0 011 001 000
75	0 100 101 100
100	0 110 010 000
125	0 111 110 100

Tab. 9.1: Formát výstupní hodnoty

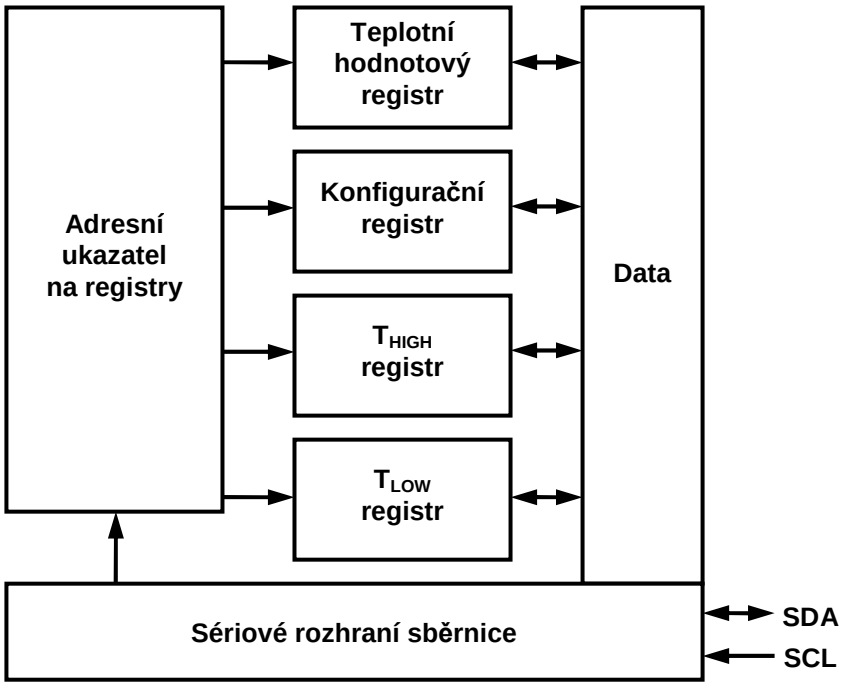
Výpočet analogové hodnoty z formátu dvojkového doplňku je následující:

$$kladná\_hodnota = \frac{data}{4} \quad (2)$$

$$záporná\_hodnota = \frac{data - 512}{4} \quad . \quad (3)$$

9.3 Vnitřní struktura registrů v AD7414

AD7414 má pět vnitřních registrů, jak je ukázáno v obr. 7.2. Čtyři z registrů jsou datové a jeden je ukazatelem na registry.



Obr. 9.2: Struktura registrů

9.4 Adresní ukazatel na registry

Jedná se o 8-mi bitový registr uchovávající adresu, která ukazuje na jeden ze čtyř datových registrů (AD7414) nebo na jeden ze dvou datový registrů (AD7415) [9]. Adresu registru určují pouze 2 bity s nejmenší váhou (P1 a P0) viz. tab. 9.2 – 9.4.

P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	výběr registru	

Tab. 9.2: Struktura adresního ukazatele na registry

P0	P1	registr	možné operace
0	**0**	teplotní hodnotový registr	čtení
0	**1**	konfigurační registr	čtění, zápis
1	**0**	T_{HIGH} registr	čtění, zápis
1	**1**	T_{LOW} registr	čtění, zápis

Tab. 9.3: Možnost naplnění registru pro AD7414

P0	P1	registr	možné operace
0	**0**	teplotní hodnotový registr	čtení
0	**1**	konfigurační registr	čtění, zápis

Tab. 9.4: Možnost naplnění registru pro AD7415

9.5 Konfigurační registr

Je 8-mi bitový registr, který je využíván pro nastavení pracovního režimu snímače. Je určen jak pro čtení, tak i pro zápis. V AD7414 jsou dva bity s nejnižší váhou nastaveny z výroby (D0 a D1) a zbylá šestice bitů nastavuje pracovní režim snímače. V AD7415 jsou jen 3 bity použity k nastavení pracovního režimu. Ostatní jsou nastaveny z výroby [9].

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	0	0	0	0	0

Tab. 9.5: Struktura konfiguračního registru

registr	funkce
D7	plné snížení výkonu, když =1
D6	přemostění SDA a SCL filtrace, když = 0
D5	vyřazení funkce ALERT, když =1
D4	nastavení funkce ALERT; 1 - nastavení maximální úrovně, 0 - nastavení minimální úrovně
D3	reset ALERT funkce, když = 1
D2	jednorázová teplotní konverze, když =1

Tab. 9.6: Možnost naplnění registru pro AD7414

registr	funkce
D7	plné snížení výkonu, když =1
D6	přemostění SDA a SCL filtrace, když = 0
D2	jednorázová teplotní konverze, když =1

Tab. 9.7: Možnost naplnění registru pro AD7415

Jestliže je bit D7 nastaven jako ‘1‘, je zahájena teplotní konverze. Ta může být povolena také bitem D2. Tím je však provedeno jen jednorázové čtení teploty (tzn. “zapne“ snímač, přečte teplotu a opět zařízení “vypne“).

9.6 Teplotní hodnotový registr

Jedná se o 10-ti bitový registr, který obsahuje hodnotu teploty čtenou z A/D převodníku ve formátu dvojkového doplňku. Registr je používán pouze ke čtení, zápis do tohoto registru není možný. Tab. 9.8 ukazuje obsah prvního bytu dat určeného ke čtení na I_2C sběrnici (platí pro oba typy snímačů) [9].

D15	D14	D13	D12	D11	D10	D9	D8
první část hodnoty teploty od MSB k LSB							

Tab. 9.8: První byte teploty

D7	druhá část hodnoty teploty
D6	druhá část hodnoty teploty
D5	příznak přetečení
D4	přetečení maximální hodnoty
D3	přetečení minimální hodnoty
D2	0
D1	0
D0	0

Tab. 9.9: Druhý byte teploty pro AD7414

D7	druhá část hodnoty teploty
D6	druhá část hodnoty teploty
D5	nedefinovaný bit
D4	nedefinovaný bit
D3	nedefinovaný bit
D2	nedefinovaný bit
D1	nedefinovaný bit
D0	nedefinovaný bit

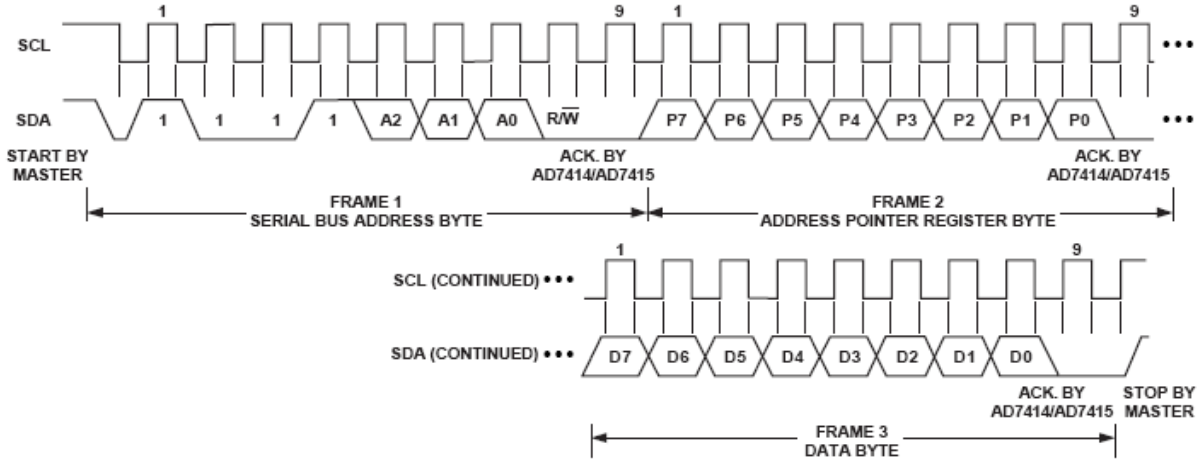
Tab. 9.10: Druhý byte teploty pro AD7415

Příznak přetečení maximální hodnoty je “1”, pokud je změřená teplota vyšší, než maximální hodnota nastavená v T_{HIGH} registru. Hodnota bitu je resetována při čtení dalšího bajtu. Pokud je teplota opět větší než požadovaná mez, příznak se zopakuje. To samé platí i pro přetečení minimální hodnoty. S tímto příznakem je spjat registr T_{LOW} [9].

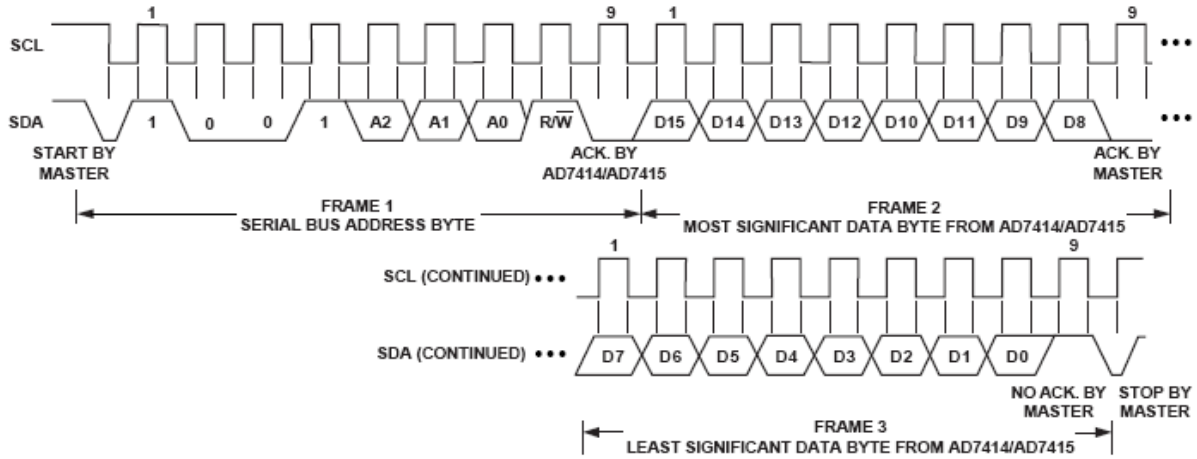
9.7 T_{HIGH} a T_{LOW} registr

Tyto registry jsou určené jak pro čtení, tak i pro zápis. Velikosti registrů je 8 bitů a uchovávají horní reps. dolní meze teplot (od MSB k LSB), jejichž překročením se aktivuje ALERT výstup. Jelikož je jejich velikost jen 8 bitů, rozlišovací schopnost činí 1 ℃.

9.8 Konkrétní použití zápisu a čtení registrů



Obr. 9.3: Příklad zápisu do příslušného registru ve vybraném zařízení



Obr. 9.4: Příklad čtení dvou bajtů dat z teplotního hodnotového registru

Snímače jsou připojeny na I_2C sběrnici jako Slave zařízení.

9.9 Adresace zařízení

AD7414 a AD7415 jsou dostupné ve čtyřech verzích, s označení -0,-1,-2 a -3. Každý snímač má 7-mi bitovou sériovou adresu. První 4 MSB bity jsou pevně nastaveny jako 1001. Zbylé 3 bity jsou vázány připojením AS pinu snímače na:

1) GND (zem)
2) napětí V_{DD}
3) plovoucí pin.

Touto možností adresace může být ke sběrnici připojeno až osm AD7414 snímačů nebo šest AD7415 zařízení.

Typ senzoru	Zapojení AS pinu	I2C adresa
AD7414-0	plovoucí	1001 000
AD7414-0	na GND	1001 001
AD7414-0	na V_{DD}	1001 010
AD7414-1	plovoucí	1001 100
AD7414-1	na GND	1001 101
AD7414-1	na V_{DD}	1001 110
AD7414-2	nedefinováno	1001 011
AD7414-3	nedefinováno	1001 111
AD7415-0	plovoucí	1001 000
AD7415-0	na GND	1001 001
AD7415-0	na V_{DD}	1001 010
AD7415-1	plovoucí	1001 100
AD7415-1	na GND	1001 101
AD7415-1	na V_{DD}	1001 110

Tab. 9.11: Adresace senzorů

10 UART

Univerzální asynchronní přijímač/ vysílač je vlastně hardware nebo software zařízení, které překládá data mezi paralelními a sériovými formami. UART je reprezentován obvykle jednotlivě nebo jako součást integrovaného obvodu užívaného pro sériovou komunikaci mezi počítači a jednotlivými periférierni a přístrojovými zařízeními. UARTy jsou také zahrnuty v mikrořadičích. Moderní mikropočítače umějí komunikovat pomocí UART také synchronně. Takové zařízení se pak nazývá USARTS [20].

10.1 Definice přenosu dat

Bity mohou být přesunuty z jednoho místa k jinému použitím vedení nebo i pomocí nějakého jiného prostředku. Se vzrůstající vzdáleností přenosu se zvyšují náklady na realizaci vedení. Pro snížení těchto nákladů na vedení se začal používat UART, jako přeměna paralelní formy přenosu na sériovou. Tím se značně redukuje počet linek na vedení. Bity jsou pomocí UART přenášeny sekvenčně. Každý UART obsahuje posuvný registr, který tvoří základní článek konverze mezi sériovými a paralelními formami [20].

UART obvykle vysílá nebo přijímá externí signály používané mezi dvěma zařízeními. Typicky je pro tuto činnost (přeměnu logických úrovní signálů na UART a zpět do externích úrovní signálů) používáno separátní zařízení. Externí signály mohou mýt různé formy, příklad standardů s různými úrovněmi napětí jsou RS-232, RS-422 a RS-485 z EIA.

