ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

BAKALÁŘSKÁ PRÁCE

MOBILNÍ LOGICKÝ ANALYZÁTOR REALIZOVANÝ POMOCÍ FPGA SPARTAN 3

2006 **Jiří Svozil**

Poděkování: Tímto bych rád poděkoval vedoucímu mé bakalářské práce panu Ing. Jiřímu Kadlecovi CSc. za pomoc, ochotu a především trpělivost.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ……. …….

podpis

Anotace:

Cílem této bakalářské práce je za pomoci programovatelného hradlového pole Spartan 3 XC3S1500L navrhnout a následně realizovat logický analyzátor. Analyzátor má 16 datových vstupů a má umožňovat základní funkce jako je měření hodnot v časové oblasti, měření hodnot signálů v ose y a uživatelem volitelné spouštěcí podmínky měření. Primárním výstupem přístroje je monitor standartu VGA. Naměřené průběhy lze ukládat do paměti flash umístěné na desce RC10, odkud je lze následně přenášet do PC pro další zpracování.

Annotation:

Aim of this bachelors work is to realize digital logical analyzer with FPGA Spartan 3 XC3S1500L. Analyzer should have 16 data inputs and should support the basic function like measuring values in time, visualization of data and support for optional trigger conditions. Prime exit of the device is a standard VGA monitor. Measured data sequences can be save to the flash memory placed on board RC10. Data can be subsequently transfered to the PC for other processing.

Obsah:

Úvod

Cílem této bakalářské práce je za pomoci programovatelného hradlového pole Spartan 3 XC3S1500L navrhnout a následně realizovat mobilní logický analyzátor. Analyzátor má umožňovat pouze základnější funkce, ale jeho hlavní výhodou je jeho rychlost, kterou poskytuje hradlové pole a dále jeho malé rozměry. Čip Spartanu 3 je totiž umístěn na desce RC10 Pilot společnosti Celoxica, která svými parametry předurčuje jeho vysokou mobilitu a tedy možnost měření i mimo běžné podmínky laboratoře.

Hlavní zobrazovací jednotkou je standardní VGA monitor. K ovládání přístroje je použit joystick umístěný na kartě RC10. Ovládání se děje přes řádkové menu přístroje, kde se pohybem po jednotlivých položkách dají nastavovat veškeré parametry přístroje. Vstupem měřených dat je 16 pinů z rozhraní ATA také umístěného na desce. Pro přenášení naměřených dat do počítače, ale také pro napájení přístroje je využito rozhraní USB.

Při vytváření mé práce jsem vycházel z již z existujícího vzorového příkladu adc.hcc, který je součástí sady ukázkových příkladů pro kartu RC10. Ukázkový program je demonstrací čtení a následného zobrazování signálů přivedených na ADC převodníky. Tento návrh mi však sloužil spíše jako náhled do principů programování FPGA v jazyce Handel-C, avšak ve výsledné verzi mého programu byl opravdu použit pouze princip pro obsluhu zobrazování na monitor, který byl rozšířen a upraven tak, aby vyhovoval potřebám pro zadaný návrh.

1. Použitý hardware

1.1. FPGA Spartan 3 XC3S1500L

Jádrem použitého hardwaru je programovatelné hradlové pole Spartan 3 XC3S1500L-4 od společnosti Xilinx, což je i důvod, proč jsem tuto kapitolu volil jako úvodní. Níže jsou však rozebírány pouze základní vlastnosti a technické parametry čipu, a to pouze ty, které považuji za důležité a pro daný návrh zásadní.

Řada Spartan 3 navazuje na úspěšnou předchozí sérii Spartan-IIE. Od této architektury se liší v počtu systémových hradel a logických buněk, velikostí interních bloků RAM, počtu I/O a implementací některých nových bloků do architektury jako jsou DCMs (Digital Clock Manager) nebo násobičky. Největší změna však nastala především v použité technologii, a to v přechodu z 150nm na 90nm.

1.1.1. Hlavní rysy architektury SPARTAN-3L

Hovoříme-li o konkrétním typu čipu Spartan 3 XC3S1500L-4, je třeba vyzdvihnout jeho největší přednost, kterou je nízkospotřebovost. Má totiž až o 60% nižší klidový proud ve standardním provozu a o 99% nižší klidový proud v spánkovém režimu než jeho alternativy ze série Spartan 3.

- až 1,5 miliónu systémových hradel
  - ekvivalentně odpovídají 29952 logických buněk
- 487 I/O pinů
  - s podporou 18 úrovňových standardů
  - podpora DDR (Double Data Rate)
- logické prvky
  - 32 integrovaných násobiček 18 x 18
  - JTAG kompatibilní s IEEE 1149.1/1532
  - široké multiplexory
  - integrovaná struktura pro konstrukci rychlých sčítaček
- RAM
  - 576Kb blokové RAM
  - 208Kb distribuované RAM
- DMC
  - 4x DMC (Digital Clock Manager)
  - umožňující digitální řešení distribuce hodinového signálu
- 8 globálních hodinových linek

1.1.2. Architektura obvodu

Architektura je složena z 5 základních funkčních bloků:

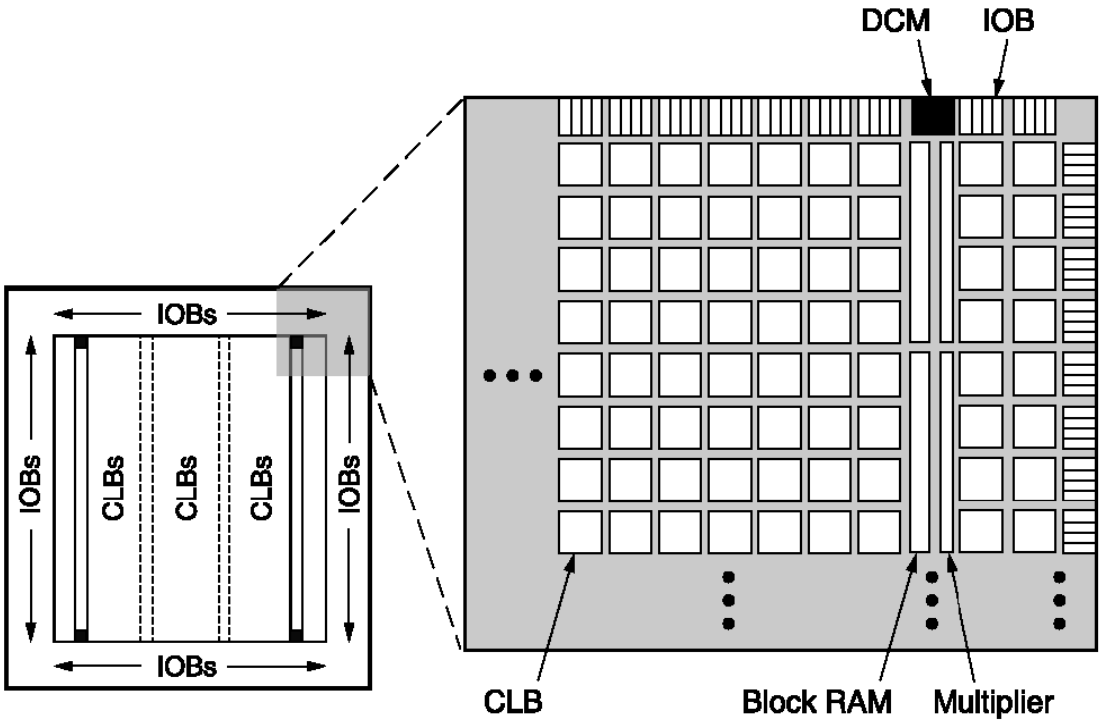
- **CLBs** (Configurable Logic Blocks) konfigurovatelné logické bloky obsahující LUTs (Look-Up Tables) na principu paměti RAM. Těmito bloky jsou tvořeny logické, aritmetické a paměťové funkce.

- **IOBs** (Input/Output Blocks) řídí tok dat mezi I/O pinem a vnitřní logikou. IOBs SPARTAN-3 umožňuje vybrat jeden ze 17 možných úrovňových standardů pro jednotlivé piny (single-ended) a jeden ze 6 pro rozdílový výstup (differential). Struktura IOBs obsahuje 3 základní signálové cesty: vstupní, výstupní a 3-stavová.

- **Block RAM –** bloková paměť RAM je uspořádaná po blocích velikosti 18Kbit. Pro uložení velkého množství dat je efektivnější použít blokovou RAM než distribuovanou RAM. Kapacita paměti je 18432bitů bez parity nebo 16384bitů s paritou. Paměť také umožňuje dvouportový přístup – může být konfigurována jako Single-Port, nebo Dual-Port RAM. Paměť je na čipu rozmístěna po blocích tvořící sloupce.

- **Multiplier** blocks **-** obsahují násobičky 18x18 bitů s 36bitovým výstupem. Mohou pracovat jako asynchronní i jako synchronní. Vstupní datové slovo může být reprezentováno dvojkovým doplňkem (buď 18bitové znaménkové nebo 17bitové neznaménkové). Kaskádním řazením lze vytvořit násobičky pracující s širším datovým slovem než jen 18bitů.

- **DCM** (Digital Clock Manager) řídí, upravuje a distribuuje hodinový signál po celém čipu. Plní 3 základní funkce: eliminace časového zpoždění, frekvenční syntéza, fázový posun signálu.



Obr. 1 : Architektura základních bloků FPGA

1.2. Deska RC10 Pilot

Pokud jsem v minulé kapitole hovořil o jádru celého hardwaru, o FPGA Spartan 3, v této části bych rozebral také jeho zbývající okolí. Pro celou práci jsem totiž využíval vývojovou desku RC10 Pilot od společnosti Celoxica. Tato je osazena právě čipem Spartan 3 XC3S1500L-4.

RC10 je vývojová deska koncipována tak, aby i méně zkušeným uživatelům umožnila velmi snadné učení, vyvíjení a rychlou tvorbu prototypů v metodě přenosu jazyka C na hardware. Na druhé straně její potenciál, vztáhnuto na periférie a celkové osazení, umožňuje vytvářet i velmi náročné a složité aplikace. Velká opora pro uživatele je i v softwarovém vybavení, které je uživatelsky velmi příjemné, a k dispozici je i řada vzorových příkladů. Konkrétnějšímu popisu použitého softwaru se věnuje kapitola *2. Softwarové vybavení.*

Obr. 2 : Deska RC10 Pilot

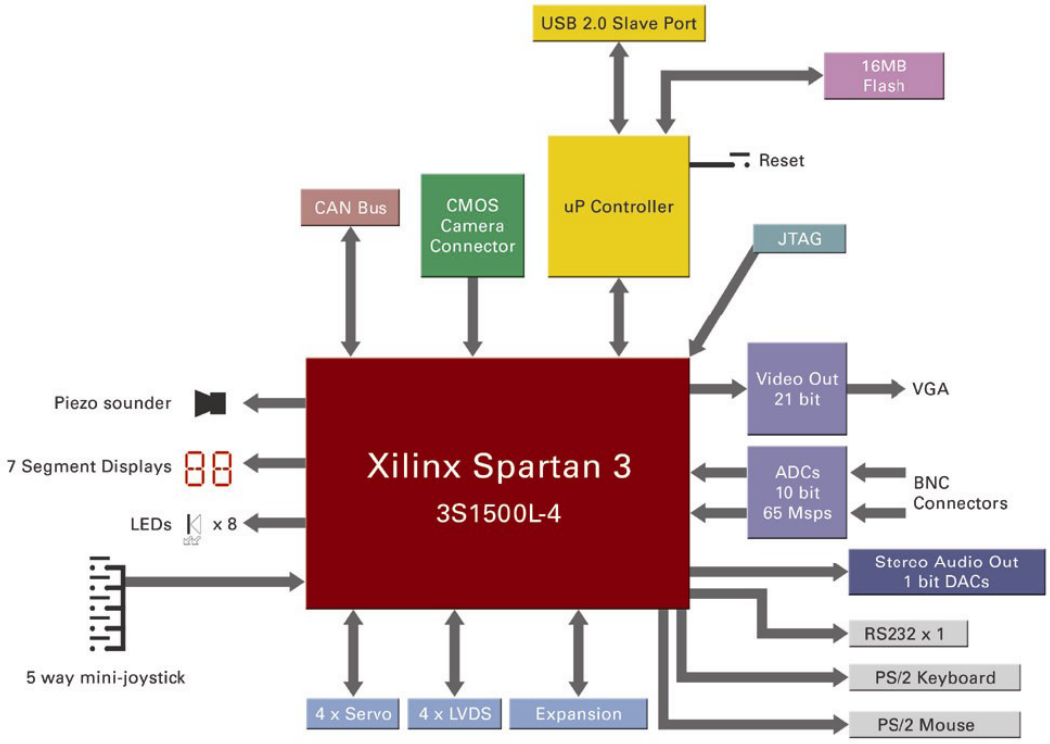
1.2.1. Přehled vybavení RC10

Dříve než se budu věnovat těm částem desky, které jsem pro návrh použil a které bych rád rozebral podrobněji, uvádím stručný přehled vybavení a periferií desky RC10 Pilot. Chtěl bych tím totiž poukázat na to, jak velký potenciál pro další vývoj a rozšíření použitá deska poskytuje.