Přítomnost napětí na vedení využívají i jiné typy přenosu, jako jsou optické vlákno, infračervený a bezdrátový systém. Některé typy přenosů využívají také modulace na nosný signál.

Komunikace může být takzvaně plně duplexní (obě zařízení zároveň posílají i přijímají) nebo poloduplexní (zařízení se střídají v odesílání a přijímání dat).

Od roku 2006 je v souvislosti UART běžně používaný systém RS-232. Svými vlastnostmi je ideální pro zprostředkování komunikace mezi zařízeními jako jsou počítače, periférie atd.

10.2 Asynchronní příjem a vysílání

UART pošle START bit, pět až osm datových bitů od nejméně významného bitu počínaje, volitelný paritní bit a nakonec STOP bit trvající jeden, jeden a půl nebo dvě periody hodinového cyklu. START bit je bit opačné polarity ke klidovému stavu. STOP bit je bit opačného charakteru proti START bitu a poskytuje zpoždění před dalším START bitem (toto je nazýváno asynchronním START – STOP přenosem). Dlouhotrvající STOP bit je používaný pro nastavení dostatku času k ukončení přenosu a také pomáhá při obnovení synchronizace.

Paritní bit dopomáhá k vytvoření požadovaného počtu bitů (sudých nebo lichých) mezi dvojicí bitů START a STOP. Může být lichý, sudý, nebo může být opomenutý. Lichá parita je více spolehlivá, protože zaručuje, že v bitech označujících data bude alespoň jeden přechod vysoké úrovně na nízkou, nebo obráceně.

V synchronním přenosu je hodinový signál oddělený od toku dat a nejsou v něm užívány START a STOP bity. To zlepšuje efektivitu přenosu tím více, čím více je

v odesílaném řetězci bitů méně ohraničovacích znaků. Asynchronní přenos nic nepřenáší, když není co posílat. Naproti tomu synchronní systém musí poslat blok znaků k tomu, aby udržel synchronizaci mezi přijímačem a vysílačem. Obvykle je to ASCII “SYN“ znak. To může být provedeno automaticky vysílacím zařízením. USART čipy obsahují jak synchronní, tak i asynchronní režim [20].

10.3 Sériově paralelní algoritmus

Puls dat se může nacházet v jednom ze dvou stavu. Pro tyto stavy existuje mnoho pojmenování. Když se puls nachází ve stavu “on“ (uzavřeném, nízkého napětí, toku proudu, logické nuly), říká se, že je v „prostorové“ podmínce. Když je puls v “off“ (otevřeném obvodu, vysokém napětí, zastaveném proudu, logické jedničce), říká se, že je v „mezní“ podmínce. Platí, že logická jednička je zaznamenaná jako logická nula.

Počáteční bit je vždy 0 (nízká logika) a signalizuje ho přijímací DTE (data terminal equipment) – koncové zařízení. Další 5 – 8 bitů představují data. V ASCII kódu může být osmý bit nastavený jako paritní. Další jeden nebo dva bity jsou v logické jedničce. Představují STOP bit a poskytují přípravu DTE k tomu, aby byl schopen přijímat další bity.

Všechny operace UART hardwaru jsou řízeny hodinovým signálem. Každý datový bit má délku právě jako jeden hodinový pulz. Přijímač testuje stav vstupního signálu při každém hodinovém pulzu (hledá začátek START bitu). Jestliže zjevný počáteční bit trvá alespoň jednu polovinu času pulzu, zařízení ho označí jako platný a signalizuje začátek přenosu dat. Jestliže je doba trvání kratší, impulz je ignorován. Na začátku přenosu vlastních dat je stav linky znovu vzorkován a jeho úroveň přesunuta do posuvného registru. Po požadovaném množství datových bitů (5 – 8) je obsah posuvného registru zpřístupněn v paralelním tvaru do přijímacího systému. UART nastaví příznak, že data jsou k dispozici a může generovat přerušení procesoru k tomu, aby procesor převedl přijatá data. V některých UART zařízeních je mezi posuvným registrem přijímače a hlavním systémovým rozhraním vložena vyrovnávací paměť (FIFO). Ta poskytne hlavnímu procesoru více času na ovládání přerušení z UART a zabraňuje ztrátám přenosu dat v rychlých přenosech.

Vysílání je jednodušší, protože probíhá pod kontrolou vysílacího systému. Jakmile jsou data uložená v posuvném registru UART, generuje START bit, příslušné množství datových bitů, paritní bit (pokud je využíván) a přidává STOP bity. Po dobu přenosu UART generuje příznak „zaneprázdněného“ stavu, aby hlavní systém neukládal nový START bit, pokud předchozí přenos nebyl dokončen.

Pro „plně“ duplexní přenos používá UART dva posuvné registry.

Vysílací i přijímací UARTy musí být pro správnou činnost nastaveny na stejnou bitovou rychlost, stejný počet datových bitů, paritu a ukončovací bity. Toto kontroluje přijímací UART. Ten pak při nevhodném nastavení aktivuje příznak pro hlavní systém.

Typická sériová rozhraní počítače používají osm datový bitů, potlačené parity a jeden koncový bit [20].

10.4 Struktura

UART obvykle obsahuje následující součásti:

- generátor hodin, obvykle vícenásobné přenosové rychlosti k tomu, aby dovolila vzorkování uprostřed periody bitu
- vstupní a výstupní posuvné registry
- kontrolu vysílání a příjmu
- řídící logiku pro čtení a zápis
- vysílací a přijímací vyrovnávací paměti (nepovinné)
- paralelní vyrovnávací paměť datových sběrnic (nepovinné)
- FIFO vyrovnávací paměť.

10.5 Chyby příjmu

10.5.1 Chyby přetečení

„Chyba přetečení“ nastává, když UART nezpracuje znak do doby, než dorazí další znak. Různá UART zařízení mají lišící se velikost vyrovnávací paměti za účelem přidržení přicházejících znaků. Procesor musí provádět odstranění bitů z vyrovnávací paměti. Jestli tomu tak není, vyrovnávací paměť se naplní a nastává chyba přetečení.

10.5.2 Chyby datového rámce

Nastává, když START a STOP bity jsou neplatné. START bit je důležitý pro identifikaci začátku přenosu dat. Vystupuje jako reference pro ostatní bity. Jestliže kanál není v klidovém stavu, když je očekáván STOP bit, vyskytne se chyba datového rámce.

10.5.3 Chyba parity

Nastane, když počet „aktivních“ bitů nesouhlasí se specifikovanou konfigurací parity. Jelikož je paritní bit volitelný, nebude se vyskytovat, jestli nebude parita využívaná.

11 RS-232

RS-232 (Recommended Standart 232) je standart pro asynchronní sériové datové komunikační signály. Slovo sériový znamená, že informace je posílána bit po bitu. Asynchronní říká, že data jsou posílána v předem deklarovaných časových úsecích, přesun dat může začít v jakémkoli čase, což má za úkol zajistit přijímač. Komunikace probíhá mezi koncovým zařízením DTE (Data Terminal Equipment) a centrálním zařízením DCE (Data Communications Eqipment) [21].

11.1 Rozsah standardu

Elektronická průmyslová aliance EIA (Electronic Industrie Aliance) definuje standardní RS-232 od roku 1969 a její znaky:

- napěťové úrovně, přenosovou rychlost, časové nastavení a frekvenci signálů, maximální dovolené napěťové úrovně, režim spojení nakrátko, maximální zatížení kapacitní reaktancí
- typy rozhraní, konektorů a přiřazení pinů
- funkci každého obvodu ve stykovém konektoru
- standardní podružné rozhraní obvodů pro vybrané aplikace.

Standard definuje také:

- znakové kódování (např.: ASCII, EBCDIC)
- rámec znaků dat (START, STOP a paritu)
- protokoly pro detekci chyb a kompresi dat
- přenosové rychlosti; standard říká, že RS-232 je určen pro přenosové rychlosti menší než 20000 bit/s. Moderní zařízení podporují rychlosti až 115200 bit/s a vyšší.
- napájení z externího zařízení.

Datový tok a přenosová rychlost je řízena sériovým rozhraním počítače - často integrovaným obvodem zvaným UART, který přeměňuje úrovně vnitřní logiky na kompatibilní úrovně pro RS-232 [21].

11.2 Napěťové úrovně

RS-232 definuje napěťové úrovně, které odpovídají logické „1“ a „0“, jako úrovně napětí $\pm(3-15)$V. Signál blízký nulovému napětí není platnou RS-232 úrovní. Logická jednička je definována jako záporné napětí, logická nula jako kladné napětí. Standart definuje maximální napětí na prázdno až 25 V a signálové úrovně $\pm$ 5V, $\pm$ 10V, $\pm$ 12V a $\pm$ 15V. Zařízení musí odolat krátkodobému nejasnému signálu.

Dále je řízena krokovací frekvence a doba náběžných a sestupných hran mezi úrovněmi. Protože napěťové úrovně jsou typicky vyšší než logické úrovně užívané integrovanými obvody, je požadováno vzájemné přeložení logických úrovní pomocí vnitřního procesu. Toto funguje i jako ochrana proti zkratům nebo přechodům, které mohou nastat na rozhraní. Dále poskytuje dostatečný proud k tomu, aby byla splněna požadovaná krokovací frekvence pro přenos dat.

Je nutné, aby mezi zemnícími piny konektorů RS-232 obvodu bylo nulové napětí, jinak může se zde může uzavřít zemní smyčka [21].

11.3 Konektory

RS-232 zařízení může být klasifikováno jako DTE nebo DCE. Standard doporučuje tzv. D-subminiaturní 25 pinový konektor.

Dále je standardem specifikováno 20 různých signálových spojení. Většina zařízení ale používá jen několik málo signálů, proto se používají i menší konektory. Například firmou IBM je používán 9-ti pinový konektor DE – 9, který je standardizován jako TIA – 574. V dnešní době je nejvíce využívaný 8-mi pinový konektor RJ – 45 (EIA/ TIA 561) nebo 10-ti pinový konektor RJ – 50. Dalšími běžnými konektory jsou DH10 a DE – 9, které jsou využívány na základních deskách.

Následující tabulka uvádí běžně užívané signály RS-232 a obsazení pinů na konektoru:

			Číslo pinů na konektorech							
Typ signálu	**Ozn.**	**Vstupní/ výstupní**	**DB-25**	**DE-9 (TIA-574)**	**EIA/ TIA 561**	**Yost**	**RJ-50**	**MMJ**	**Cisco RJ-45**	**Hirschmann RJ-45**
Uzemění	G	-	7	5	4	4, 5	6	3, 4	4, 5	4
Vysílaná data	TxD	výstupní	2	3	6	3	8	2	3	3
Přijímaná data	RxD	vstupní	3	2	5	6	9	5	6	5
Připravenost zařízení	DTR	výstupní	20	4	3	2	7	1	2	-
Připravenost dat	DSR	vstupní	6	6	1	7	5	6	7	-
Požadavek na vysílání	RTS	výstupní	4	7	8	1	4	-	1	-
Připravenost na vysílání	CTS	vstupní	5	8	7	8	3	-	8	-
Detekce spojení	DCD	vstupní	8	1	2	7	10	-	-	-
Skupinový indikátor	RI	vstupní	22	9	9	-	2	-	-	-

Tab. 11.1: Obsazení jednotlivých pinů na různých konektorech

Spojení pinu 1 (ochranná zem) a pinu 7 (signální referenční zem) je v praxi běžné, avšak není doporučené. Použití společné země je nevýhodou RS-232 [21].

11.4 Kabely

Používá se tzv. přímý kabel. Jde vlastně o spojení identických pinů v každém konektoru. Používají se také přechodky „samec/ samice“, které řeší změny mezi konektory. Pro funkční komunikaci přes rozhraní sériového portu je nutné shodné nastavení vysílacího a přijímacího zařízením (tzv. UARTu) příslušného přístroje.

Maximální délka kabeláže mezi zařízeními je nejvíce diskutovanou vlastností ve světě RS–232. Standart definuje maximální délku 15 metrů (50 stop), nebo délku

rovnou kapacitní reaktanci 2500 pF. To znamená, že použití kabelu s nízkou reaktancí dovoluje dosahovat delších vzdáleností [21].

Se vzdáleností pak souvisí přenosová rychlost, tzv. Baud rate. Dle firmy Texas Instrument následovně, kde 1 baud = 1 b/s.