Seznam funkčních bloků desky RC10 Pilot:

- Xilinx Spartan 3
- Dva 10 bitové ADC převodníky s rychlostí 65Msps
- 24 bitový VGA výstup
- Konektor pro 2Mpixel CMOS kameru
- 16MByte Flash paměti
- CANbus konektor
- Mikrokontrolér sloužící pro:
    - USB 2.0 komunikaci
    - FPGA konfiguraci a rekonfiguraci
    - správu Flash paměti
- sériový port RS232
- PS/2 port pro klávesnici a myš
- 8 – uživatelem programovatelných LED
- 2 sedmi-segmentové displeje

- piezo reproduktor
- 50 rozšiřujících pinů obsahující :
    - 33 hlavních I/0 pinů (kompatibilních s ATA UDMA-4 nebo vyšší)
    - 3 napájecí piny (+12V, +5V, +3.3V)
    - 2 hodinové piny
- JTAG konektor
- konektor pro řízení až pro 4 servomotorů
- Stereo audio výstup (1 bitové DAC)



Obr. 3 : Blokové schéma desky RC10

(zdroj www.celoxica.com)

1.2.2. Využité části RC10

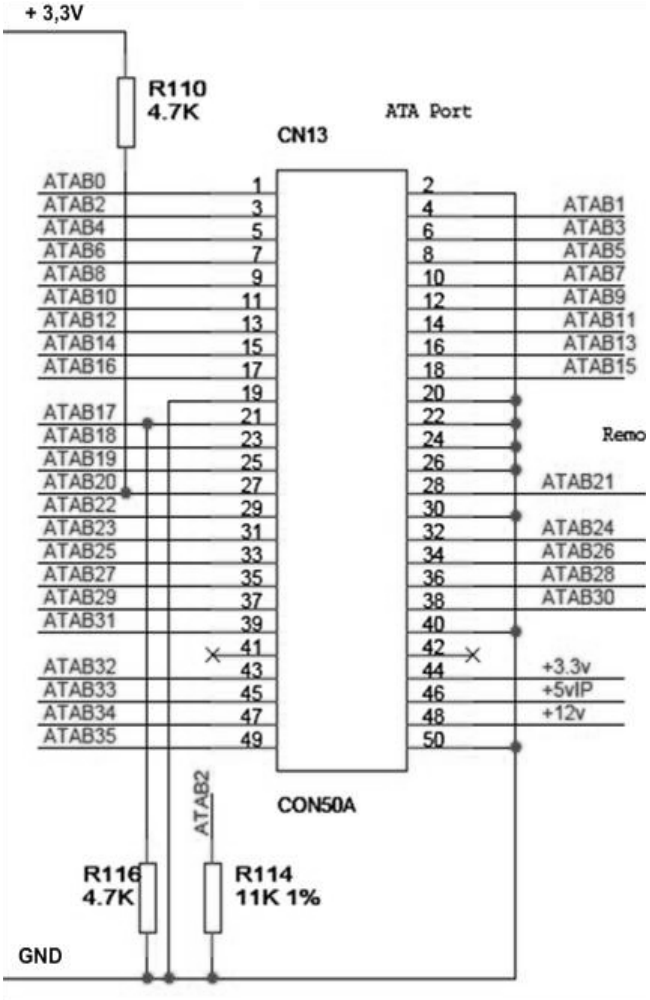
Použitá vývojová deska obsahuje velké množství funkčních částí. V této kapitole tedy podrobně uvádím ty, které jsem využíval při tvorbě mé práce. Níže následují jednotlivé části s krátkým popisem, které však záměrně neobsahují informace o tom, jakým způsobem s danými periferiemi pracuji, případně jak se provádí jejich obsluha, neboť se jimi zabývá kapitola *3.2. Popis realizovaných funkčních částí analyzátoru.*

Joystic

Joystic neboli kolébkové tlačítko je umístěno v dolní části desky (viz *Obr. 5*) a v návrhu je to jediný ovládací prvek logického analyzátoru jako takového. Je využíván především proto, že je již součástí přípravku, a přitom uživateli umožňuje velmi příjemné ovládání bez nutnosti připojení dalších periférií jako je myš nebo klávesnice. Tím se celý přípravek stává mobilnějším. Prvek umožňuje pět poloh, které lze rozpoznat. Čtyři základní polohy, do kterých se dá naklápět (nahoru, dolu, doleva doprava) plus jeho stisk.

ATA port

Rozhraní obsahuje celkem 48 funkčních pinů, přičemž každý pin může fungovat jako vstupní tak i jako výstupní. Standardně jsou piny nastaveny jako vstupní a konfigurovány na 3,4V LVCMOS. Při návrhu jsem však využil pouze 16 pinů, a to jako vstupní, a 4 jako výstupní. Piny 3-18 jsou tedy použity jako vstupy pro logický analyzátor a piny 39, 37, 35, 33 jsou použity jako výstup pro generátor signálů, který slouží k testování analyzátoru.



Obr. 4 : Zapojení ATA portu na desce RC10

(zdroj www.celoxica.com)

Videovýstup

RC10 má implementovány dva videovýstupy VGA a TFT. Tento návrh využívá pouze VGA, který je fyzicky 21 bitový, a to 7 bitů pro každou barvu (červená, zelená a modrá), ale programově se z důvodu kompatibility užívá 24 bitově. Zmiňovaný převod je uskutečňován zanedbáním posledního bitu. Rozsah rozlišení se pohybuje od 640x480 pixelů při obnovovací frekvenci 60Hz až do 1600x1200 pixelů při frekvenci 85Hz. Rozlišení je odvozeno od frekvence hodinového signálu. Ten je v případě tohoto návrhu 64MHz, čemuž odpovídá rozlišení 1024x768 pixelů a obnovovací frekvenci 59Hz.

Mikrokontrolérové rozhraní, USB

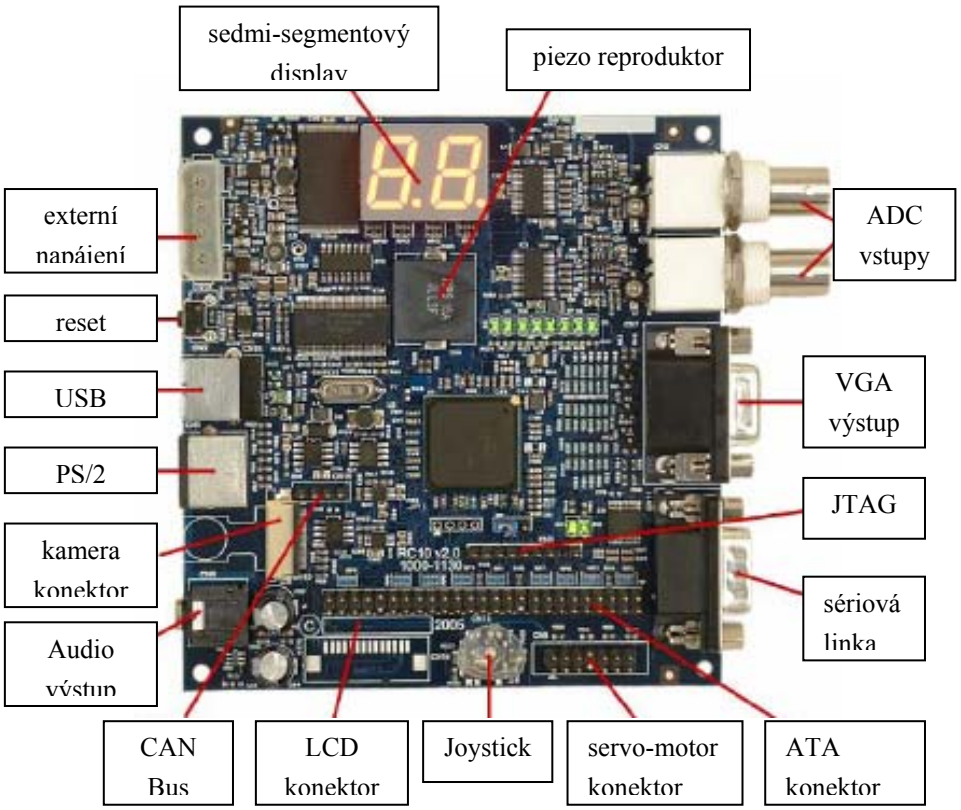
Pomocí mikrokontroléru Cypress CY7C68013-56pvc FX2 umístěného na desce je realizována komunikace jednak mezi počítačem připojeném přes port USB a jednak komunikace FPGA čipu s pamětí Flash též umístěné na desce. Rozhraní USB je standartu USB 2.0 a umožňuje uživateli dosažení přenosové rychlosti až 25 MB/s. Další důležitou funkcí, ke které je rozhraní používáno, je napájení samotné desky RC10, čímž se celý přípravek stává nezávislý na externím napájení.

Flash paměť

Paměť flash RAM je o velikosti 16MB a je přístupná jak z FPGA tak i přes rozhraní USB pouze přes výše zmíněný mikrokotrolér Cypress. Paměť je organizována do jednoduchého file-systému o 254 položkách a této práci je používána především k ukládání naměřených průběhů, jejich zpětnému zobrazování a přenášení přes USB rozhraní do počítače.

Sedmi-segmentový displej

Tento displej je tvořen dvěma sedmi-segmentovými LED displeji, umístěnými v horní části desky (viz *Obr. 5*). Nicméně ve výsledné verzi projektu není používán pro prioritní zobrazování, ale slouží spíše jako informativní číselník pro hodnoty použité v menu analyzátoru.



Obr. 5 : RC10 popis osazení

(zdroj www.celoxica.com)

2. Softwarové vybavení

2.1. DK Design Suite

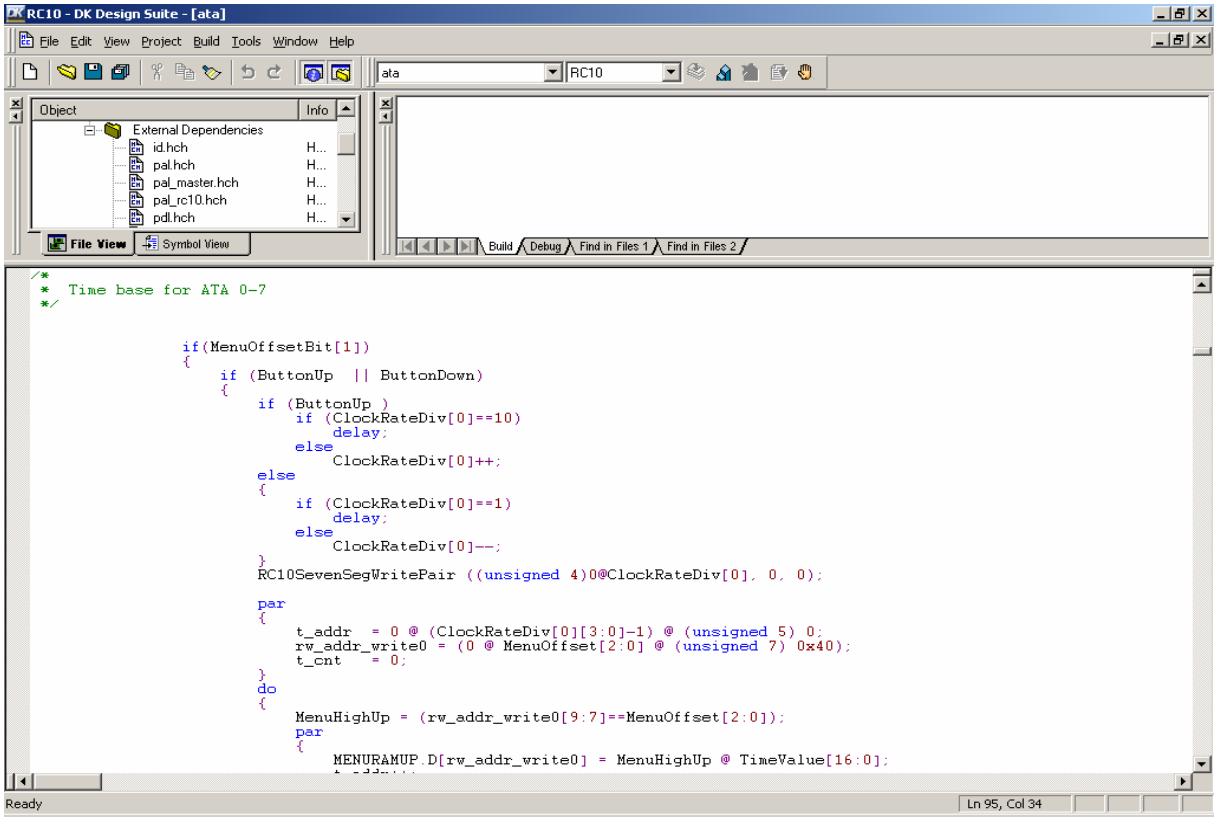
Pro psaní kódu programu, ze kterého se pak překladem vytvořil hardware realizující logický analyzátor, jsem využíval vývojové prostředí DK Design Suite od společnosti Celoxica. Toto prostředí a jazyk Handel-C jsou velice efektivní při vytváření a přenášení aplikací do FPGA/PLD. Pro programátora je však zajímavé především tím, že obsahuje integrovanou podpůrnou knihovnu PDK (Platform Developer's Kit).

PDK obsahuje tři funkční vrstvy: PSL (Platform Support Library), PAL (Platform Abstraction Layer) and DSM (Data Stream Manager). Tyto vrstvy podporují různé oblasti vývoje aplikací, přesto však jsou vzájemně jednotné a společně podporují vytváření autonomních i integrovaných Handel-C aplikací.

Využívání těchto platforem zvyšuje efektivnost především z toho důvodu, že vývojář, který vytváří aplikaci, nemusí znát všechny detaily hardwaru, se kterým pracuje. Nejnižší

úrovně obsluhy jednotlivých částí hardwaru desky RC jsou totiž již napsány v některé z platforem a stačí je tedy pouze využívat.

V mém návrhu jsem pracoval především s platformu PSL. Tato by se dala charakterizovat jako balík ovladačů k hardwaru, tak jak to známe z PC, což znamená, že poskytuje podporu komunikace s konkrétním hardwarem na desce RC. Toto její specifikum umožňuje uživateli velké pohodlí při práci s daným hardwarem. Nevýhodou je však její malá univerzálnost a především nemožnost simulace na PC, což naopak podporuje vrstva PAL.



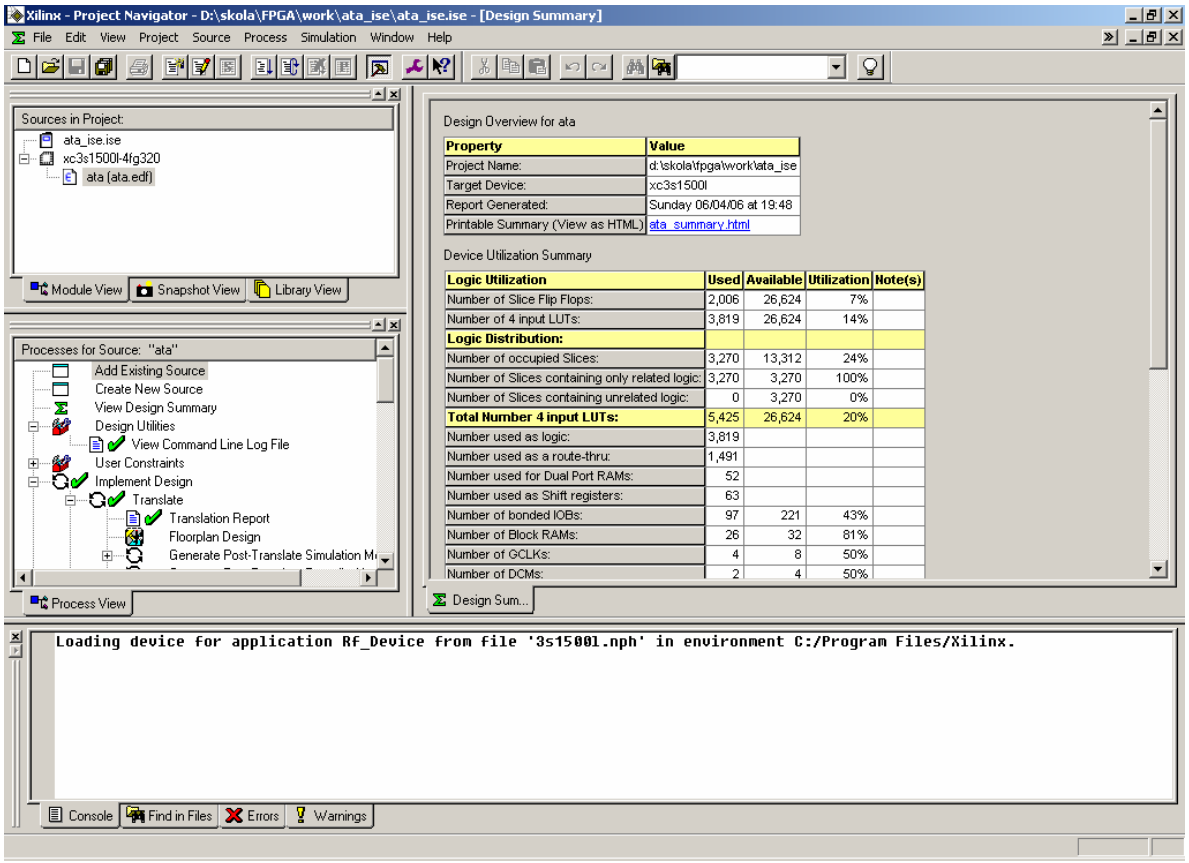
Obr. 6 : Ukázka prostředí DK 4.0

2.2. Xilinx ISE 7.1

Prostředí Xilinx ISE 7.1 bylo využíváno především k vytváření souborů pro konfiguraci FPGA čipu tzv. bit streamů. Tyto soubory jsou vytvářeny automaticky při překladu v prostředí DK od společnosti Celoxica. DK totiž vytvoří překladem mapu hardwaru tzv. EDIF, a z ní je pak dále generován pomocí backandu ISE konfigurační soubor čipu. Toto se děje bez nutnosti spouštění ISE jako samostatného programu. Program však musí být na daném počítači nainstalován.

Pokud by uživatel chtěl využívat všechny funkce ISE, je třeba nejprve přeložit zdrojový kód na soubor typu VHDL (pomocí DK Celoxica) a ten pak samostatně otevřít. Prostředí ISE pak poskytuje velkou škálu podrobných informací o daném návrhu. Umožňuje definování

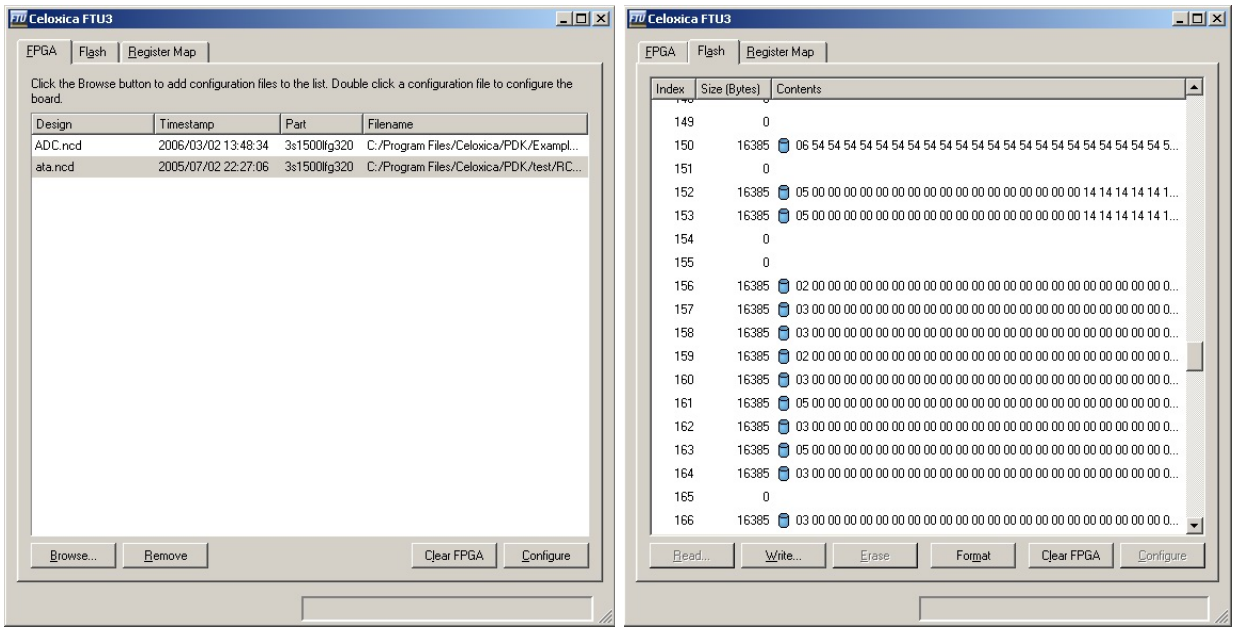
parametrů týkajících se rozmisťování hardwaru na čipu, mapování a podobně. Náhled prostředí je na *Obr. 7 : Prostředí Xilinx ISE 7.1*.



Obr. 7 : Prostředí Xilinx ISE 7.1

2.3. FTU3

FTU3 (File Transfer Utility) je samostatná počítačová aplikace, která pomocí rozhraní USB umožňuje komunikaci s kartou RC10. Při mé práci jsme tuto aplikaci vypoužíval především ke konfiguraci FPGA vytvořenými návrhy. Toto prostředí také umožňuje práci s pamětí flash umístěné na desce, čehož jsem využil především pro kontrolu při vytváření částí hardwaru, který právě s pamětí flash pracuje.