Baud rate [Bd]	Maximální délka kabelu [m]
19200	15
9600	150
4800	300
2400	900

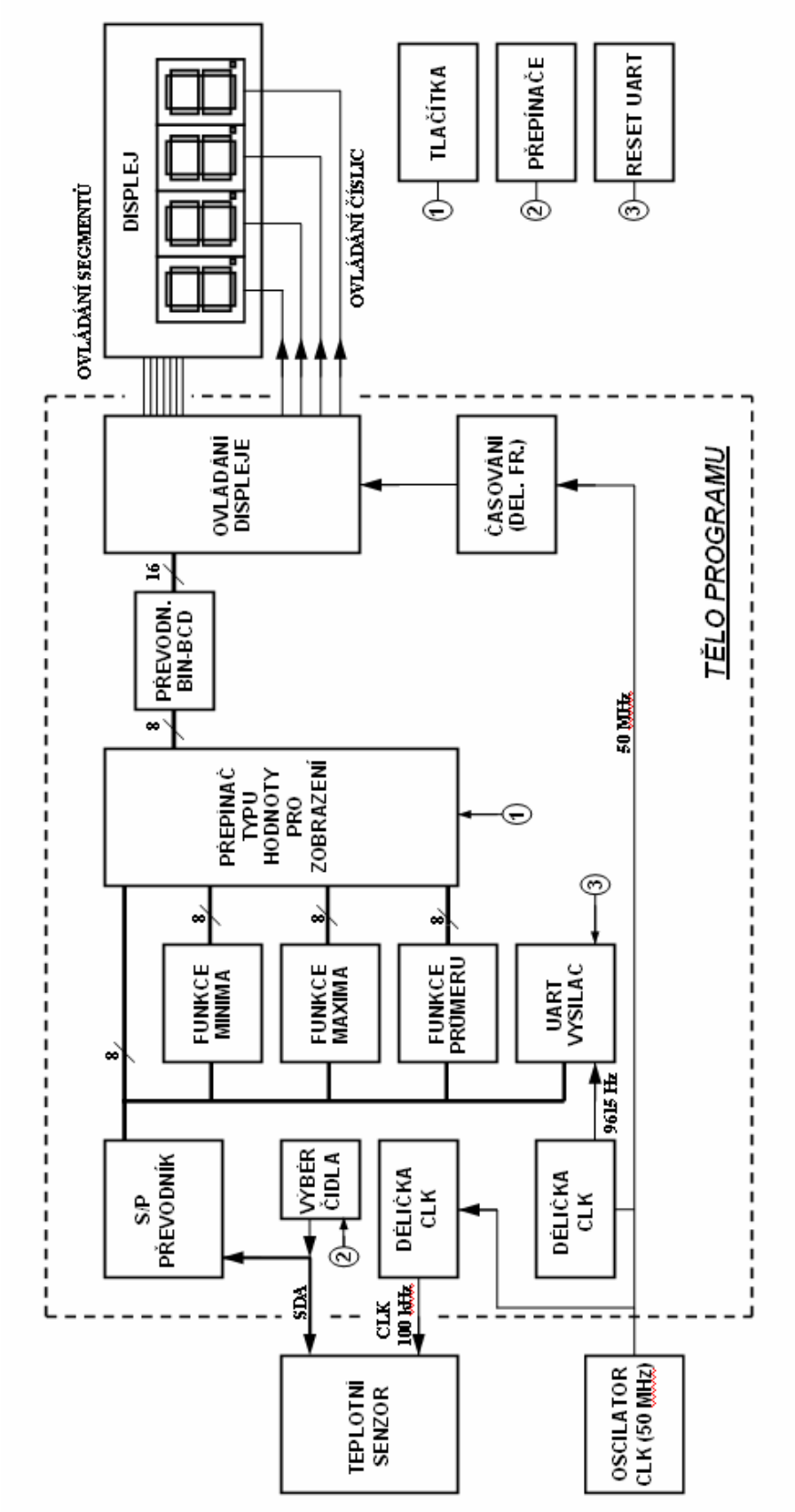
Tab. 11.2: Závislost délky kabelu na přenosové rychlosti

11.5 Související standardy

Další sériová rozhraní jsou:

- RS - 422: vysokorychlostní systém podobný RS – 232, ale s odlišným signalizováním
- RS - 423: vysokorychlostní systém podobný RS – 422, ale s nesouměrným signalizováním
- RS - 449: mechanické rozhraní využívající signály z RS – 422 a RS – 423
- RS – 485: odnož RS – 422, která může být využívána jako sběrnice s vícebodovým propojením
- MIL STD – 188: podobný RS – 232 s lepší impedancí a programovým řízením
- EIA 530: vysokorychlostní systém využívající elektrických vlastností RS – 422 nebo RS – 423 v konfiguraci EIA – 232
- TIA 574: standardizuje 9-ti pinový D - subminiaturní konektor pro použití s EIA - 232

12 Schéma uspořádání bloků programu



Obr. 12.1: Blokové schéma programu

13 Vlastní programy

Pro vlastní vytvoření programu pro logický obvod bylo nutné zamyšlení nad požadovanými vlastnostmi programu. Z hlediska složitosti návrhu byla práce rozčleněna na několik relativně malých bloků. Tyto bloky mají výhodu snadnějšího programování. Každý blok zpracovává vstupní data a představuje určitou jednoduchou funkci programu. Všechny dílčí bloky programu jsou pak sestaveny do výsledného konečného zápisu zdrojového kódu.

13.1 Sériově paralelní převodník

Jelikož přenos informace z teplotního čidla bude zprostředkován pomocí sériové sběrnice SMBUS ,a displej pracuje s paralelní interpretací informace, je nutné sériová data převézt na paralelní. K tomu slouží převodník, který požadovanou činnost vykoná. Převodník tvoří první blok programu.

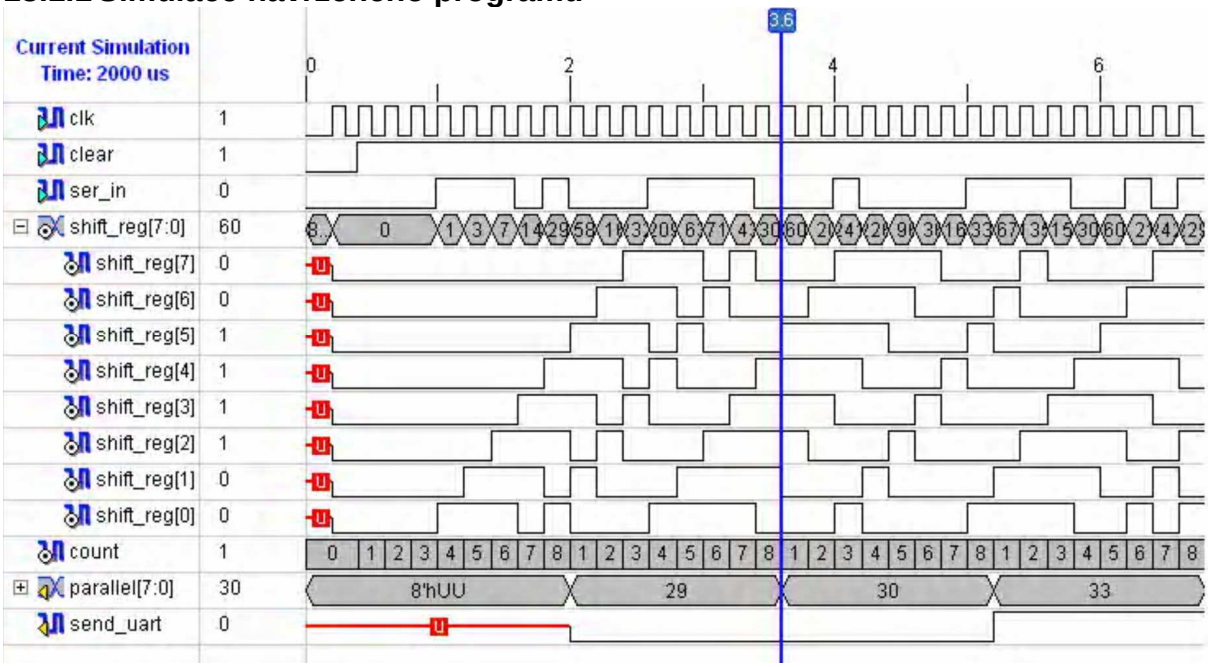
Zdrojový kód programu je viz. příloha A.

Vstupní port programu je nazván *ser_in*. Tímto portem přicházejí data z teplotního čidla, která dále zpracováváme. Vstup *ser_in* je definovaný jako 1-bitový (tzv. sériová linka) signál. Další vstupem je port *clear*, jehož definice je stejná jako v předcházejícím případě. Signálem *clear* je řízeno zahájení měření teploty. Jeho port může být připojen na jakékoliv čidlo, tlačítko, ovládací procesor atd., který při potřebě měření nastaví signál na tomto portu do vysoké úrovně. V práci je tento vstupní port připojen na napájecí napětí, které zaručuje nepřetržité měření teploty. Výstup z programu je realizován jako port *parallel*. Jeho velikost je 8 bitů. Druhým výstupním signálem je *send_uart*, který uvádí do aktivního stavu část programu sloužící k vysílání dat do počítače (uart vysílač). Informace z tohoto portu jsou vstupními hodnotami pro program převodníku binárního kódu na kód BCD. Tato část programu je ovládána hodinovým signálem, který je nazván *clk*.

Hlavní tělo obsahuje ještě signály *shift_reg*, který je používán jako 8-bitový registr pro účely posuvu příchozího signálu, a dále signál *count* celočíselného typu *integer*, který slouží pro pomocné čítání hodinových pulsů.

Program na začátku hledá náběžnou hranu pulsu hodinového signálu. Pokud je při náběžné hraně *clk* signál *clear* ve vysoké úrovni, přepíše data ze vstupu *ser_in* do pomocného registru *shift_reg(0)*. Dále do registrů *shift_reg (1 – 7)* se vloží signál ze vstupu *ser_in* tak, že průběh je do jednotlivých registrů zapsán vždy o dobu trvání jedné periody hodinového signálu později, než v předchozím registru. Současně se do proměnné *count* přičítá počet pulsů hodinového signálu. Po dosažení osmi pulsů se hodnota registrů přepíše na výstup *parallel*. Pokud je signál *clear* v nízké úrovni, bude ve všech registrech nulová hodnota (nízká úroveň). Nulová bude pak i hodnota v proměnné *count* [1], [2], [4].

13.1.1 Simulace navrženého programu



Obr.: 13.1: Simulace sériově – paralelního převodníku

Z obrázku simulace je zřejmé, že vytvořený program pracuje správně. Jednotlivé hodnoty registrů vypisuje na výstup právě po osmi cyklech hodinového signálu. Hodnota na výstupu pak odpovídá paralelní interpretaci signálu přivedeného na vstup v odpovídajícím čase.

13.2 Převodník binárního kódu na kód BCD

Druhý blok programu tvoří převodník binárního čísla na číslo v BCD tvaru.

Binární číslo je číslo zapsané ve dvojkové soustavě. Pro zápis se požívají pouze dva symboly - nejčastěji 0 a 1. Symboly 0 a 1 se používají právě z toho důvodů, že sami odpovídají stavům elektrického obvodu (vypnuto, zapnuto). Zápis binárního čísla tvoří posloupnost nul a jedniček tak, že jejich pozice v zápisu odpovídají mocninám čísla 2. Nejnižší váhu má symbol (bit) nacházející se nejvíce v pravé časti (na konci) čísla a to 2^0. Tento bit se nazývá bit LSB. Váha dalších bitů roste od LSB směrem doleva po jednotkách argumentu (2^0, 2^1, 2^2…) až po bit s nejvyšší váhou MSB, který se nachází zcela vlevo ve výčtu jedniček a nul.

$$93_{10} = 01011101_2 = 1{\cdot}2^6 + 0{\cdot}2^5 + 1{\cdot}2^4 + 1{\cdot}2^3 + 1{\cdot}2^2 + 0{\cdot}2^1 + 1{\cdot}2^0 \qquad (4)$$

BCD kód je odnoží binárního kódu. Ke své interpretaci užívá dvojkové soustavy (tzn. 0 a 1). Číslo z desítkové soustavy se zapíše v tomto kódu tak, že hodnoty zbytků po celočíselného dělení deseti se zapíší pomocí posloupnosti čtyř symbolů 0 a 1 [2]. Před zapsáním další části čísla do BCD soustavy se vždy stávající číslo vydělí deseti a znovu se vypočte zbytek po celočíselném dělení takto:

$$128_{10} = 0001\ 0010\ 1000_{BCD} \qquad (5)$$

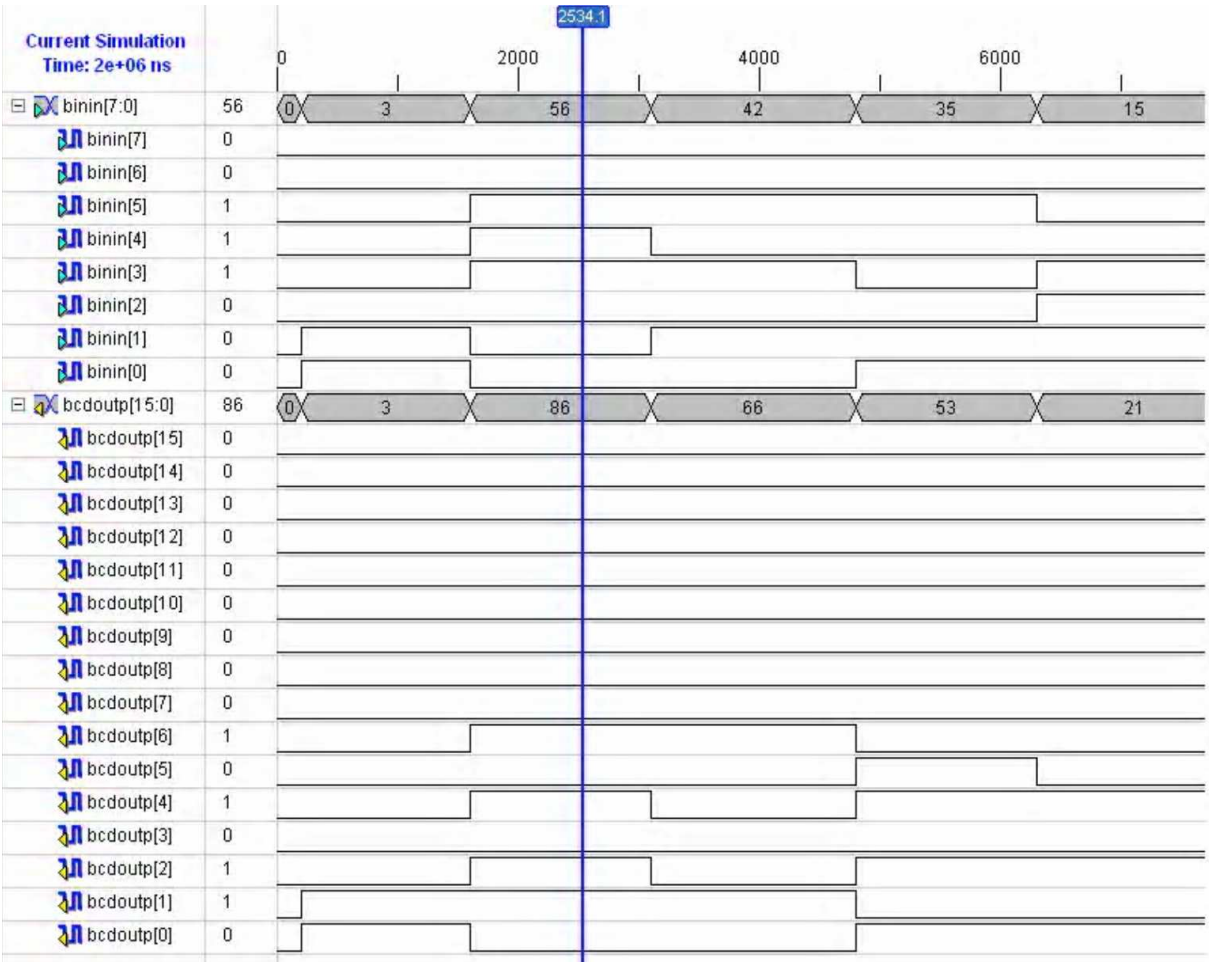
$$1_{10} = 0001_2$$

$$2_{10} = 0010_2$$

$$8_{10} = 1000_2$$

Vstup do tohoto bloku tvoří port *BinIn*, který má velikost 8 bitů. Výstupem je port *BCDoutp* o velikosti 16 bitů. Vlastní program převádí vstupní osmi-bitovou hodnotu binárního kódu na kód BCD metodou přičítání jednotlivých vstupních bitových příspěvků s korekcí desetinných čísel [1], [2], [4].

13.2.1 Simulace navrženého programu



Obr. 13.2: Simulace převodníků binárního kódu na kód BCD

Na simulaci je patrné, že vytvořený program pracuje správně. Vstupní binární hodnota v čase umístění kurzoru je $(0011\ 1000)_2$. Tato hodnota odpovídá hodnotě 56_{10}. Vyjádření v BCD kódu je $(0101\ 0110)_{BCD}$, což odpovídá teoretickým předpokladům.

13.3 Ovládání displeje

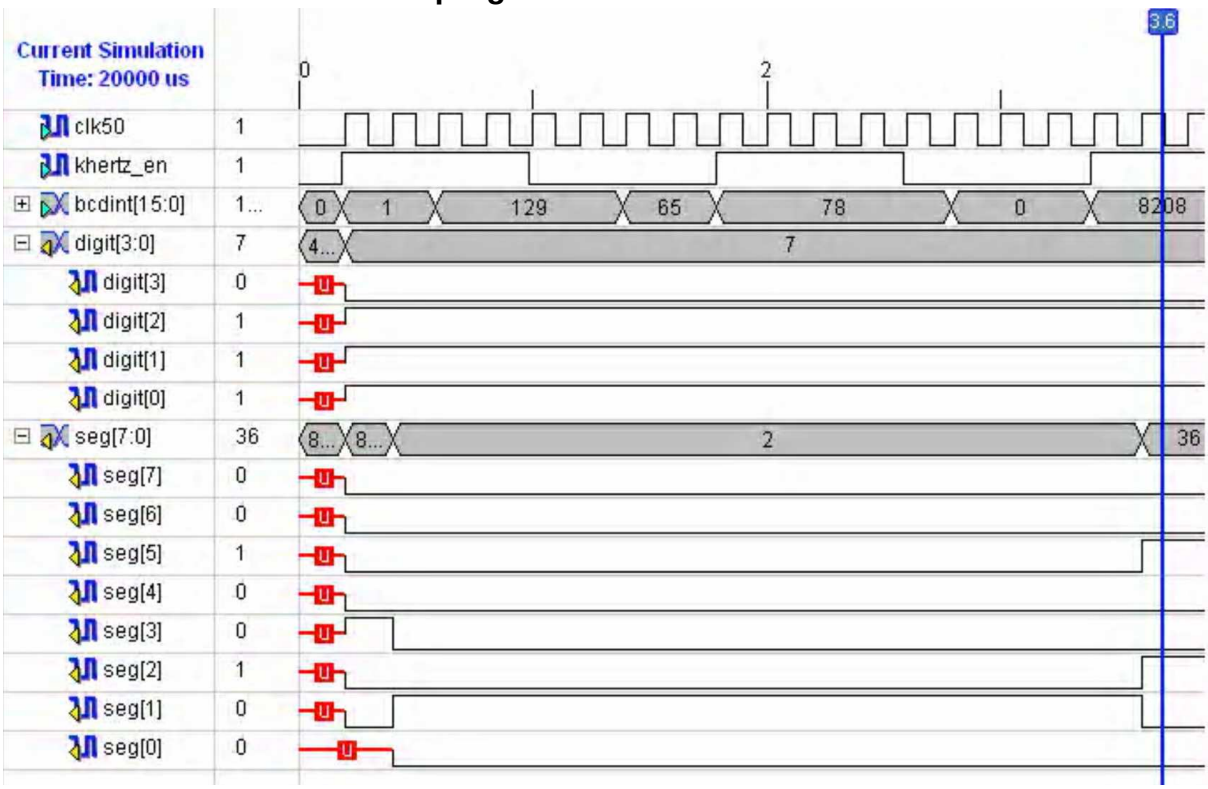
Další blok programu tvoří posloupnost příkazů, které zajišťují zobrazení hodnoty měřené teplotním čidlem na displeji. Displej je složen ze čtyř sedmisegmentových zobrazovacích LED. Hodnota však není zobrazena tak, že všechny číslice svítí současně, ale vždy svítí jen jedna číslice z celé hodnoty. Jednotlivé číslice jsou postupně rozsvěcovány s takovou frekvencí, že lidské oči nevnímají “problikávání” znaků, ale vidí svit celého čísla. Frekvenci zobrazování jednotlivých číslic tento program ale nezajišťuje.

Návrh jednotky displeje není součástí návrhu při bakalářské práci. Jednotka je tvořena jako již zkonstruovaná část vývojové desky.

Vstupních portů do tohoto programu je několik. Zejména jsou to *bcdint*, což je signál přivedený z výstupního portu převodníku binárního čísla na kód BCD. Velikost portu je 16 bitů. Dalším vstupem je jedno-bitový port *khertz_en*, který zajišťuje rychlost zobrazování jednotlivých číslic. Program zajišťuje také zobrazování desetinných teček, které pro zobrazení nejsou použity. Vstupy pro jejich indikaci jsou porty *dp3* – *dp0*, jejichž velikost je opět jeden bit. Program je ovládán hodinovým signálem *clk*. Výstup tvoří port *digit*, který vybírá jednotlivé sedmisegmentové LED displeje, na kterých budou číslice zobrazovány. Velikost je 4 bity. Druhým výstupem z programu je *seg*, který představuje rozsvěcování jednotlivých segmentů displeje. Rozsah je 8-bitový.

První dvě části vlastního zdrojového textu nejsou v našem případě využívány. První část zajišťuje zhasnutí desetinných teček a druhá resetování displeje. V další části nejprve program hledá současné náběžné hrany signálů *clk* a *khertz_en*, následně zajišťuje časování zobrazovaných číslic na jednotlivých pozicích displeje. Následuje rozdělení 16-ti bitového vstupního signálu na čtyři 4-bitové tak, že nejnižší číslice je tvořená bity 0 – 3 vstupního signálu atd. Tomuto rozdělení jsou následně přiřazeny pozice zobrazení na displeji. Poslední část navřeného programu má za úkol zobrazování jednotlivých číslic 0 až 9 pomocí rozsvěcování příslušných segmentů na displeji [1], [2], [4].

13.3.1 Simulace navrženého programu



Obr. 13.3: Simulace programu na ovládání displeje

Simulace ukazuje zobrazení číslice v nejvyšším řádu. Zobrazený znak je 2.

13.4 Dělička clk pro ovládání displeje

Program zajišťuje časování hodinového signálu clk. Zjednodušeně řečeno je to dělička frekvence signálu clk. Program je nutností pro to, aby číselná hodnota na

LED displeji byla vnímána jako současně rozsvěcené znaky, což se ve skutečnosti neděje. To bylo již popsáno v odstavci pro popisu programu pro ovládání displeje.

Vstupním portem pro tento program je signál *Clk*, výstup tvoří proměnná *KHzEn*, která má stejnou velikost jako *Clk*.

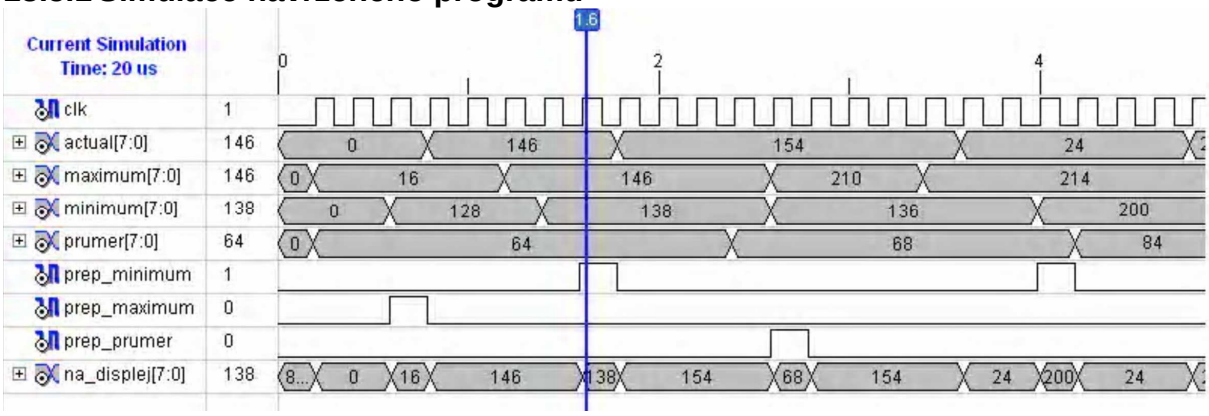
Tělo je tvořeno ze dvou částí. První část dělí signál o kmitočtu 50 MHz padesáti. Výsledkem je tedy signál o kmitočtu 1 MHz. Ten se nadále dělí tisícem. Výsledkem je tedy obdélníkový signál *KHzEn* o kmitočtu 1 kHz. Ten je pak použit jako vstupní port pro podprogram na ovládání displeje [1], [2], [4].

13.5 Přepínač právě indikovaných hodnot

Program má za úkol dle požadavků uživatele přepínat hodnoty teploty zobrazené na sedmisegmentovém LED displeji. Přepínání je prováděno mezi hodnotou minimální, maximální, průměrné a aktuální teploty. Přepínání je realizováno tak, že je ovládáno mikrotlačítky na vývojové desce logického obvodu. Základní zobrazovanou hodnotou je aktuální teplota. Je-li některé z mikrotlačítek přidrženo ve stisknutém stavu, na LED displeji je zobrazena příslušná hodnota. Pokud je tlačítko uvolněno, na displeji je zobrazována opět aktuální hodnota teploty.

Program je ovládán hodinovým signálem *clk*. Vstupními signály pro tento program jsou výstupní signály z programů pro indikaci maximální (*maximum*), minimální (*minimum*), průměrné (*prumer*) a aktuální (*actual*) hodnoty. Všechny vstupní porty mají stejnou velikost 8 bitů. Výstupem je hodnota, která je následně zobrazovaná na LED displeji, na portu *na_displej.* Tento port je stejně velký, jako porty vstupní. Při náběžné hraně *clk* signálu je aktuální hodnota teploty převedena na výstupní port *na_displej.* Dále je zkoumáno stisknutí mikrotlačítek. Při stisknutí některého z nich je na výstupní port převedena hodnota s ním spjatá, tzn. minimální, maximální nebo průměrná.

13.5.1 Simulace navrženého programu



Obr. 13.4: Simulace programu pro přepínání zobrazovaných hodnot

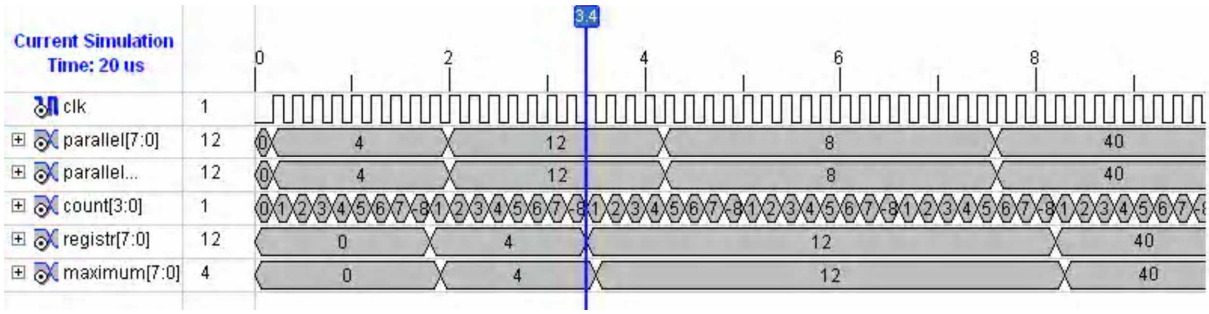
Simulace ukazuje, že program pracuje správně. Není-li stisknuté žádné z mikrotlačítek, je výstupu přiřazena aktuální hodnota. Na místě časové značky (maker) je nasimulováno stisknutí tlačítka pro indikaci minimální hodnoty. Právě v tomto čase je na výstupní port přiřazena odpovídající hodnota teploty (tedy minimální), což ukazuje správnou funkci programu.