Obr. 8 : Aplikace FTU3. Vlevo ukázka konfigurace FPGA, vpravo práce s pamětí Flash

2.4. Handel C

Pro návrh veškerého hardwaru analyzátoru jsem použil programovací jazyk Handel C, který z velké části využívá základ konvenčního jazyka C. Je však doplněn o možnost programování v paralelních blocích. To znamená, že lze psát kód programu tak, aby se některé jeho části vykonávaly současně v jeden hodinový cyklus. Tím je dosaženo vysoké efektivnosti a hlavně rychlosti takto psaných procedur.

Kód lze samozřejmě psát i sekvenčně, tak je to ostatně i implicitně nastaveno. Chceme-li však dosáhnout rychlosti, kterou nám paralelní mód poskytuje, měli bychom ho maximálně využívat. V tom také spočívá velká výhoda FPGA oproti obyčejným mikročipům, kde se programy dají psát pouze sekvenčně a tedy i méně efektivně.

3. Návrh a realizace logického analyzátoru

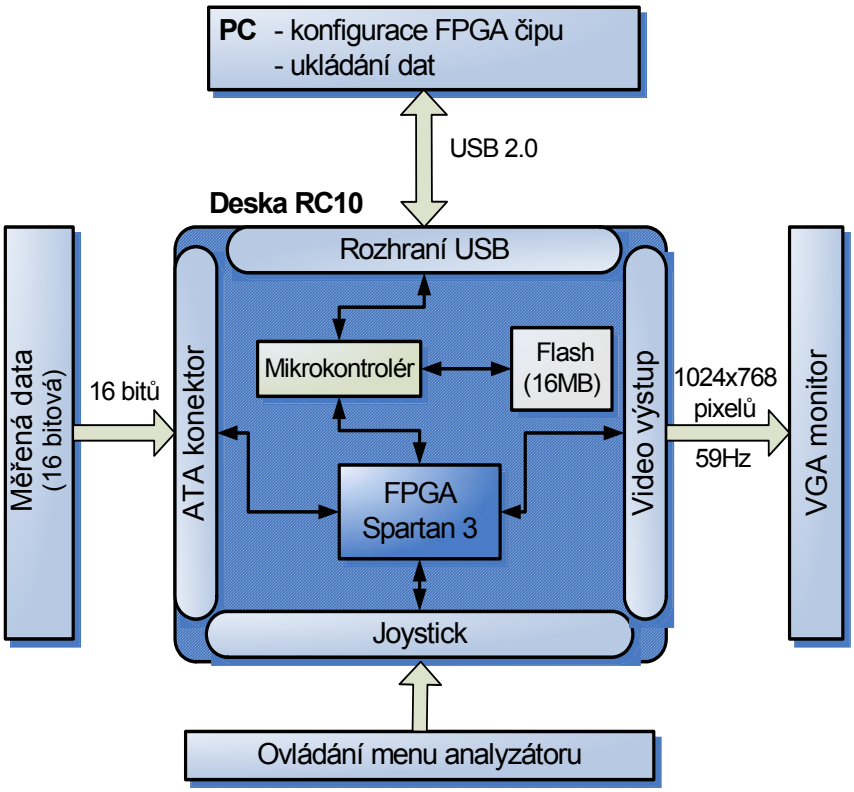
3.1. Návrh programu

Při vytváření mého návrhu jsem vycházel z již existujícího vzorového příkladu adc.hcc, který je součástí sady ukázkových příkladů pro RC10. Program je demonstrací čtení a následného zobrazování signálů přivedených na ADC převodníky. Tento návrh nám však sloužil spíše jako náhled do principů programování FPGA v jazyce Handel-C. Nicméně ve výsledné verzi programu byl opravdu použit princip pro obsluhu zobrazování na monitor, který byl rozšířen a upraven tak, aby vyhovoval nárokům na zadaný návrh. Detailním popisem zobrazování se zabývá kapitola *3.2.4 Zobrazování na monitor*.

Vytvořený návrh je složen z několika částí, přičemž každý blok vykonává určitou úlohu v programu, avšak části jako takové jsou na sobě takřka nezávislé. Takovýto princip byl užit především z důvodu dosažení co nejvyšší universálnosti a také rychlosti návrhu. Jednotlivé části jsou totiž vykonávány současně, a tak je využito velké výhody FPGA čipů oproti klasickým mikročipům. Hlavní výhodou je právě paralelní vykonávání velkého počtu operací během jednoho hodinového cyklu. Proto se také při realizaci povedlo dosáhnout vzorkovací rychlosti měřených signálů 64MHz, byť zadání úlohy znělo pouze na 48MHz.

3.2. Popis realizovaných funkčních částí analyzátoru

Program realizující logický analyzátor je možné rozdělit do několika základních částí. Jednak to jsou bloky starající se o sběr měřených dat a jejich zpracování, dále ovládání, jednotlivé funkce analyzátoru a v neposlední řadě blok zajišťující vizuální výstup na monitor.



Obr. 9 : Schéma návrhu logického analyzátoru

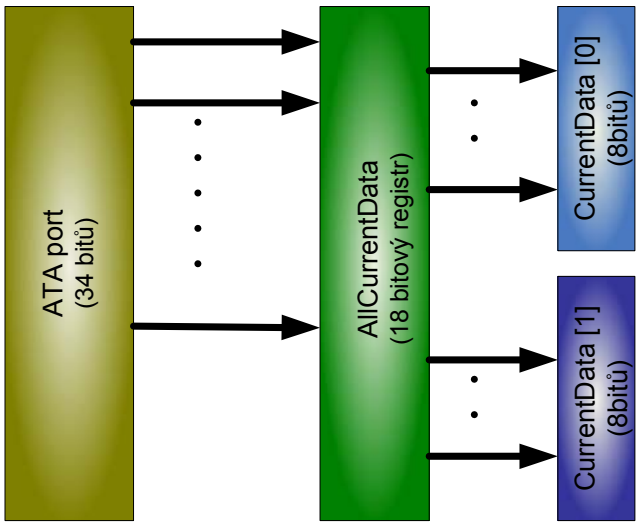
3.2.1. Čtení měřených dat

Sběr měřených dat pro logický analyzátor probíhá z 16 pinů ATA sběrnice tak, jak je uvedeno v kapitole *1.2.2. Využité části RC10*. A i když menu přístroje umožňuje deset hodnot pro nastavení časové základny, údaje ze všech vstupů jsou zde odečítány současně každý

hodinový cyklus, což odpovídá rychlosti 64MHz. O tom, jaká je nastavená hodnota časové základy, tedy jakou rychlostí mají být data dále zpracovávána, je rozhodováno až v části hardwaru nazvané *Zpracování měřených dat*, kapitola *3.2.2.*

Odběr ze sběrnice je realizován pomocí funkce *RC10ExpansionReadMask()*, která vrací 34 bitovou hodnotu odpovídající aktuálnímu stavu sběrnice ATA. Důležitým parametrem funkce je, že při přiložení signálu na sběrnici v jeden hodinový cyklus, je hodnota sběrnice funkcí načtena až v cyklu následujícím. Načítané hodnoty mají tedy jednocyklové zpoždění. Tato aplikace však při hodinové frekvenci 64MHz toto zpoždění plně respektuje.

Načítané hodnoty z portu jsou tedy 34bitové, využíváno je ale pouze osmnáct dolních bitů, které jsou ukládány do 18bitového registru *AllCurrentData*. A i z těch je použito pouze horních šestnáct bitů z toho důvodu, že nultý a první bit jsou pro čtení nevhodné (viz kapitola *1.2.2.*). Vyšších 16 bitů je dále rozděleno do dvou osmibitových registrů, které jsou následně používány jako výchozí pro další zpracování v logickém analyzátoru. Data jsou dělena především proto, že zde byla snaha o dosažení co možná nejvyšší univerzálnosti celého přístroje a to tím směrem, aby zařízení mohlo pracovat buď se dvěma osmibitovými kanály na sobě nezávislými, a nebo s jedním šestnáctibitovým. Detailněji se tímto zabývá následující kapitola *3.2.2.* nazvaná *Zpracování měřených dat*.



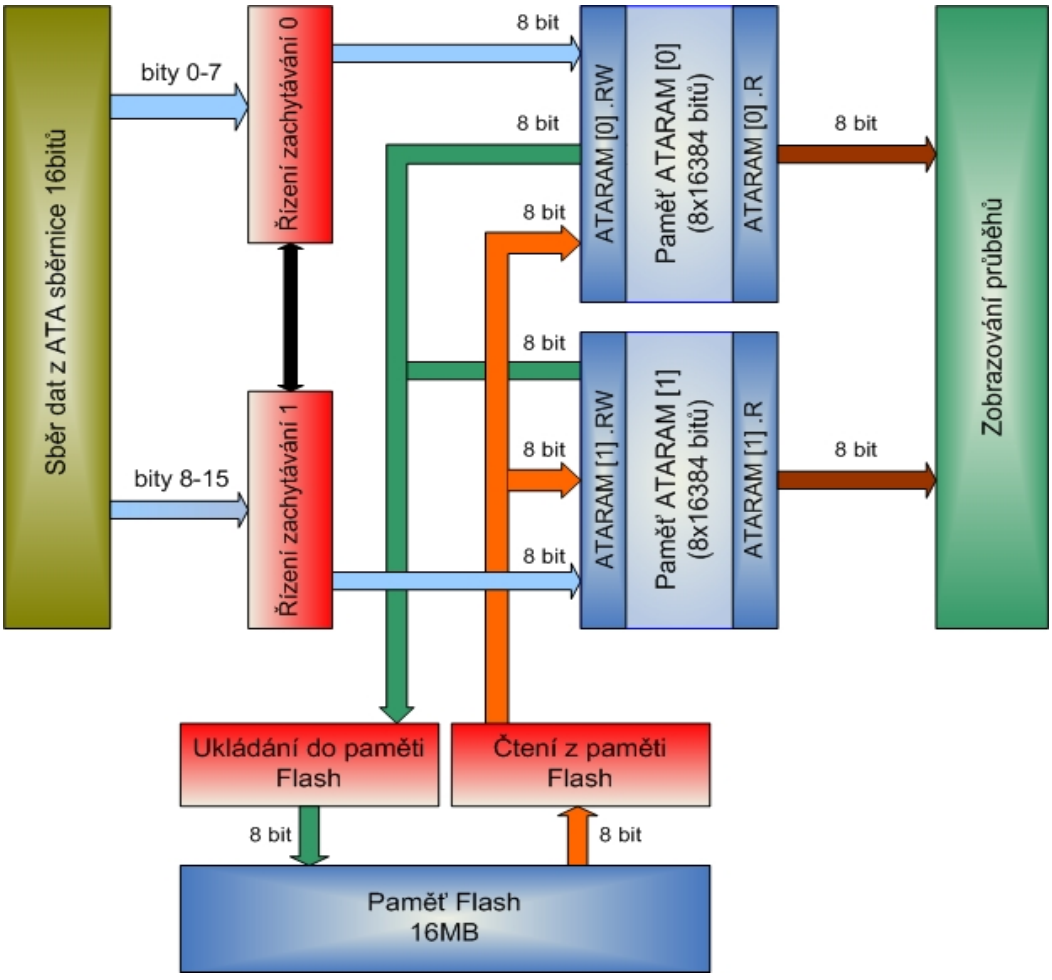
Obr. 10 : Sběr dat z ATA portu

3.2.2. Zpracování měřených dat

Základní princip zpracování měřených dat je takový, že po načtení aktuálních hodnot rozhraní ATA do registrů *CurrentData[0]* a *CurrentData[1]* jsou tyto hodnoty testovány v kapitole *4.2.3. Řízení zachytávání*, kde je rozhodováno, zda jsou platná. Pokud pak splňují uživatelem nastavené podmínky, data se začínají ukládat do dvou pamětí RAM

D0

nazvaných *ATARAM[0]* a *ATARAM[1]*, přičemž každá z pamětí slouží pro jeden osmibitový kanál (viz *Obr. 11: Princip zpracování měřených dat*). Paměti jsou dvouportové, každá o velikosti 8x16384 bitů. Zapsaná data jsou následně čtena v nekonečném cyklu a zobrazována na výstup. Ke čtení je využíván port *ATARAM[X].R*, který slouží jako zdroj dat právě pro zobrazování. Druhý port *ATARAM[X].RW* je určen jak pro čtení tak i pro zápis. Zápis je prováděn jednak z již výše zmíněné části *Řízení zachytávání 3.2.3.*, a dále pak při načítání uložených dat z paměti flash, kdy dochází ke kopírování uložených dat z paměti *flash* do paměti *ATARAM* tak, aby mohla být opět zobrazena. Čtení z tohoto portu je pak použito při ukládání hodnot do paměti flash. Oba ty tyto principy jsou detailně popsány v kapitolách *3.2.6. Ukládání naměřených dat do paměti flash* a *3.2.7. Čtení a znovuzobrazování uložených dat z paměti flash.*



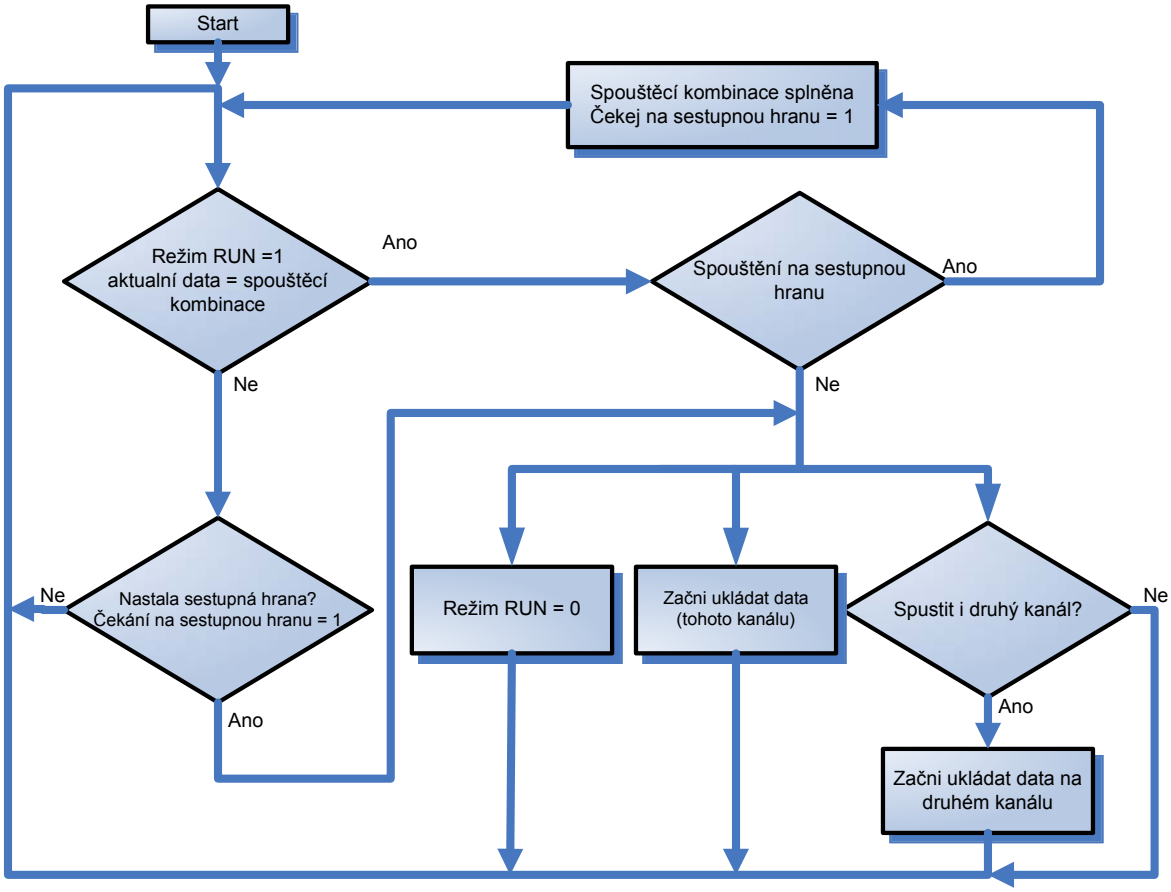
Obr. 12 : Princip zpracování měřených dat

3.2.3. Řízení zachytávání

Tato část hardwaru slouží k rozhodování, zda mají být aktuální data na sběrnici ATA načítána do paměti RAM či nikoliv. Vstupním argumentem jsou obsahy registrů *CurrentData[0]* a *CurrentData[1]*. Ty jsou každý hodinový cyklus přepisovány novými údaji ze sběrnice ATA. Proto je velmi důležité, aby i rozhodování, zda data odpovídají nastavené spouštěcí kombinaci, po které se má spustit ukládání do paměti ATARAM, bylo velmi rychlé, tj. o délce jednoho hodinového cyklu. Testování platnosti se tedy děje každý hodinový cyklus nezávisle na aktuální hodnotě časové základny. Tím, že se hodnoty na sběrnici testují maximální možnou rychlostí, je dosaženo velmi vysoké přesnosti pro rozhodování, zda jsou data již platná a mají být načítána do pamětí ATARAM a to při vysoké hodnotě časové základy. Pokud by se totiž mělo testovat spouštění stejnou rychlostí jako je aktuální hodnota časové základny, a ta by byla vysoká, mohlo by docházet k nezaznamenání rychlých přechodových dějů například zákmitů. A právě i pro takovéto měření chyb je analyzátor určen.

Spouštěcí kombinace je složena ze tří částí. Zaprvé zda je kanál analyzátoru ve spuštěném stavu (stavu *RUN),* zda je hodnota registru *CurrentData* totožná s nastavenou spouštěcí kombinací (kombinaci odpovídá registr *StartCombination*) a také zda se mají data ukládat s nástupnou či sestupnou hranou signálu. Unikátní podmínkou spouštění je tzv. spouštění od druhého kanálu. Což představuje možnost že jsou oba kanály spouštěny současně po splnění podmínek na jednom z nich.

Na níže uvedeném obrázku *(Obr. 9)* je znázorněna logika realizující řízení zachytávání. Rozhodování se děje pro každý osmibitový kanál zvlášť s možností spouštět jeden kanál na základě druhého.



Obr. 13 : Logika rozhodování zachytávání dat (jeden 8bitový kanál)

Ukázka kódu pro realizaci hardwaru:

Toto je ukázka kódu, který realizuje hardware pro výše vysvětlovanou logiku rozhodování zachytávání dat.

```
If ((CurrentData[k]==StartCombination[k]) && Store[k] )
   {
        if (!Edge[k])
        {
             par
             {
                  Triggered[k] = 1;
                  Store[k]=0;
                  if(StoreAll[k]==1)
                       Triggered[!k]=1;
             }
        }
        else
        {
             TriggeredEdge[k] = 1;
        }
   }
   else
   {
```

```
     if((TriggeredEdge[k])&&(CurrentData[k]!=StartCombination[k]))
          {
          par
          {
               Triggered[k] = 1;
               TriggeredEdge[k] = 0;
               Store[k]=0;
               if(StoreAll[k]==1)
                    Triggered[!k]=1;
          }
     }
     else
        delay;
  }
  else
     delay;
```

3.2.4. Zobrazování na monitor

Vizuální výstup je uskutečňován na monitor standartu VGA. Velikost rozlišení je 1024x768 bodů při obnovovací frekvenci 59Hz (viz kapitola *1.2.2 Využité části RC10*). Při návrhu hardwaru realizujícího výstup na monitor jsem vycházel z již existujícího příkladu, jak je uvedeno již v úvodu práce. Tento příklad uskutečňoval neustálé zobrazování úrovní dvou ADC vstupů karty RC10. Tento princip byl upraven a doplněn tak, aby vyhovoval zadání naší práce.

Hlavní změnou je, že zobrazované průběhy již nejsou dva ADC vstupy ale šestnáct vstupů z rozhranní ATA. Princip ukládání měřeného signálu do několika pamětí BlockRam a následné nekonečné čtení této paměti pro zobrazování byl zjednodušen, ale v jádru zůstal zachován. Navržený princip je zobrazen na *Obr.11: Princip zpracování měřených dat.*

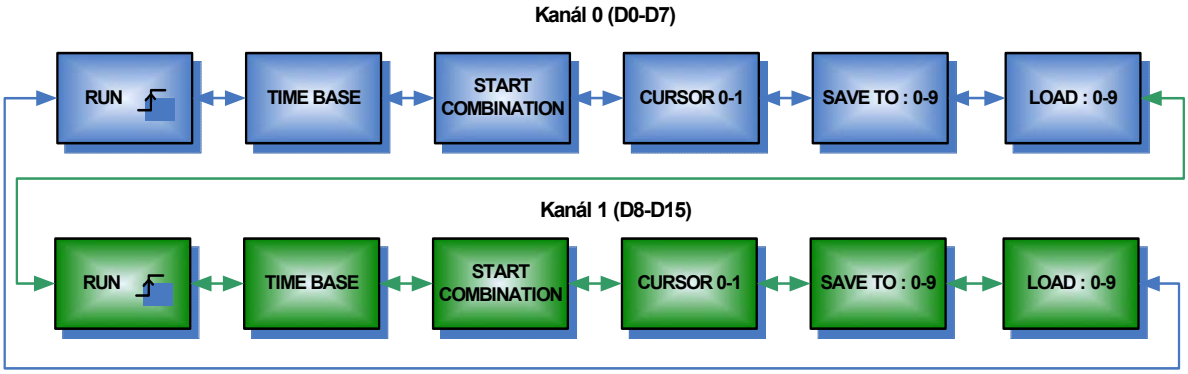
Zpracování výstupu na monitor je značně usnadněno tím, že se v programu není třeba zabývat rozklady a synchronizací signálů pro monitor na základní úrovni. K zobrazování je totiž použita jedna z funkcí PAL knihovny, kterou je práce značně usnadněna. Jedná se o funkci *PalVideoOutWrite ()*, jejímž vstupním argumentem je pouze barva jediného pixelu. Tento pixel je pak při následném hodinovém cyklus vykreslen na pozici, na které se aktuálně nachází zobrazovací paprsek monitoru. To však znamená, že je třeba zajistit, aby byla známa barevná hodnota každého pixelu na obrazovce. Pokud by měli ale všechny tyto informace mít uloženy v paměti přístroje, kladlo by to při rozlišení 1024x768 nárok na 768kbitů paměťového prostoru, což je v tomto případě nemožné. Proto je pro většinu zobrazovaných prvků využíváno funkcí, které podle aktuální pozice na obrazovce rozhodují, zda se daná informace má zobrazit či nikoliv.

K jednoduché synchronizaci aktuální pozice zobrazovacího paprsku na obrazovce je opět využíváno funkcí knihovny PAL a to *ScanX* a *ScanY*.