13.6 Program pro indikaci maximální hodnoty

Program slouží k zobrazení maximální změřené teploty na sedmisegmentovém LED displeji.

Vstupní daty do programu, nazvané portem *paralel,* jsou výstupní signály ze sériově-paralelního převodníku. Tento port má rozsah 8 bitů. Program reaguje na náběžnou hranu hodinového impulsu, při níž se vstupní signál převede na celočíselný typ, se kterým se nechá již v programovém prostředí Xilinx ISE 9.2i matematicky pracovat. Jelikož se hodnota teploty mění po 8-mi bitech (teplota je A/D převodníkem interpretovaná pomocí osmi bitů), je v programu použito počítadlo, které je po dosáhnutí hodnoty 8 nulováno. Čítání pak začíná od 0 a dále inkrementuje svojí hodnotu. Při dosažení právě 8-mi hodinových pulsů se aktuální hodnota porovnává s hodnotou uloženou v pomocné proměnné *registr* (počáteční hodnota registru je nulová). Pokud je aktuální hodnota vyšší, než hodnota uložená v registru, je registr přepsán právě vyšší hodnotou. Nakonec je hodnota uložená v registru převedena na vektor hodnot a vložena do výstupní proměnné.

13.6.1 Simulace navřeného programu



Obr. 13.5: Simulace programu pro indikaci maximální hodnoty

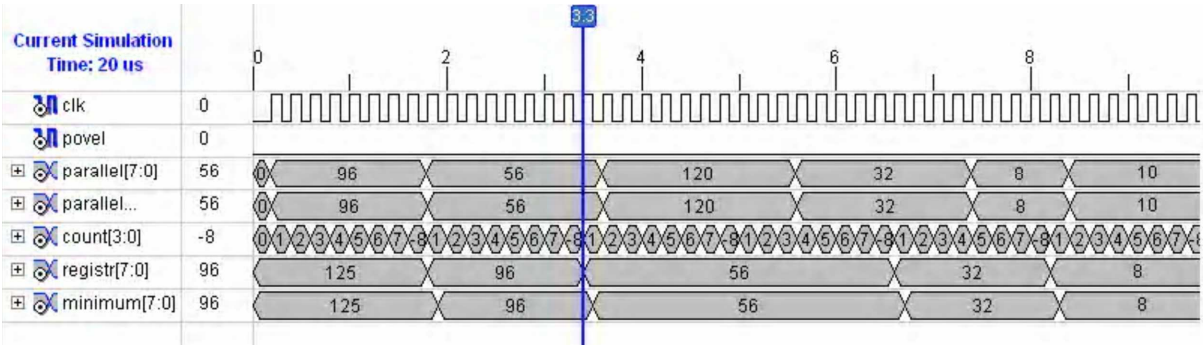
Ze simulace je patrné, že program porovnává aktuální hodnotu s hodnotou uloženou v registru každých 8 hodinových cyklů signálu *clk*. Registr vždy přepisuje jen vyšší hodnotu. Ta je následně i výstupem celého programu.

13.7 Program pro indikaci minimální hodnoty

Program slouží k indikaci minimální změřené hodnoty teploty na sedmisegmentovém displeji.

Vstupem do programu je signál výstupního portu z A/D převodníku. Velikost tohoto vstupního portu je 8 bitů,a je označen jako *paralel.* Program reaguje na náběžnou hranu hodinového signálu *clk*, při kterém je aktuální hodnota na vstupu převedena na celočíselný typ. Prostředí Xilinx ISE 9.2i totiž neumí pracovat s matematickými operacemi hodnot typu číselných vektorů. V programu je použito počítadlo hodinových pulsů *count*, které počítá od nuly do osmi a následně se vynuluje, proces se nepřetržitě opakuje. Aktuální hodnota se mění právě po 8-mi bitech. Při hodnotě *count = 8* je aktuální hodnota srovnávána s hodnotou uloženou v pomocném registru s názvem *registr*. Pokud je aktuální hodnota menší, přepíše se registr právě touto hodnotou. Nakonec se počítadlo vynuluje a hodnota registru se převede na vektor hodnot a na výstupní port programu *minimum*.

13.7.1 Simulace navrženého programu



Obr. 13.6: Simulace programu pro indikaci minimální hodnoty

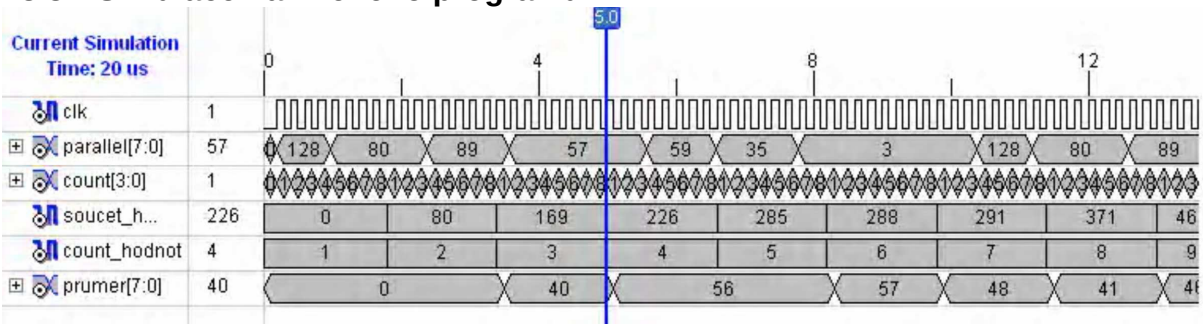
Ze simulace je patrné, že program porovnává aktuální hodnotu s hodnotou uloženou v registru každých 8 hodinových cyklů signálu *clk*. Registr vždy přepisuje jen nižší hodnotu. Ta je následně i výstupem celého programu.

13.8 Program pro indikaci průměrné hodnoty

Program slouží k zobrazení průměrné hodnoty teploty na sedmisegmentovém displeji.

Program využívá pomocného registru *soucet_hodnot*, do kterého se po uplynutí 8-mi pulsů hodinového signálu přičítá aktuální teplota. Tak je pak vydělena proměnnou *count_hodnot*, která zaznamenává počet hodnot přičtených do proměnné *soucet_hodnot.* Výsledkem je průměr teplot za určitý čas, který je přiřazen výstupnímu portu *průměr*. Pro výpočet průměru musí být proměnné převedeny na celočíselné typy.

13.8.1 Simulace navrženého programu



Obr. 13.7: Simulace programu pro indikaci minimální hodnoty

Simulace ukazuje správnou funkčnost programu. V místě kurzoru je součet hodnot roven 169, počet hodnot je 3. Výsledkem je tedy průměrná hodnota 56.

13.9 UART vysílač

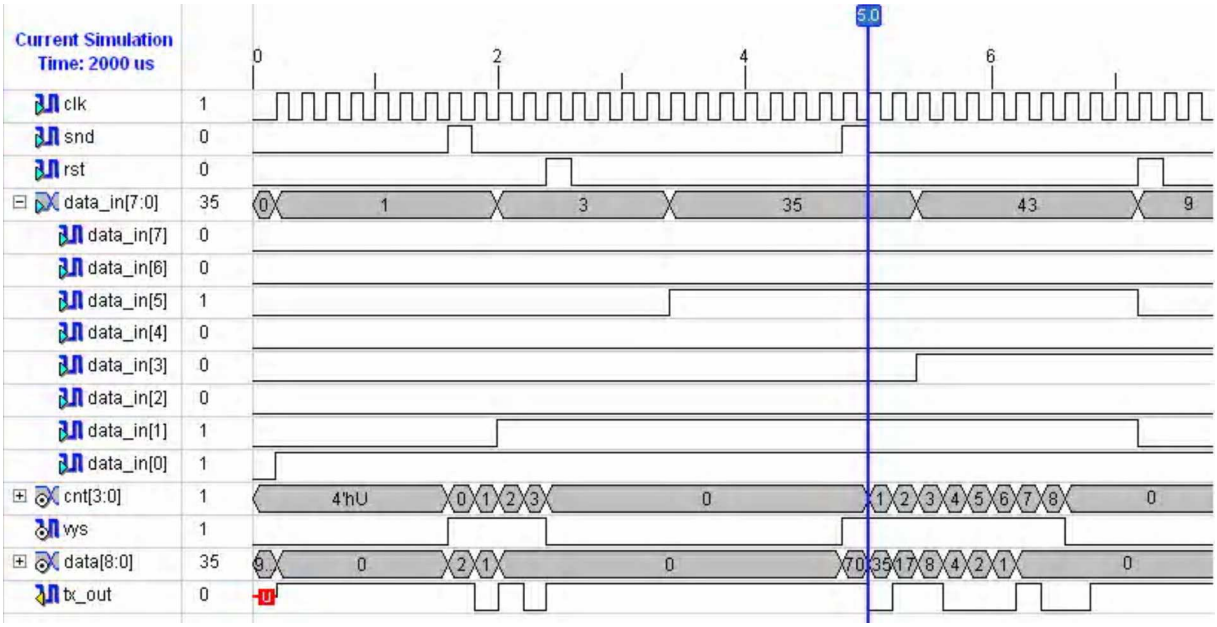
Program slouží pro vysílání dat pomocí rozhraní UART (protokolu RS-232) do počítače.

Program je řízen hodinovým signálem o kmitočtu 9600 Hz. To znamená, že signál generovaný krystalovým oscilátorem vývojové desky musí být vhodným způsobem podělen. Pro tento účel je použita dělička frekvence pro UART vysílač, která je popsána níže. Samotný program obsahuje několik vstupů. Pro zahájení

vysílání slouží port *snd*. Do tohoto portu jsou vysílány signály ze sériově - paralelního převodníku. Jedná se o jednobitový signál. Dalším vstupním jednobitovým portem je *rst*, který slouží pro resetování vysílače. Data potřebná k odeslání jsou přivedena 8-mi bitovým portem *data_in*. Výstup vysílače je jednobitový port s názvem *tx_out*. V programu je dále použito počítadlo hodinových pulsů *cnt*. To slouží pro zdárný převod paralelního signálu na sériový.

Pokud je stisknuté tlačítko potřebné pro reset vysílače, je počítadlo nulováno a na výstupním portu se objeví signál ve vysoké úrovni. Při příchodu impulsu pro povolení vysílaní dat (*snd = 1*) začne počítadlo *cnt* inkrementovat svoji hodnotu v závislosti na hodinovém signálu *clk*. Vstupní signál je pak pomocí posuvných registrů převeden na sériový, který je následně přiřazen na výstup. Jakmile *cnt* = 1000_2 je počítadlo vynulováno.

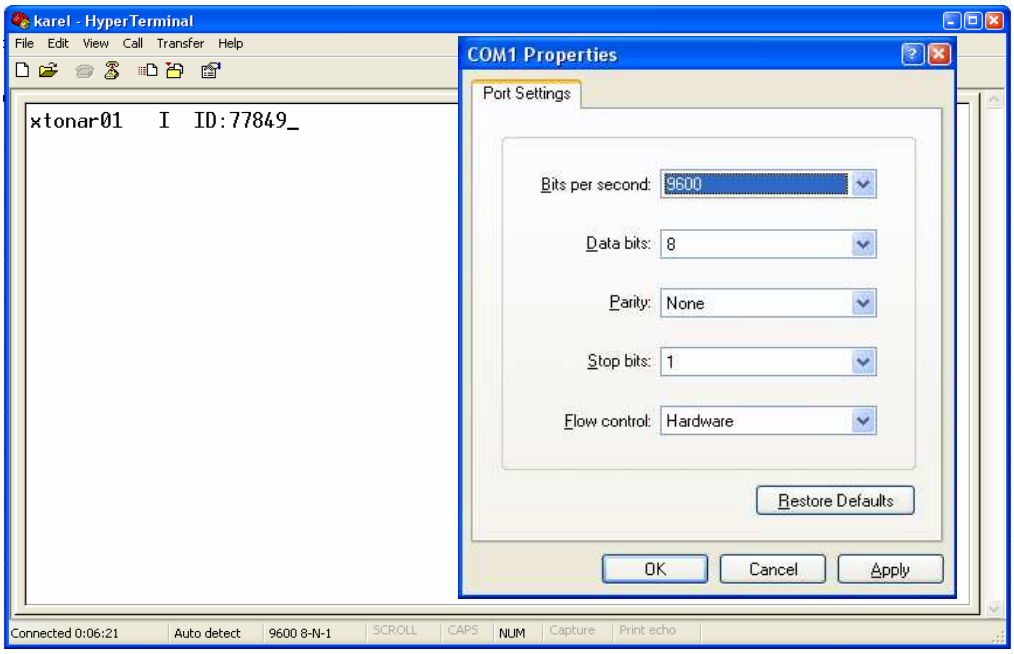
13.9.1 Simulace navrženého programu



Obr. 13.8: Simulace programu vysílače UART

Simulace ukazuje, že program vysílače pracuje správně. V čase asi 2 µs se nachází impuls signálu *snt.* Tzn., že na výstupu se začíná objevovat sériová interpretace vstupního signálu. Po uplynutí 3 hodinových pulsů se však objevuje impuls *rst*, který převod zastavuje. V místě časové značky (po impulsu signálu *snd*) převádí paralelní hodnotu (00100011_2) na sériovou tak, že posloupnost bitů je odesílána od bitu s nejnižší váhou (LSB) po bit s nejvyšší váhou (MSB).

Simulace je doplněna o ukázku funkčnosti UART zařízení, kde pomocí posuvných přepínačů a ASCII tabulky byl vytvořen text, viz. obr. 13.10.



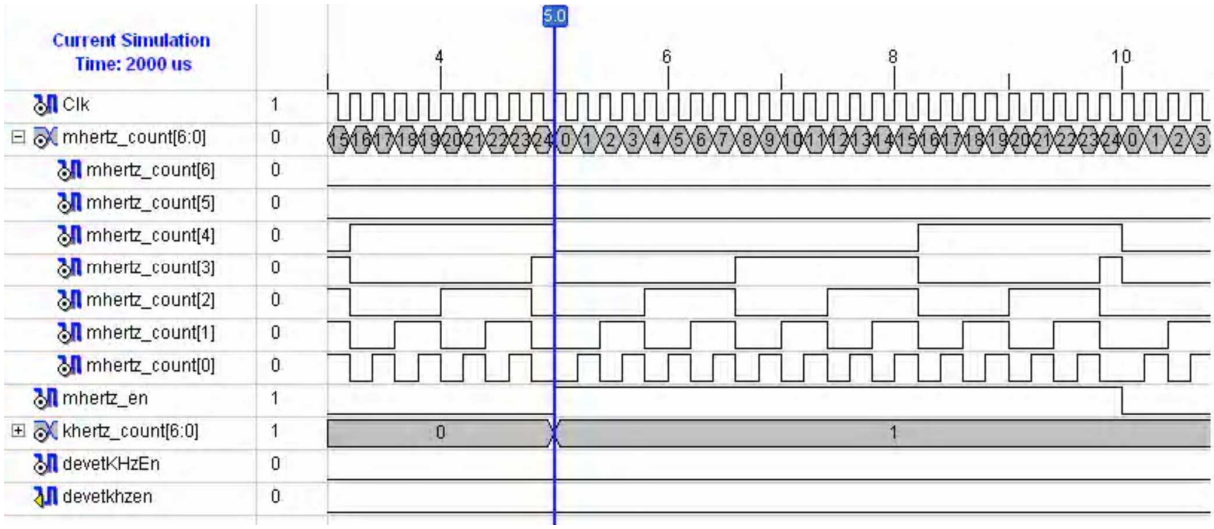
Obr. 13.9: Uživatelské prostředí HyperTerminal

13.10 Dělička *clk* signálu pro UART

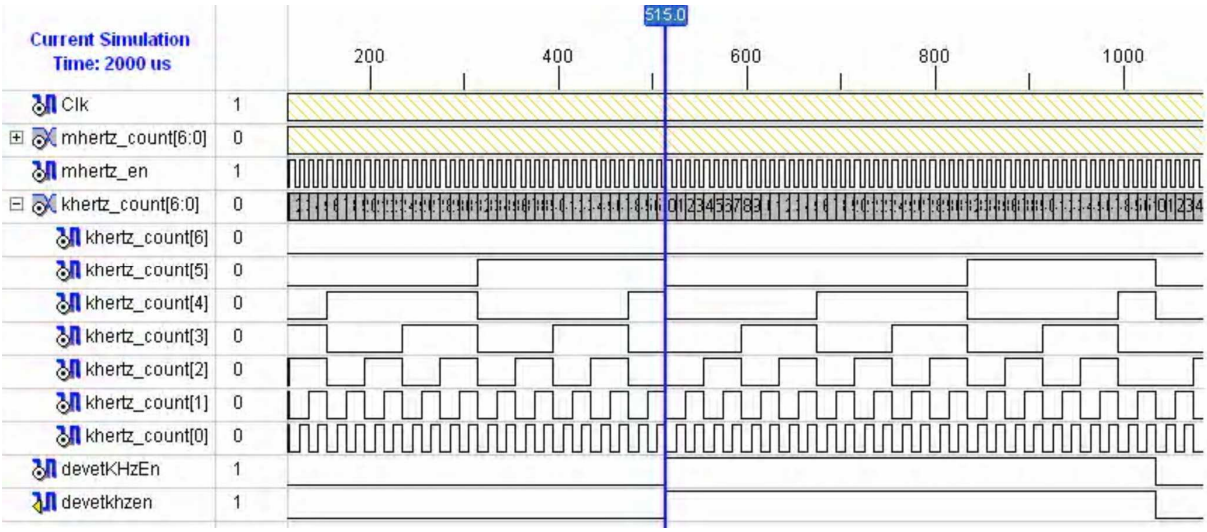
Jelikož UART vysílač pracuje s hodinovým signálem o kmitočtu 9600 Hz, je nutné hodinový signál z krystalového oscilátoru o kmitočtu 50 MHz vydělit.

Dělička je rozdělená na dvě části. Jelikož se signál musí invertovat v polovině periody, je v první části vstupní signál 50 MHz podělen číslem 50. Změna úrovně signálu nastává při 25. taktu *clk*. Po 50. taktu *clk* se úroveň změní zpět, do výchozího stavu. Tím je dosaženo signálu s kmitočtem 1 MHz. Ve druhé části dochází k dělení číslem 104. Tím je dosaženo signálu s kmitočtem, 9615,4 Hz, což se nejvíce přibližuje potřebnému signálu 9600 Hz.

13.10.1 Simulace navrženého programu



Obr. 13.10: Simulace děličky clk pro UART 50 MHz→1 MHz



*Obr. 13.11: Simulace děličky clk pro UART 1 MHz→*9615,4 Hz

Ze zobrazených simulací děličky clk je patrné, že vzniklý program pracuje správně.

13.11 Program pro výběr čidla

Jelikož je možné na sběrnici připojit více než jedno slave zařízení (v našem případě teplotních senzorů), bylo nutné vytvořit program, který bude určovat, ze kterého teplotního čidla bude hodnota zpracovávána a indikována na sedmisegmentovém LED displeji.

Program pro výběr teplotního čidla je ovládán hodinovým signálem *clk*. Výběr tlačítka je ovládán pomocí posuvných přepínačů, které jsou přítomny na vývojové desce SPARTAN-3. Na sběrnici může být připojeno celkem až osm zařízení AD7414 nebo šest zařízení AD7415. Výběr zařízení spočívá v tom, že master umístí na SDA linku data o délce 29 bitů, které se skládají z:

- 1 START bit nastaven na ‘0‘
- 4 bity určují zápis do adresního registru (1111) nebo čtení z teplotního hodnotového registru (1001)
- 3 bity – adresa zařízení viz tab. 9.11
- 1 bit $R/\overline{W}$, který určuje zápis, nebo čtení (‘1‘ – čtení, ‚0’ – zápis)
- 1 bit potvrzovací ACK nastaven na ‘0‘
- 8 bitů – adresa registru, ve kterém chceme změnit některý bit
- 1 bit potvrzovací ACK nastaven na ‘0‘
- 8 bitů, které přestavují jednotlivé bity v registru
- 1 bit potvrzovací ACK nastaven na ‘0‘
- 1 STOP bit nastaven na ‘1‘.

Výsledná posloupnost umístěných dat na sběrnici tedy je:

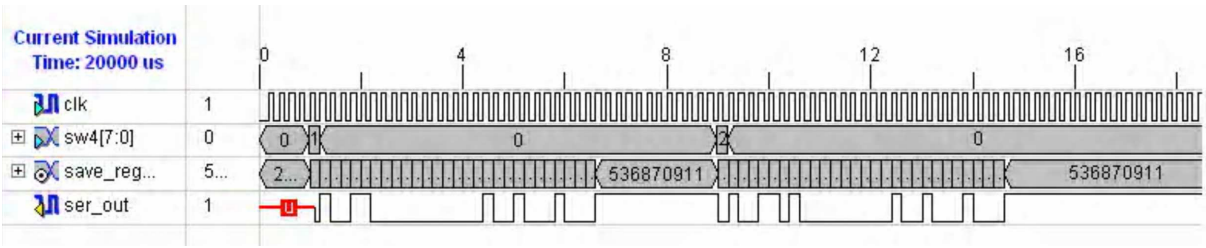
10001000100100000000**001**10010

, přičemž pro adresaci zařízení se mění jen tučně zvýrazněné bity podle tab. 9.11, jinak jsou všechny bity nastaveny stejně pro jakýkoliv teplotní senzor.

Tato posloupnost je v programu interpretována jako signál o velikosti 29 bitů. Aby bylo možné signál umístit na sběrnici je nutné ho z paralelního tvaru převézt na

tvar sériový. Tento úkon provádí zpětný posuvný registr. Nakonec je signál přiřazen jednobitovému výstupnímu portu *ser_out*.

13.11.1 Simulace navrženého programu



Obr. 13.12: Simulace programu pro výběr čidla

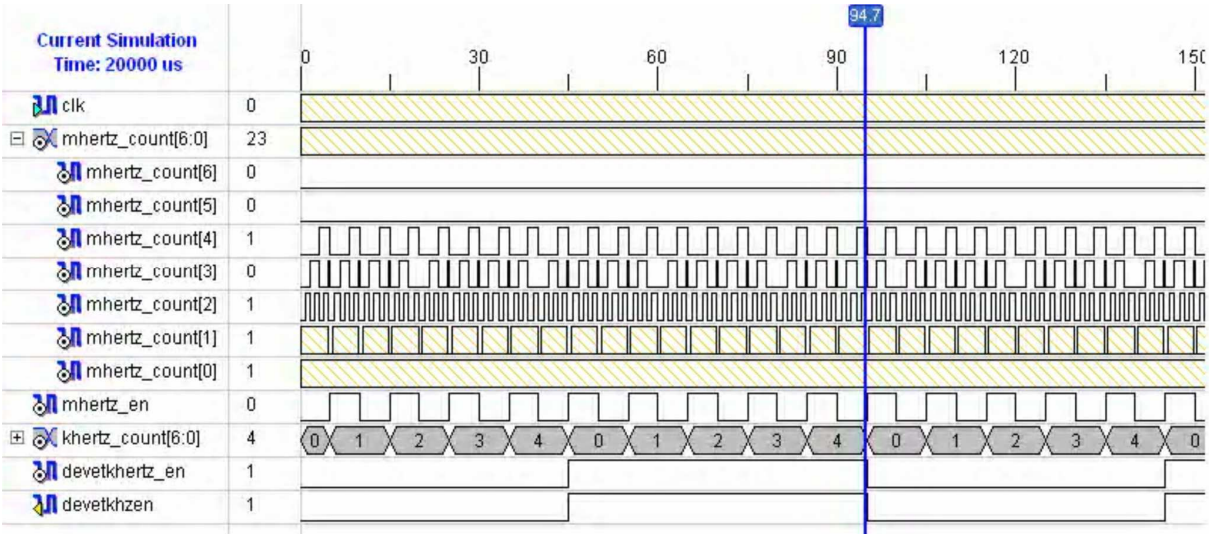
Simulace ukazuje, že program funguje správně. Po stisku (přepnutí) tlačítka sw4(0) je výstupu přiřazena posloupnost bitů pro adresaci modulu s teplotním snímačem AD7414-1. Pokud je však stisknuto tlačítko sw4(1), je adresováno zařízení s teplotním senzorem AD7415-1.

13.12 Dělička *clk* signálu pro modul teploty

Program má za úkol posílat hodinový signál do teplotního senzoru. Výstupní kmitočet z programu je 100 kHz, což vyhovuje požadavkům AD7414 (AD7415). Tato součástka umí pracovat do kmitočtu f = 400 kHz.

Program je navržen dle děličky pro UART vysílač. První část programu je prakticky shodná. Simulace první části je na obr. 13.11. Druhá část dělí signál o kmitočtu 1 MHz na signál s kmitočtem 100 kHz.

13.12.1 Simulace navrženého programu



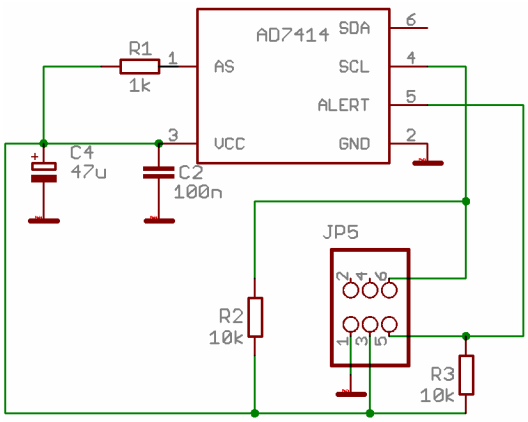
Obr. 13.13: Simulace děličky clk signálu pro modul s teplotní senzorem

Ze zobrazené simulace je patrné, že vytvořený program pracuje dle zvolených požadavků.