Pro vykreslování menu přístroje je použito obdobného principu jako při vykreslování průběhů signálů. Informace o vzhledu menu jsou uloženy v pamětech typu RAM. Paměti jsou o velikosti 18x1024 bitů a obsahují údaje o jednotlivých pixelech menu. Každému řádku menu přísluší jedna paměť této velikosti. Tento princip však není zcela mým dílem. První koncept řešení jednořádkového menu s alfanumerickými znaky realizovaného pomocí pamětí RAM, navrhl vedoucí naší bakalářské práce pan Ing. Jiří Kadlec CSc. Já jsem tento systém pak dále upravoval a rozšiřoval tak, aby vyhovoval požadavkům mé aplikace.

3.2.5. Obsluha analyzátoru, menu

Obsluha analyzátoru se provádí pomocí menu, které je umístěno v dolní části obrazovky. Menu přístroje je dvouřádkové, přičemž každý z řádků přísluší jednomu osmibitovému kanálu. Horní řádek menu odpovídá hornímu osmibitovému kanálu a dolní řádek zase dolnímu osmibitovému kanálu. Menu je i barevně rozlišeno, barva označujícího políčka odpovídá barvě kurzorů daného kanálu. V horním řádku je tedy modrý a v dolním zelený (viz *Obr. 14: Princip pohybu po menu přístroje*).



Obr. 14 : Princip pohybu po menu přístroje

Pohyb po menu je prováděn pomocí joystiku umístěného na kartě RC10. Pomocí naklánění joystiku doleva a doprava se pohybujeme po jednotlivých položkách menu, přičemž změna aktuální položky je prováděna pohybem nahoru dolů. Potvrzení dané volby se provádí stisknutím joysticku, které bylo pro kontrolu doplněno o pípnutí piezo reproduktoru na kartě RC10.

Jednotlivých položek je v menu celkem šest pro každý kanál a jak je patrné z fotografie obrazovky níže (*Obr. 16: Obrazovka logického analyzátoru*), máme k dispozici ještě další dvě pozice pro rozšíření. Nyní bych rád rozeberal jednotlivé položky menu tak, jak jdou chronologicky za sebou.

RUN (spouštění)

První položkou v menu je RUN. Jedná se o položku pro spouštění měření. Tato položka má tři podpoložky, které lze zvolit. Jednak je to spouštění na nástupnou hranu, spouštění na sestupnou hranu a také spouštění od druhého kanálu (EXT) viz *Obr. 14: Ukázka obsahu jednotlivých položek menu* . Spouštění od druhého kanálu odpovídá tomu, že přístroj se chová jako jeden šestnáctibitový kanál. Vzorky ze všech vstupů jsou načítány současně, a to po splnění podmínky spouštění na kanálu, který není v režimu EXT.

Změny těchto režimů se dějí pohybem joystiku nahoru a dolů, přičemž pro potvrzení aktuální nabídky je třeba joystik stisknout. Po stisknutí přechází přístroj okamžitě do zvoleného režimu. To je signalizováno zeleným čtverečkem vpravo od nápisu RUN. Pokud měření již v pořádku proběhlo, přístroj setrvává v klidu, což je také opět signalizováno, tentokrát zčervenáním čtverečku.

TIME BASE (časová základna)

Druhou v pořadí je položka pro nastavení časové základny přístroje. Ta obsahuje deset volitelných hodnot. Hodnoty jsou 0,5; 1; 2; 5; 10; 20; 50; 100; 200 a 500µs/div. Pohyb mezi těmito hodnotami je opět realizován pohybem joystiku nahoru a dolů. U této položky není třeba potvrzování, stačí pouze ponechat požadovanou hodnotu jako aktuální.

START COMBINATION (spouštěcí kombinace)

Spouštění na určitou kombinaci vstupů slouží především k zachycení požadovaného okamžiku průběhu signálu. Po splnění této kombinace na vstupech začíná být signál vzorkován. Proto umožňuje tato položka nastavení hodnot od 00h do FFh. Hodnoty jsou v hexadecimální soustavě, což bylo voleno pouze z důvodu úspory místa v menu.

Změna spouštěcí hodnoty je opět volitelná pomocí pohybu joystiku nahoru a dolů bez nutnosti potvrzení.

CURSOR (kurzor)

Každý kanál má k dispozici dva kurzory, kterými se lze pohybovat po celé délce naměřených dat. Kurzory jsou označeny indexy 0 a 1. K přepínání mezi kurzory je použit stisk joystiku. Pohyb kurzoru po záznamu je realizován naklápěním joystiku nahoru a dolů.

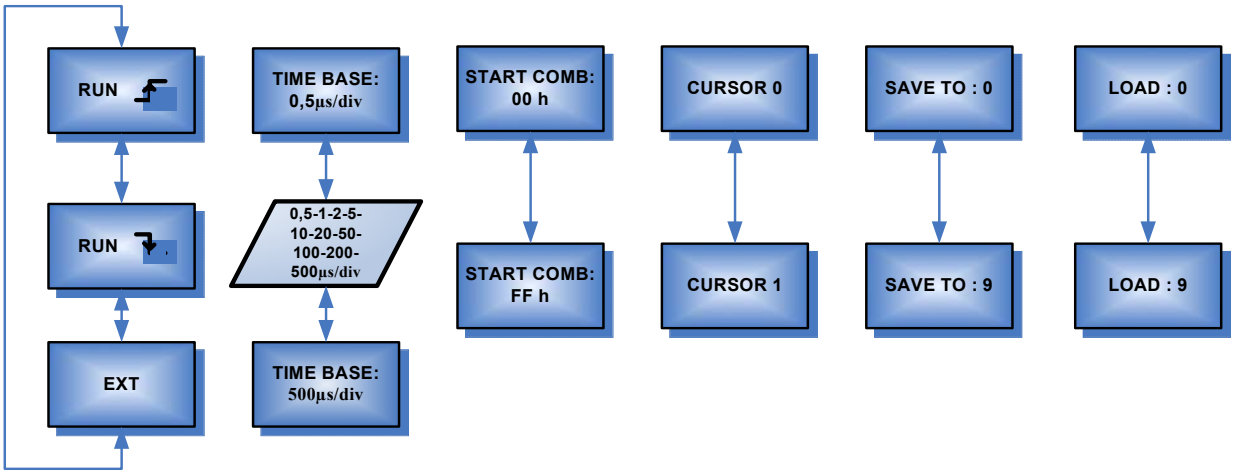
Kurzory slouží především k odměřování časových údajů jednotlivých průběhů. Časové údaje o aktuální pozici daného kurzoru jsou totiž neustále zobrazovány ve zvýrazněném pruhu pod každým osmibitovým kanálem (viz *Obr. 15: Obrazovka logického analyzátoru*). První zobrazovanou hodnotou je aktuální hodnota průběhu signálu na pozici, kde je umístěn kurzor. Následuje údaj o časové vzdálenosti kurzoru od počátku a údaj vyznačený deltou udávající hodnotu rozdílu časů obou kurzorů. Časové údaje jsou vypisovány v µs, údaj o hodnotě průběhu je přepočítáván a zobrazován v hexadecimální soustavě.

SAVE TO (ukládání)

Tato položka slouží k ukládání aktuálně naměřených průběhů analyzátoru do paměti flash. Pro ukládání je k dispozici deset položek indexovaných od čísla 0 do čísla 9. Pro uložení dat stačí nastavit požadovanou položku, kam mají být data umístěna a stiskem joystiku je tato operce potvrzena. Data jsou pak okamžitě zkopírována do paměti flash. Detailním popisem této procedury se zabývá kapitola *3.2.6 Ukládání naměřených dat do paměti flash*.

LOAD (načtení)

Položka LOAD je inverzní k položce SAVE. Načítá tedy uložené průběhy v paměti flash a zobrazuje je jako aktuálně naměřené. Obdobně jako u SAVE stačí zvolit požadovaný záznam s indexem od 0 do 9 a potvrdit stiskem. V případě, že jsou data platná, dochází k jejich načtení a zobrazení stejně tak, jako tomu bylo při měření.



Obr. 15 : Ukázka obsahu jednotlivých položek menu

Obr. 16 : Obrazovka logického analyzátoru

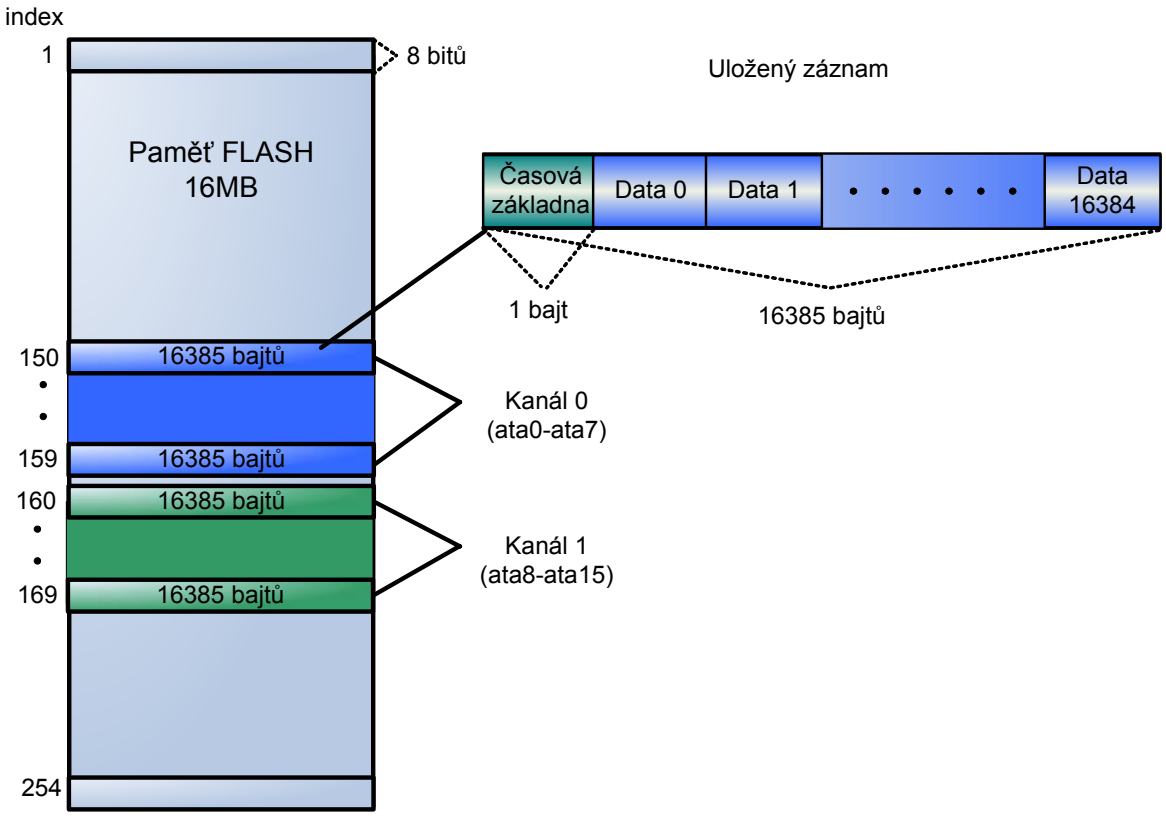
3.2.6. Ukládání naměřených dat do paměti flash

Naměřené průběhy je možno ukládat do paměti flash. Pro každý osmibitový kanál je k dispozici deset pozic pro uložení naměřených průběhů. Celkem se tedy ukládá do dvaceti pozic v paměti. Průběhy jsou ukládány v celé naměřené délce, tj. 16384 bitů na jednotlivé místa paměti flash. Paměť flash je od výrobce rozdělena do 254 položek libovolné velikosti. V návrhu přístroje jsou využity však jen některé indexy. Od pozice 150 do 159 jsou ukládány průběhy kanálu 0 (tj. *ata0-ata7*) a od pozice 160 do 169 kanálu 1 (*ata8-ata15*).