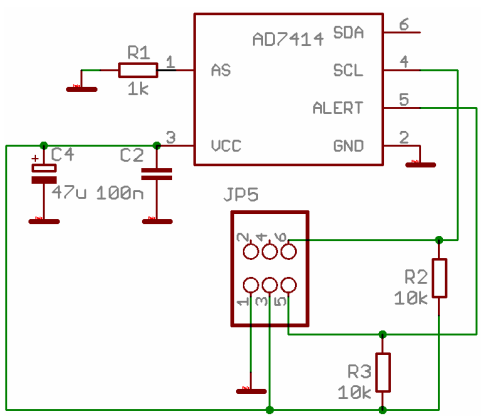
14 Návrh hardware

Hardwarová část bakalářské práce se zabývá návrhem modulu pro měření teploty. Srdcem modulu je teplotní senzor AD7414 (AD7415). Dále jsou k senzoru přiřazeny součástky, které slouží pro filtraci signálu, napájecího napětí a adresaci zařízení. Modul je napájen přímo z vývojové desky SPATAN-3, kde napětí na výstupní pinu 3,3 V splňuje požadavky pro napájení senzoru. K modulu je z vývojové desky také přiveden zemnící vodič GND. Samozřejmostí je připojení SDA a SCL linky taktéž z vývojové desky. Z toho vyplývá, že k modulu povedou 4 vodiče.

Elektronické zapojení součástek modulu bylo převzato z datasheetu teplotního senzoru [9]. Úpravy provedené na schématu jsou minimální. Výstupy z teplotního senzoru, které jsou dále přivedeny na sběrnici musí být opatřeny zdvihacími rezistory. Doporučená hodnota odporu je 10 kΩ. Jedná se o výstup hodinového signálu CLK, sériovou datovou linku SDA a výstup ALERT, který slouží pro využití senzoru jako součást termostatu. Přivedené napájecí napětí je opatřeno blokovacím a filtračním kondenzátorem. Tvoří je tantalový kondenzátor s kapacitou 47 μF a keramický kondenzátor o kapacitě 100 nF. Adresní vstup teplotního senzoru AS je podle typu adresy zařízení připojen přes rezistor s odporem 1 kΩ na napájecí napětí V_{DD} nebo na zem GND. Vstupní pin senzoru ALERT v případě použití čidla typu AD7414 je plovoucí.

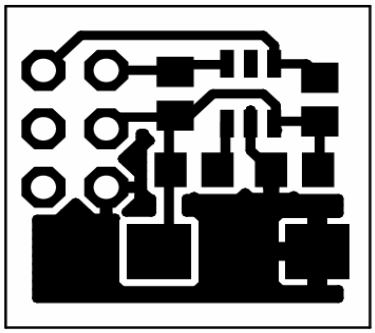


Obr. 14.1: Schéma zapojení modulu s AD7414

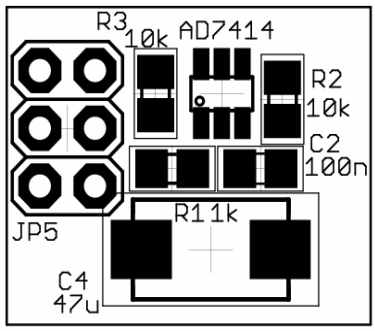


Obr. 14.2: Schéma zapojení modulu s AD7415

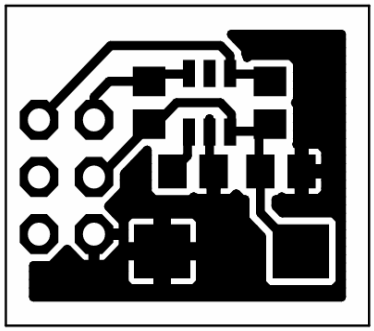
Modul je vytvořen napájením SMD součástek na desku plošných spojů (DPS). Snahou bylo využití co nejmenšího rozměru modulu. Pouzdra SMD součástek byla volena v typovém označení 0805. Návrh byl realizován v programu EAGLE Layout 4.16r2, který je ve verzi freeware dostupný na internetu nebo na www stránkách Ústavu radioelektroniky. DPS modulu může být dále umístěna například v šasi nebo libovolném pouzdře.



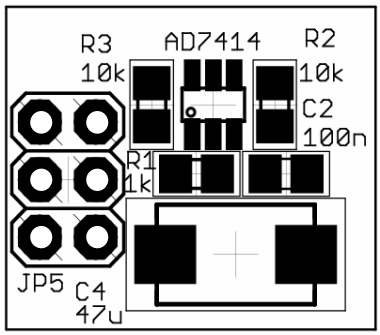
Obr. 14.3: Deska plošných spojů modulu s AD7414 (pohled ze strany součástek)



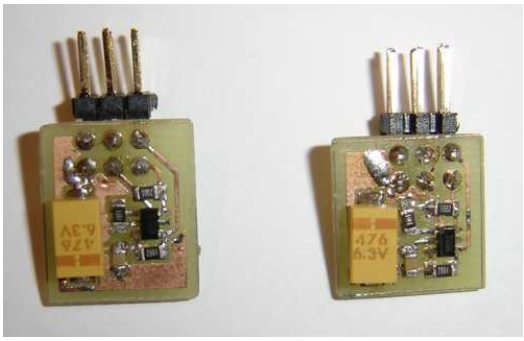
Obr. 14.4: Osazovací výkres modulu s AD7414



Obr. 14.5: Deska plošných spojů modulu s AD7415 (pohled ze strany součástek)



Obr. 14.6: Osazovací výkres modulu s AD7415



Obr.: 14.7: Sestrojené moduly

15 Závěr

Zpracováním bakalářské práce jsem si osvojil znalosti v oblasti konfigurace logických obvodů v jazyce VHDL a prostředí programu Xilinx ISE 9.2i. Konkrétně s tímto jazykem a programátorským prostředím jsem se setkal vůbec poprvé.

Cílem bylo vytvoření konfigurace programovatelného logického obvodu CPLD, který bude komunikovat s navrhnutým modulem pro měření teploty pomocí dvoudrátové sběrnice SMBUS. Program má zpracovávat data přijímaná od teplotního senzoru. Mezi jeho funkce patří zobrazování aktuální, maximální, minimální a průměrné teploty na LED sedmisegmentovém displeji. Dále by měl umět přijímaná data odesílat pomocí rozhraní UART do počítače nebo jiných podobných zařízení. Dalším bodem zadání byl návrh a konstrukce modulu pro měření teploty a s tím i spjatý výběr vhodného senzoru. Posledním úkolem bylo zobrazení teploty v PC a její zaznamenávání.

Z množství teplotních senzorů byl vybrán senzor s označením AD7414 (AD7415) firmy Analog Devices, který splňuje požadované vlastnosti. Modul, který jej obsahuje byl navržen s malými úpravami podle doporučeného zapojení vyrábějící firmy. Program navržený v jazyce VHDL byl bez problémů simulován. Simulace probíhaly podle teoretických předpokladů. Pro odladění navržených programů mi byla vedoucím práce poskytnuta vývojová deska nesoucí název X2C-XL. Deska obsahuje dva programovatelné logické obvody (CoolRunner-II XC2C256 CPLD a XC9572XL CPLD). Při návrhu UART komunikace jsem však zjistil, že zmíněná vývojová deska byla nevyhovující, jelikož neobsahovala port RS-232. Proto byla vyměněna za jinou, která nekoliduje se zadáním práce. Nová deska s názvem SPARTAN-3 však nese místo obvodu CPLD obvod FPGA (XC3S200FT256), což je v nesouladu se zadáním, ale na bakalářkou práci to nemá vliv. Komunikace mezi vývojovou deskou a počítačem (UART vysílač) pracuje bez problémů. Hodnoty jsou zobrazovány v PC pomocí programu HyperTerminal, kde jsou také ukládány. Bohužel se však nepodařilo zprostředkovat komunikaci mezi vývojovou deskou s logickým obvodem a modulem pro měření teploty.

Problém jsem řešil nejen se svým vedoucím práce ale i s jinými odborníky, nic ale nevedlo ke zdárnému konci. Možnou příčinou je špatné časování a synchronizace povelů na sběrnici resp. vyčítání hodnoty z příslušného registru teplotního senzoru. Zdrojový kód, který toto provádí jsem implementoval do programu pro výběr čidla. Z důvodu nefunkčnosti jsem ho v textu bakalářské práce neuvedl. Jeho verze jsou však umístěny na přiloženém CD. Zapojení modulu je shodné s doporučeným zapojením vyrábějící firmy Analog Devices a tudíž se domnívám, že je správné. Jakékoliv jiné mnou vytvořené programy implementované do obvodu FPGA byly funkční, avšak tyto programy byly založeny buď na přijmu nebo vysílání dat a neprováděli obě funkce v jednom programu současně.

Seznam literatury

[1] KOLOUCH, J.: *Programovatelné logické obvody.* [Skriptum FEKT VUT v Brně.] Brno, 2005

[2] PINKER J., POUPA M.: *Číslicové systémy a jazyk VHDL*. BEN – technická literature, Praha 2006

[3] Inteligentní digitální senzory teploty Analog Device**,** Dostupné na WWW: <http://automatizace.hw.cz/view.php?cisloclanku=2006010101>

[4] Volnei A. Pedroni: *Circuit Design With VHDL*. MIT Press, Cambridge, massachusetts, London, England 2004

[5] Součástky Xilinx, Dostupné na WWW: <http://www.asix.cz/xilsystems.htm>

[6] Datasheet obvodu LM73, Dostupný na WWW: <http://automatizace.hw.cz/storage/LM73.pdf>

[7] Datasheet obvodu LM95231, Dostupný na WWW: <http://automatizace.hw.cz/storage/LM95231.pdf>

[8] Datasheet obvodu MAXIM DS1631, Dostupný na WWW: <http://www.grifo.com/PRESS/DOC/Dallas/DS1631.pdf>

[9] Datasheet obvodu AD7414, Dostupný na WWW: <http://www.analog.com/UploadedFiles/Data_Sheets/266781917AD7414_7415.pdf>

[10] Maxim-IC. Sunnyvale, CA. Thermal Management, Sensors, and Sensor Conditioners. 2007. Firemní webové stránky. Dostupné na <http://www.maxim-ic.com/Sensors.cfm>

[11] Vývojová deska XC2-XL. Dostupná na WWW: <http://digilentinc.com/Products/Detail.cfm?Prod=XC2XL&Nav1=Products&Nav2=Programmable>

[12] Programové prostředí Xilinx ISE 9.2i. Dostupné na WWW: <http://www.xilinx.com/ise/logic_design_prod/webpack.htm >

[13] Vývojová deska SPARTAN-3. Dostupná na WWW: <http://digilentinc.com/Products/Detail.cfm?Prod=S3BOARD&Nav1=Products&Nav2=Programmable>

[14] Měření teploty v průmyslu, Dostupné na WWW: <http://www.hw.cz/Teorie-a-praxe/Dokumentace/ART1149-Mereni-teploty-v-prumyslu.html>

[15] Programovatelné logické obvody, Dostupné na WWW: <http://sweb.cz/fpga/>

[16] System Management Bus Specification, Revision 1.1, 1998, Dostupné na WWW: <http://alpha1.dyns.net/files/SMBus/smbus110.pdf>

[17] SPARTAN-3 Board, Dostupné na WWW: <http://digilentinc.com/Products/Detail.cfm?Prod=S3BOARD&Nav1=Products&Nav2=Programmable>

[18] SPARTAN-3 Board, Reference manual, Dostupné na WWW: <http://digilentinc.com/Data/Products/S3BOARD/S3BOARD_RM.pdf>

[19] Návrhový systém Xilinx ISE, Dostupné na WWW: <http://www.onic.cz/university/36dp/DP7.doc>

[20] UART, Dostupné na WWW: <http://en.wikipedia.org/wiki/UART>

[21] RS232, Dostupné na WWW: <http://en.wikipedia.org/wiki/RS232>

Seznam příloh

A Zdrojový kód sériově paralelního převodníku

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity AD_prevodnik is
	PORT (	clk: IN std_logic;
		clear: IN std_logic;
		ser_in: IN std_logic;
		send_uart: OUT std_logic;
		parallel: OUT std_logic_vector(7 downto 0));
end AD_prevodnik;
architecture Behavioral of AD_prevodnik is
	SIGNAL shift_reg: std_logic_vector (7 DOWNTO 0);
	SIGNAL count: integer :=0;

begin
PROCESS (clk,ser_in,clear)
	begin
		if (clk'event and clk = '1') then
			if clear='1' then
				shift_reg(0) <= ser_in;
				shift_reg(7 downto 1) <= shift_reg(6 downto 0);
				count<=count+1;
					if count = 8 then
						parallel<=shift_reg;
						send_uart<=ser_in;
						count<=1;
					end if;
			else
				shift_reg <= "00000000";
				count<=0;
			end if;
		end if;
end PROCESS;
end Behavioral;
```

B Zdrojový kód převodníku binárního kódu na kód BCD

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity prevodnik_kodu is
	PORT (	BinIn: IN std_logic_vector (7 DOWNTO 0);
		BCDoutp: OUT std_logic_vector (15 DOWNTO 0));
end prevodnik_kodu;

architecture Behavioral of prevodnik_kodu is

begin
PROCESS(BinIn) -- DUT ARCH2
	VARIABLE Sum: std_logic_vector(15 DOWNTO 0);

	BEGIN
		-- BinIn(2 DOWNTO 0) – initialization ------------------------
		Sum := (2=>BinIn(2),1=>BinIn(1),0=>BinIn(0),OTHERS=>'0');
		-- BinIn(3) – 1-st stage: Add 8hex, BCDout max. 15 -----------
		IF (BinIn(3) = '1') THEN Sum := Sum + 8;
			IF Sum > "1001" THEN Sum := Sum + 6; END IF;
			END IF;
		-- BinIn(4) – 2-nd stage: Add 16hex, BCDout max. 31 ----------
		IF (BinIn(4) = '1') THEN Sum := Sum + X"16";
			IF Sum(3 DOWNTO 0) > "1001" THEN Sum := Sum + 6; END IF;
			END IF;
		-- BinIn(5) – 3-rd stage: Add 32hex, BCDout max. 63 ----------
		IF (BinIn(5) = '1') THEN Sum := Sum + X"32";
			IF Sum(3 DOWNTO 0) > "1001" THEN Sum := Sum + 6; END IF;
			END IF;
		IF (BinIn(6) = '1') THEN Sum := Sum + X"64";
			IF Sum(3 DOWNTO 0) > "1001" THEN Sum := Sum + 6; END IF;
			END IF;
		IF (BinIn(7) = '1') THEN Sum := Sum + X"128";
			IF Sum(3 DOWNTO 0) > "1001" THEN Sum := Sum + 6; END IF;
			END IF;
		BCDoutp <= Sum;
END PROCESS;
end Behavioral;
```