Ukládání se děje postupným načítáním hodnot z paměti *ATARAM* a následným ukládáním do paměti flash. Pozice v paměti flash, na kterou se má daný průběh uložit, je nejdříve smazána pomocí funkce *RC10FlashErase ()*. Na první pozici v daném záznamu je uložena aktuální hodnota časové základny. Následně jsou čteny osmibitové hodnoty naměřených dat z paměti *ATARAM* a ukládány do paměti flash. Ke čtení jsou využívány porty *ATARAM[X].RW* pamětí *ATARAM* (viz *Obr. 17 : Princip zpracování měřených dat*). Ukládání je realizováno pomocí funkcí *RC10FlashAppendBegin()* určujících délku a

začátek záznamu a funkce *RC10FlashAppend ()*. Schéma organizace paměti flash je znázorněna na *Obr. 18 : Schéma organizace uložených dat v paměti flash*.

Každý uložený záznam má délku 16385 bajtů (v přepočtu 16kB) a celková velikost paměti flash je 16MB (16384kB). Z toho vyplývá dostatečná paměťová rezerva pro možnost rozšíření ukládání na všech 254 pozic paměti a nebo při změně systému ukládání schopnost uložení až 1024 naměřených průběhů.



Obr. 19 : Schéma organizace uložených dat v paměti flash

3.2.7. Čtení a znovuzobrazování uložených dat z paměti flash

Naměřená data, která jsou uložena v paměti flash, je možno znovu načítat a zobrazovat na monitor analyzátoru, přičemž jsou zachovány všechny dostupné funkce analyzátoru tak, jako by se jednalo o právě pořízené měření.

Hardware, realizující čtení dat z paměti flash, je vytvořen tak, že po zvolení příslušného záznamu, který má být načten, je v menu analyzátoru nejprve načtena hodnota časové základny, pro kterou byl daný záznam pořízen, a dále pak jsou po bajtech čtena jednotlivá naměřená data. Ta jsou ukládána do paměti ATARAM. Ukládání se děje přes port *ATARAM[X].RW* (viz *Obr. 20: Princip zpracování měřených dat*). Hodnota časové základny

je nejprve ukládána a nyní čtena proto, aby bylo možno správně počítat časové pozice jednotlivých kurzorů a také případně zobrazení časů při přenosu dat na PC.

Systém načítání je do značné míry podobný samotnému měření dat s tím rozdílem, že zde je zdrojem dat paměť flash. Zároveň je možné díky univerzálnosti všech částí pracovat s daty tak, jako by byla aktuálně naměřena.

3.2.8. Odměřování časových hodnot kurzorů

Pod každým ze dvou 8bitových kanálů je zvýrazněný pruh, ve kterém jsou zobrazovány aktuální časové pozice kurzorů (viz *Obr. 16: Obrazovka logického analyzátoru*). Prvním údajem je časová vzdálenost aktuálního kurzoru od počátku a druhým pak rozdíl časových hodnot obou kurzorů kanálu.

Pro výpočet údaje času, o který je daný kurzor vzdálen od počátku, jsou vyžity údaje registrů udávající pozice kurzorů na obrazovce (tento údaj je v pixelech), hodnoty použité časové základny a také hodnota hodinového signálu. Zápis výpočtu je naznačen v rovnici *Rovnice 1: Výpočet vzdálenosti kurzoru od počátku*. Výsledný údaj je pak ještě násoben milionem, a tím je hodnota vypočítávána přímo v µs.

Tento vztah však není v hardwaru realizován přímo, protože by bylo velmi náročné pro systém vykonávat takto obtížné výpočty v jednom hodinovém taktu. Výpočet je tedy prováděn po částech a to pouze základními aritmetickými operacemi jako je sčítání a odčítání. Násobení je tedy nahrazeno opakovaným přičítáním a dělení naopak opakovaným odčítáním.

$$TimeOfCursor = \frac{PositionOfCursor * TimeBase}{ClockRate} * 1000000 \left[\mu s = \frac{pixels * -}{Hz} \right]$$

Rovnice 1: Výpočet vzdálenosti kurzoru od počátku

Výpočet rozdílu vzdáleností kurzorů je pak velmi podobný předchozímu výpočtu. Jediný rozdíl je v nahrazení hodnoty pozice kurzoru údajem o rozdílu hodnot obou kurzorů.

$$TimeOfDeltaCursor = \frac{|PositionOfCursor0 - PositionOfCursor1| * TimeBase}{ClockRate} * 1000000$$

$$\left[\mu s = \frac{|pixels - pixels| * -}{Hz} \right]$$

Rovnice 2: Výpočet rozdílů vzdáleností kurzorů

Výpis obou časových hodnot je pak realizován pomocí definované RAM paměti, která je, stejně jako paměti určené pro menu přístroje, neustále vykreslována na obrazovku. K přepisu těchto pamětí dochází, ale pouze při změně některé ze vstupních hodnot a to z toho důvodu, aby nedocházelo k blikání vykreslovaných hodnot.

Aby mohlo dojít k vypsání vypočtených hodnot na obrazovku, je ještě třeba jejich převod do desítkové soustavy a oddělení jednotlivých cifer celého čísla. Výpis se totiž provádí po jednotlivých cifrách tak, že z vyhrazené ROM paměti, která obsahuje předepsanou sadu fontů, se načte hodnota požadovaného fontu, která je následně zkopírována do vykreslované paměti RAM. Tento postup je dále opakován pro všechny cifry vypočteného čísla.

3.2.9. Odměřování hodnot kurzorů v o se y

Mimo údajů o časech kurzorů jsou ve vyznačených pruzích pod jednotlivými kanály vypisovány i informace udávající hodnotu signálu v místě, kde se nachází aktuální kurzor. Výpis údajů je pro jednoduchost udáván v hexadecimální soustavě a je taktéž jako hodnoty času vypisován pouze při změně. Toto jednak zamezuje blikání, ale také to snižuje počet přístupů k daným pamětím.

3.3. Přenos a zpracování naměřených dat

Možnost přenášet a zpracovávat měřená data na PC byla vytvořena především z toho důvodu, že navržený logický analyzátor se omezuje spíše na základnější funkce pro zpracovávání měřených dat a rozšířenou možnost zpracovávat naměřená data dále pomocí počítače jistě ocení každý uživatel. Zda zpracování bude speciálně vytvořeným softwarem, či některým z již existujících, zůstává na něm.

3.3.1. Přenos naměřených dat to PC

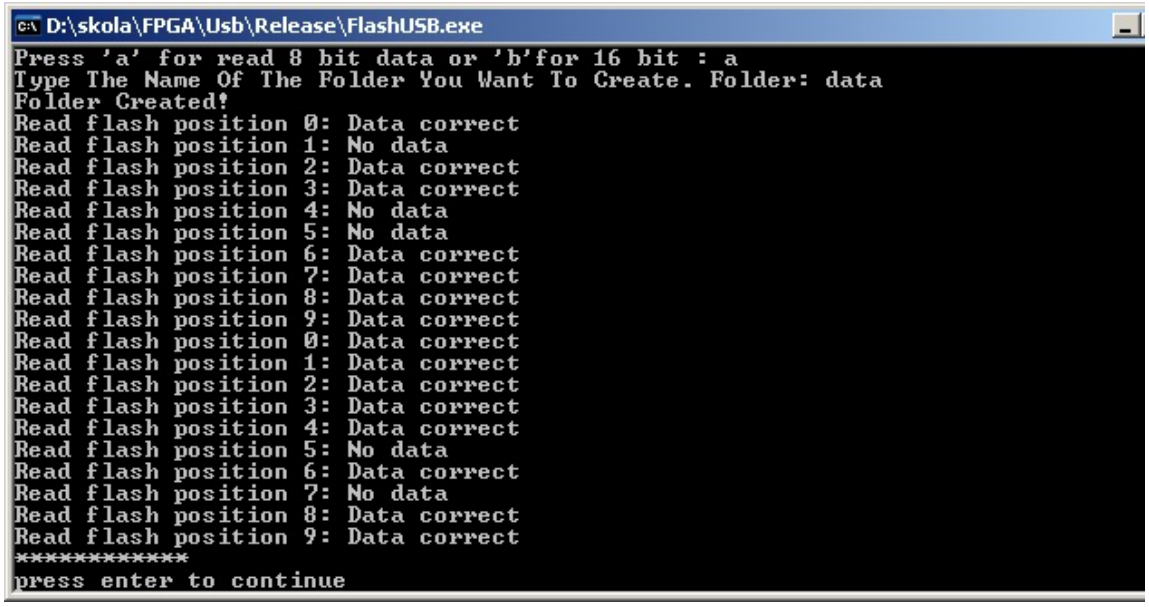
Pro přenos naměřených dat z paměti flash karty RC10 se využívá rozhraní USB připojené k desce a speciálně vytvořený program. Program je napsán v jazyku C++ a po spuštění na PC čte data z paměti flash a ukládá je do souborů typu txt na disk počítače. Tento program lze spouštět takřka kdykoliv bez předchozí inicializace. Program využívá pro komunikaci s deskou knihovnu *rc.lib* s hlavičkovým souborem *rc.h*, která je součástí softwaru *DK Design Suite* a je určena právě pro komunikaci s deskou.

Jelikož samotný logický analyzátor lze provozovat jako jeden 16bitový a nebo dva 8bitové kanály, napsaný software je pro to přizpůsoben. Umožňuje tedy čtení hodnot z paměti flash jako 8bitových a nebo 16bitových pro snadné další zpracování. Princip a systém ukládání dat je popisován v kapitole *3.2.6. Ukládání naměřených dat do paměti flash*.

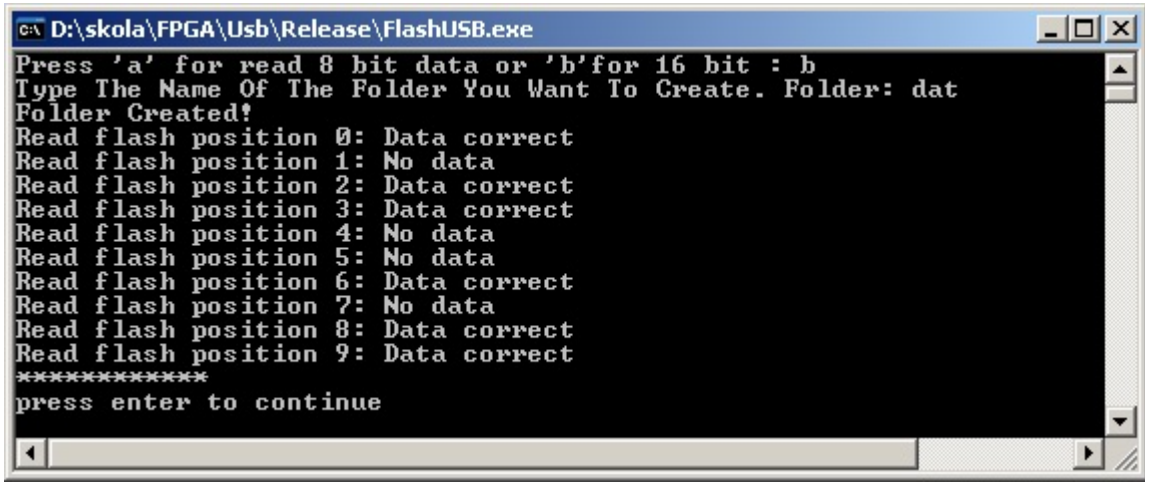
Data jsou uložena po jednotlivých vzorcích, přičemž na začátku každého záznamu je uložena i hodnota časové základny, pro kterou bylo dané měření pořízeno. Jednotlivé naměřené vzorky jsou v paměti reprezentovány jako číselné hodnoty, které jsou čteny v desítkové soustavě a programem převáděny do dvojkové. Ty jsou pak zapisovány

s mezerou mezi každým číslem do souboru typu txt. Na začátek každého řádku je pak ještě přidána aktuální hodnota času, která je uváděna v mikrosekundách a je programově vypočítávána z hodnoty časové základy použité při měření.