C Zdrojový kód programu pro ovládání displeje

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity dysplej is
	port(	clk50: in std_logic;
		khertz_en: in std_logic;
		 bcdint: in std_logic_vector(15 downto 0);
		dp3,dp2,dp1,dp0: in std_logic;
		digit: out std_logic_vector(3 downto 0);
		seg: out std_logic_vector(7 downto 0));
end dysplej;

architecture Behavioral of dysplej is
	signal cd: std_logic_vector(1 downto 0);
	signal curr: std_logic_vector(3 downto 0);
	signal dp: std_logic;

begin
	-- dp3 <= '1'; -- zadna des.tecka nesviti pro dp = '1'
	-- dp2 <= '1';
	-- dp1 <= '1';
	-- dp0 <= '1';
process (clk50) begin
	-- if rst = '1' then -- reset neni potrebny
	-- seg <= (others => '1');
	-- digit <= (others => '1');
	-- cd <= (others => '0');
	-- curr <= (others => '0');
if clk50'event and clk50 = '1' then
		if khertz_en = '1' then
			cd <= cd + 1;
		end if;
	case cd(1 downto 0) is -- curr je zobr. cifra, digit je jeji pozice na dosol.
		when "00" =>   curr <= bcdint(3 downto 0);   digit <= "1110"; dp <= dp0;
		when "01" =>   curr <= bcdint(7 downto 4);   digit <= "1101"; dp <= dp1;
		when "10" =>   curr <= bcdint(11 downto 8);  digit <= "1011"; dp <= dp2;
		when others => curr <= bcdint(15 downto 12); digit <= "0111"; dp<=dp3;
	end case ;
	case curr is
		when "0000" => seg <= "0000001" & dp; -- 0
		when "0001" => seg <= "1001111" & dp; -- 1
		when "0010" => seg <= "0010010" & dp; -- 2
```

```
                when "0011" => seg <= "0000110" & dp; -- 3
                when "0100" => seg <= "1001100" & dp; -- 4
                when "0101" => seg <= "0100100" & dp; -- 5
                when "0110" => seg <= "0100000" & dp; -- 6
                when "0111" => seg <= "0001111" & dp; -- 7
                when "1000" => seg <= "0000000" & dp; -- 8
                when others => seg <= "0000100" & dp; -- 9
        end case;
end if;
end process;
end Behavioral;
```

D Zdrojový kód děličky clk pro ovládání displeje

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity delicka_clk1 is
	Port (  Clk:   in  STD_LOGIC;
		KHzEn: out STD_LOGIC);
end delicka_clk1;
architecture Behavioral of delicka_clk1 is
	signal mhertz_count,khertz_count: std_logic_vector(9 downto 0);
	signal mhertz_en,khertz_en: std_logic;

begin
	-- generates a 1 Mhz signal from a 50 Mhz signal - mhertz_en
	process (clk) begin
		if clk'event and clk = '1' then
			mhertz_count <= mhertz_count + 1 ;
			if mhertz_count = "110010" then
				mhertz_en <= '1' ;
				mhertz_count <= (others => '0') ;
			else
				mhertz_en <= '0' ;
			end if ;
		end if ;
	end process ;
	-- generates a 1 kHz signal from a 1Mhz signal - khertz_en
	process (clk) begin
		if clk'event and clk = '1' then
			if mhertz_en = '1' then
				khertz_count <= khertz_count + 1 ;
				if khertz_count = "1111101000" then
					khertz_en <= '1' ;
					khertz_count <= (others => '0') ;
				else
					khertz_en <= '0' ;
				end if ;
			else
				khertz_en <= '0' ;
			end if ;
		end if ;
	end process ;
	KHzEn <= khertz_en;
end Behavioral;
```

E Zdrojový kód přepínače indikované hodnoty

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity prepinac is
	Port (  clk: in std_logic;
		actual: in std_logic_vector (7 downto 0);
		maximum: in std_logic_vector (7 downto 0);
		minimum: in std_logic_vector (7 downto 0);
		prumer: in std_logic_vector (7 downto 0);
		prep_minimum : in STD_LOGIC;
		prep_maximum : in STD_LOGIC;
		prep_prumer : in STD_LOGIC;
		na_displej: out std_logic_vector (7 downto 0));
end prepinac;

begin
process(clk, prep_minimum, prep_maximum, prep_prumer) begin
	if clk'event and clk = '1' then
		na_displej<=actual;
		if prep_minimum ='1' then
			na_displej<=minimum;
		elsif prep_maximum ='1' then
			na_displej<=maximum;
		elsif prep_prumer ='1' then
			na_displej<=prumer;
		end if;
	end if;
end process;
end Behavioral;
```

F Zdrojový kód programu pro indikaci maximální hodnoty

```
library IEEE;
        use IEEE.STD_LOGIC_1164.ALL;
        use IEEE.STD_LOGIC_ARITH.ALL;
        use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity maximum3 is
        Port (  Clk:   in  STD_LOGIC;
                parallel: in std_logic_vector (7 downto 0);
                maximum: out std_logic_vector (7 downto 0):="00000000");
end maximum3;

architecture Behavioral of maximum3 is
        signal count: std_logic_vector(3 downto 0):= (others => '0');
        signal parallel_uns, registr: unsigned (7 downto 0):= (others => '0');

begin
process (clk) begin
        if clk'event and clk = '1' then
                count <= count + 1 ;
                parallel_uns<=unsigned(parallel);
                if count = "1000" then
                        if parallel_uns>registr                 then
                                registr<=parallel_uns;
                        end if;
                        count <="0001";
                end if ;
        end if ;
        maximum<= std_logic_vector (registr(7 downto 0));
end process ;
end Behavioral;
```

G Zdrojový kód programu pro indikaci minimální hodnoty

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity minimum1 is
	Port (  Clk: in STD_LOGIC;
		parallel: in std_logic_vector (7 downto 0);
		minimum: out std_logic_vector (7 downto 0):="00000000");
end minimum1;

architecture Behavioral of minimum1 is
	signal count: std_logic_vector(3 downto 0):= (others => '0');
	signal parallel_uns: unsigned (7 downto 0):= (others => '0');
	signal registr: unsigned (7 downto 0):="01111101";

begin
process (clk) begin
	if clk'event and clk = '1' then
		count <= count + 1 ;
		parallel_uns<=unsigned(parallel);
		if count = "1000" then
			if parallel_uns<registr		then
				registr<=parallel_uns;
			end if;
			count <="0001";
		end if ;
	end if ;
	minimum<= std_logic_vector (registr(7 downto 0));
end process ;
end Behavioral;
```

H Zdrojový kód programu pro indikaci průměrné hodnoty

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity prumer3 is
Port (  Clk: in  STD_LOGIC;
                parallel: in std_logic_vector (7 downto 0);
                prumer: out std_logic_vector (7 downto 0):="00000000");
end prumer3;

architecture Behavioral of prumer3 is
        signal count: std_logic_vector(28 downto 0):= (others => '0');
        signal soucet_hodnot, parallel_int, prumer_int: integer:=0;
        signal count_hodnot:integer:=1;

begin
process (clk) begin
        if clk'event and clk = '1' then
                count <= count + 1;
                parallel_int<=conv_integer(parallel);
                if count = "10110010110100000101111000000" then
                        soucet_hodnot<=soucet_hodnot+parallel_int;
                        prumer_int<=soucet_hodnot/count_hodnot;
                        count_hodnot <= count_hodnot + 1;
                        if prumer_int>32500 or prumer_int<-32500  then
                                prumer_int<=0;
                        end if;
                        count <="0001";
                end if ;
        end if ;
        prumer<= conv_std_logic_vector(prumer_int,8);
end process ;
end Behavioral;
```

I Zdrojový kód UART vysílače

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity uart_transm is
	Port (  clk : in  STD_LOGIC;
		rst : in  STD_LOGIC;
		snd : in  STD_LOGIC;
		data_in : in  STD_LOGIC_VECTOR (7 downto 0);
		tx_out : out  STD_LOGIC);
end uart_transm;

architecture Behavioral of uart_transm is
	signal data : STD_LOGIC_VECTOR (8 downto 0);
	signal cnt : STD_LOGIC_VECTOR (3 downto 0);
	signal vys : std_logic := '0';		-- nemuselo by být

begin
process(clk,rst,snd,data_in) begin
	if rst ='1' then
		tx_out<='1';
		cnt<="0000";
		vys<='0';
	elsif snd ='1' then
		data <= data_in & '0';
		cnt<="0000";
		vys<='1';
	elsif clk'event and clk='1' then
		if vys ='1' then
			tx_out<=data(0);
			cnt<=cnt+1;
			if cnt="1000" then
				vys<='0';
				cnt<="0000";
			end if;
			data(7 downto 0) <= data(8 downto 1);
		else
			data<="000000000";
			tx_out<='1';
		end if;
	end if;
end process;
end Behavioral;
```

J Zdrojový kód děličky clk pro UART

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity delicka_uart is
	Port (  Clk:   in  STD_LOGIC;
		devetKHzEn: out STD_LOGIC);
end delicka_uart;

architecture Behavioral of delicka_uart is
signal mhertz_count,khertz_count: std_logic_vector(6 downto 0):= (others => '0');
signal mhertz_en,devetkhertz_en: std_logic:='0';

begin
	-- generates a 1 Mhz signal from a 50 Mhz signal - mhertz_en
	process (clk) begin
		if clk'event and clk = '1' then
			mhertz_count <= mhertz_count + 1 ;
			if mhertz_count = "011000" then
				mhertz_en <= NOT mhertz_en ;
				mhertz_count <= (others => '0') ;
			end if ;
		end if ;
	end process ;

-- generates a 9615,4 Hz signal from a 1Mhz signal - khertz_en
	process (mhertz_en) begin
		if mhertz_en'event and mhertz_en = '1' then
			khertz_count <= khertz_count + 1 ;
			if khertz_count = "0110011" then
				khertz_count <= (others => '0') ;
				devetkhertz_en <= not(devetkhertz_en) ;
			end if ;
		end if ;
	end process ;
	devetKHzEn <= devetkhertz_en;
end Behavioral;
```

K Zdrojový kód programu pro výběr čidla

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity adresace is
	port(	clk: in std_logic;
		ser_out:out std_logic;
		Sw4:	IN std_logic_vector(7 downto 0));
end adresace;

architecture Behavioral of adresace is
	signal save_reg:std_logic_vector(28 downto 0);

begin
process(clk)
	begin
		if Sw4(0)='1' then
			save_reg<="10001000100100000000000110010"; --AD7414Vdd
		elsif clk'event and clk='1' then
			save_reg(27 downto 0)<= save_reg (28 downto 1);
		end if;
		if Sw4(1)='1' then
			save_reg<="10001000100100000000010110010"; --AD7415GND
		elsif clk'event and clk='1' then
			save_reg(27 downto 0)<= save_reg (28 downto 1);
		end if;
end process;
ser_out<=save_reg(0);
end Behavioral;
```

L Zdrojový kód děličky clk pro modul pro měření teploty

```
library IEEE;
	use IEEE.STD_LOGIC_1164.ALL;
	use IEEE.STD_LOGIC_ARITH.ALL;
	use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity delickamodul1 is
	Port (  Clk: in  STD_LOGIC;
		stoKHzEn: out STD_LOGIC);
	end delickamodul1;
architecture Behavioral of delickamodul1 is
signal mhertz_count,stokhertz_count: std_logic_vector(6 downto 0):=(others =>'0');
signal mhertz_en,stokhertz_en: std_logic:='0';

begin
	-- generates a 1 Mhz signal from a 50 Mhz signal - mhertz_en
	process (clk) begin
		if clk'event and clk = '1' then
			mhertz_count <= mhertz_count + 1 ;
			if mhertz_count = "011000" then
				mhertz_en <= NOT mhertz_en ;
				mhertz_count <= (others => '0') ;
			end if ;
		end if ;
	end process ;

	-- generates a 1kHz signal from a 1Mhz signal
	process (mhertz_en) begin
		if mhertz_en'event and mhertz_en = '1' then
			stokhertz_count <= stokhertz_count + 1 ;
			if stokhertz_count = "0000100" then
				stokhertz_count <= (others => '0') ;
				stokhertz_en <= not(stokhertz_en) ;
			end if ;
		end if ;
	end process ;
	stoKHzEn <= stokhertz_en;
end Behavioral;
```

M Schéma celého programu