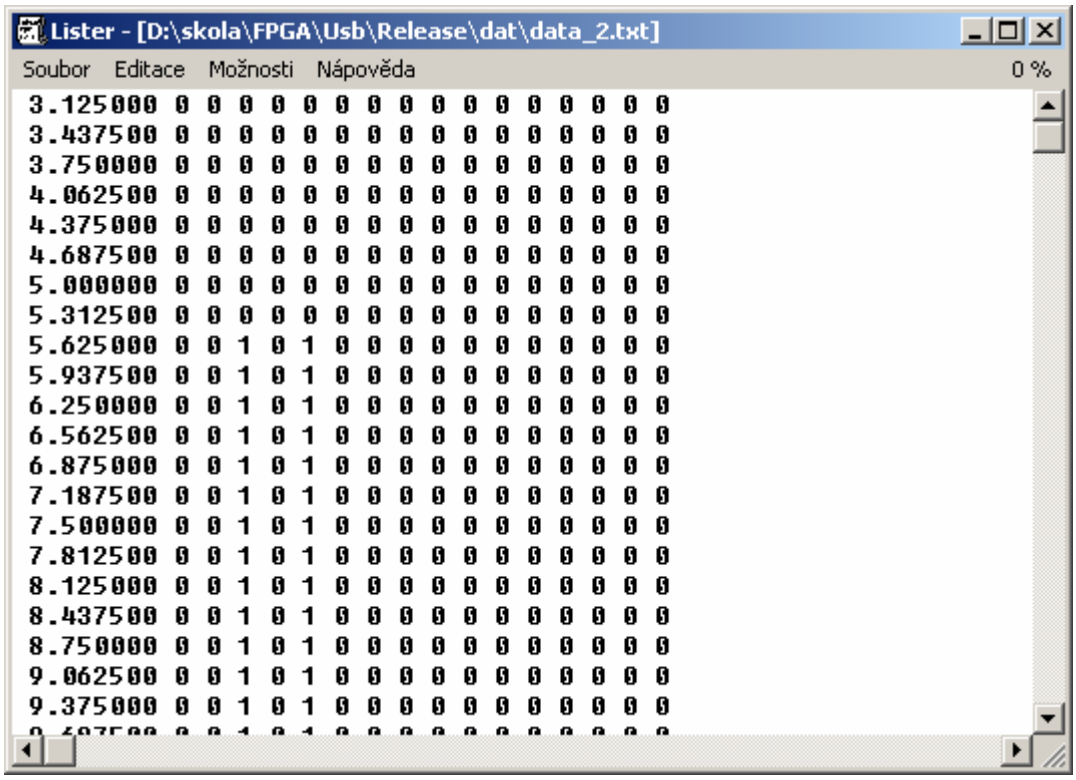
Po spuštění programu se tedy zadává volba, zda mají být data 8bitová či 16bitová a název adresáře, do kterého se mají uložit. Pokud adresář již existuje, dojde k přepsání všech souborů uvnitř novými. Pokud dosud neexistuje, je vytvořen a načtená data jsou do něj uložena. Průběh načítání je pak zobrazován na následujících řádcích tak, že u každé paměťové pozice dochází k výpisu, zda jsou data v pořádku načtena či zda daná pozice data neobsahuje. Toto je reprezentováno slovy *Data correct* nebo *No data*. Po dokončení čtení se okno po stisknutí klávesy enter zavírá. Ukázky čtení pro 8bitové a 16bitové hodnoty jsou na *Obr. 21* a *Obr. 22*.



Obr. 21 : Okno programu pro ukládání naměřených dat do PC. Čtení pro dva 8. bitové kanály



Obr. 22 : Okno programu pro ukládání naměřených dat do PC. Čtení pro jeden 16. bitový kanál.

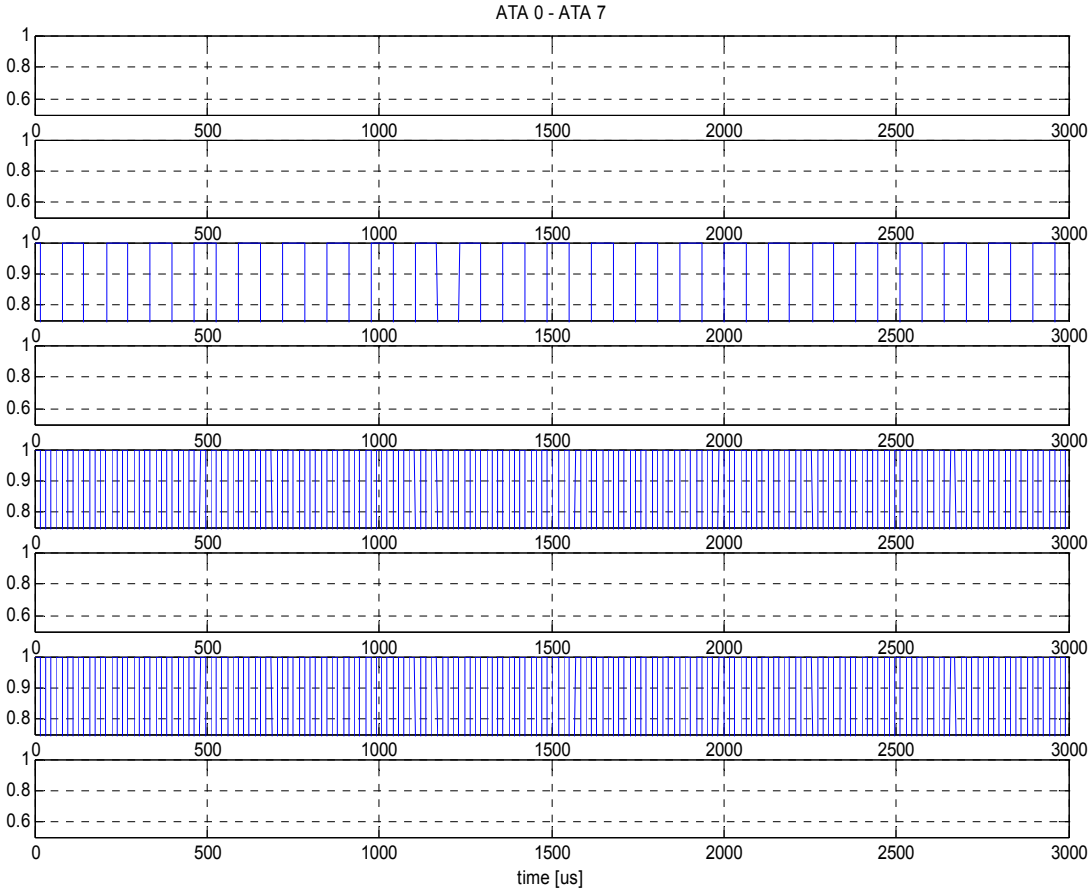


Obr. 23 : Ukázka uložených dat (pro 16 bitový kanál)

3.3.2. Zpracování uložených dat v PC

Jelikož formát ukládaných dat byl vytvářen se snahou o co nejvyšší univerzálnost, dají se údaje zpracovávat mnoha způsoby. Jedním z nich je určitě vlastní implementace softwaru, který by data zpracovával dle požadavků uživatele. Nicméně já jsem využil méně náročnou cestu, kterou je zpracování v již existujících programech. Mezi ně se může řadit i Matlab.

A právě pro program Matlab jsem vytvořil jednoduchý skript, který načítá uložená data ze souborů a zobrazuje jednotlivé průběhy. Skript je volán jako jednoduchá funkce, kde vstupním parametrem je název uloženého souboru, který chceme zobrazit. Program si pak načte soubor po jednotlivých sloupcích, přičemž každý sloupec je uložen jako samostatná matice o velikosti 16384x1 (náhled uloženého souboru viz *Obr. 24:: Ukázka uložených dat (pro 16bitový kanál*). Po načtení máme tedy jednu matici představující hodnotu času a dalších 8 nebo 16 hodnot průběhů. Pak už jsou volány jednoduché funkce *plot()* a *subpolot()* pro vykreslení obsahů matic. Náhled takto vytvořeného zobrazení je na *Obr. 24: Ukázka zobrazení uložených průběhů pomocí programu Matlab*.



Obr. 25 : Ukázka zobrazení uložených průběhů pomocí programu Matlab

4. Závěr

Vytvořený návrh analyzátoru, jak je v této práci prezentován, splňuje všechny hlavní body zadání. Některé části se dokonce povedlo vytvořit lépe a nad rámec požadavků. Tím největším úspěchem je dosažení vzorkovací rychlosti až 64MHz, což umožňuje měření logických signálů až do frekvence 32MHz. Návrh hardwaru přitom zůstal natolik efektivní, že jsem nebyl nucen používat rekonfiguraci čipu. Celý návrh je tedy tvořen pouze jedním obrazem čipu FPGA. Projekt je pak možno s menšími úpravami využívat i na čipech FPGA, které neumožňují rekonfiguraci jako například čipy Altera, jenž jsou levnější alternativou použitého Spartanu.

Realizovaný návrh je ukázkovým příkladem výhod programovatelných hradlových polí oproti standardním mikročipům. Mikroprocesor zpracovává pouze jednu programovou instrukci až několik hodinových taktů a není schopen vykonávat několik úkonů současně. Jeho rychlost je tím do značné míry omezena. Naopak u čipu FPGA, kde se programový kód převádí na hardware, je možno vykonávat několik instrukcí současně a to i v jednom hodinovém cyklu. Rychlost zpracování dat je tím mnohonásobně vyšší.

Délka pořizovaného záznamu je 16 384 vzorků. Maximální vzorkovací frekvence je 64MHz, což umožňuje měření logických signálů až do frekvence 32MHz. Délka pořizovaného záznamu je při maximální vzorkovací frekvenci 256µs. Navzorkované průběhy je možno uložit do 20 pozic ve vnitřní paměti flash přístroje.

Hodnota časové základny	Vzorkovací frekvence přístroje	Maximální frek. měřeného signálu	Délka pořízeného záznamu
0,5 µs/div	64 MHz	32 MHz	256 µs
1 µs/div	32 MHz	16 MHz	512 µs
2 µs/div	16 MHz	8 MHz	1024 µs
5 µs/div	6,4 MHz	3,2 Mhz	2,56 ms
10 µs/div	3,2 Mhz	1,6 Mhz	5,12 ms
20 µs/div	1,6 Mhz	800 kHz	10,24ms
50 µs/div	640 kHz	320 kHz	25,6 ms
100 µs/div	320 kHz	160 kHz	51,2 ms
200 µs/div	160 kHz	80 kHz	102,4 ms
500 µs/div	64 kHz	32 kHz	256 ms

Tabulka 1: Srovnání režimů logického analyzátoru

5. Použitá literatura

Ke své práci jsem nečerpal z žádné tištěné literatury, ale využíval jsem především elektronickou dokumentaci, která je součástí instalací softwaru DK 4.0. Dále pak internetových zdrojů, a to stránek výrobců jak karty RC10 www.celoxica.com tak výrobce FPGA čipů Xilinx www.xilinx.com.