VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

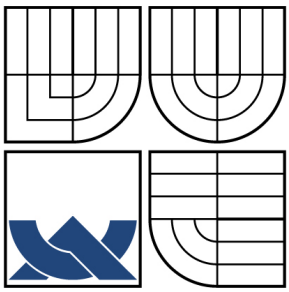
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

PROGRAMOVÁNÍ OBVODŮ PLD POMOCÍ MIKROPROCESORŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

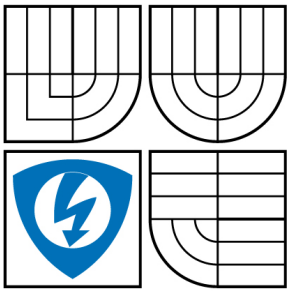
AUTOR PRÁCE MICHAL PETRILAK
AUTHOR

BRNO 2010



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

PROGRAMOVÁNÍ OBVODŮ PLD POMOCÍ MIKROPROCESORŮ

PROGRAMMING OF PLD DEVICES USING MICROPROCESSORS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

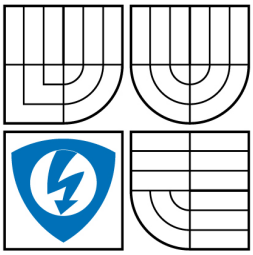
AUTOR PRÁCE — MICHAL PETRILAK

AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL KUBÍČEK, Ph.D.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student:	Michal Petrilak	***ID:***	106709
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Programování obvodů PLD pomocí mikroprocesorů

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s metodami programování obvodů typu CPLD a FPGA s využitím mikrokontrolérů. Vyberte metodu, která bude vyžadovat minimum přídavného hardwaru a softwaru ke standardnímu PC a umožní libovolnému uživateli plně kontrolovat konfiguraci obvodu PLD. Navrhněte schéma zapojení jednoduchého přípravku s obvodem FPGA, který bude využívat navrženou metodu konfigurace. Realizujte přípravek s obvodem FPGA a demonstrujte možnost rekonfigurace obvodu vámi vybranou metodou. V případě potřeby vytvořte jednoduché uživatelské prostředí pro řízení konfigurace pomocí PC.

DOPORUČENÁ LITERATURA:

[1] Xilinx In-System Programming Using an Embedded Microcontroller [online]. Xilinx application note 2009 [cit. 2009-05-15]. Dostupné na www: http://www.xilinx.com/support/documentation/application_notes/xapp058.pdf

Termín zadání:	8.2.2010	***Termín odevzdání:***	28.5.2010

Vedoucí práce: Ing. Michal Kubíček, Ph.D.

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá konfigurací obvodů FPGA firmy Xilinx pomocí mikrokontroleru, popisuje jednotlivé metody konfigurace a vybírá nejvhodnější pro realizaci. Dále je pozornost věnována vyběru mikrokontroleru a přídavného hardwaru. Závěrečná část je věnována tvorbě softwaru pro konfigurační mikrokontroler i obslužný softwar pro počítač. Navrhovaný programátor má sloužit jako levná náhrada komerčních programátorů od firmy Xilinx.

KLÍČOVÁ SLOVA

FPGA, CPLD, JTAG, mikrokontrolér, konfigurace PLD

ABSTRACT

This work is focused on configuring of Xiling's FPGA devices by embedded microcontroller. Several methods of configuration are described and the most suitable method is chosen. The attention is also paid to choose the microcontroller and auxiliary hardware components. The last part of the work describes the creation of the software for the microcontroller and for the PC. The suggested device should serve as an inexpensive substitute for commercial programmers from Xilinx.

KEYWORDS

FPGA, CPLD, JTAG, microcontroller, PLD configuration

PETRILAK, Michal *Programování obvodů PLD pomocí mikroprocesorů*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2010. 38 s. Vedoucí práce byl Ing. Michal Kubíček, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Programování obvodů PLD pomocí mikroprocesorů“ 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.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Tato práce se věnuje návrhu jednoduchého programátoru pro konfiguraci obvodů FPGA a CPLD firmy Xilinx, jehož účelem je levná náhrada za originální programátor.

Obvody FPGA a CPLD dnes tvoří velmi významnou platformu pro návrh číslicových systémů pro zpracování řečových a obrazových signálů v reálnem čase a také mají velké uplatnění ve vysokorychlostních komunikačních systémech. Umožňují velmi efektivní realizaci výkonných výpočetních systémů, které bylo dříve možné realizovat jen zakáznickými integrovanými obvody nebo spojením více dílčích obvodů na desce plošných spojů. Jejich snadná rekonfigurovatelnos značně přispívá k rychlosti vývoje moderních aplikací.

Práce popisuje základní koncepci obvodů FPGA/CPLD a jednotlivé metody nahrávání konfiguračních dat do těchto obvodů. Z jednotlivých metod je vybrána jedna, která se jeví jako nejvhodnější pro realizaci. Důležité při výběru metod je jednoduchost realizace, počet potřebných konfiguračních vodičů a univerzálnost.

Dále je věnován výběr mikrokontroleru vhodnému z hlediska ceny, dostupnosti a připojení mikrokontroleru k PC.

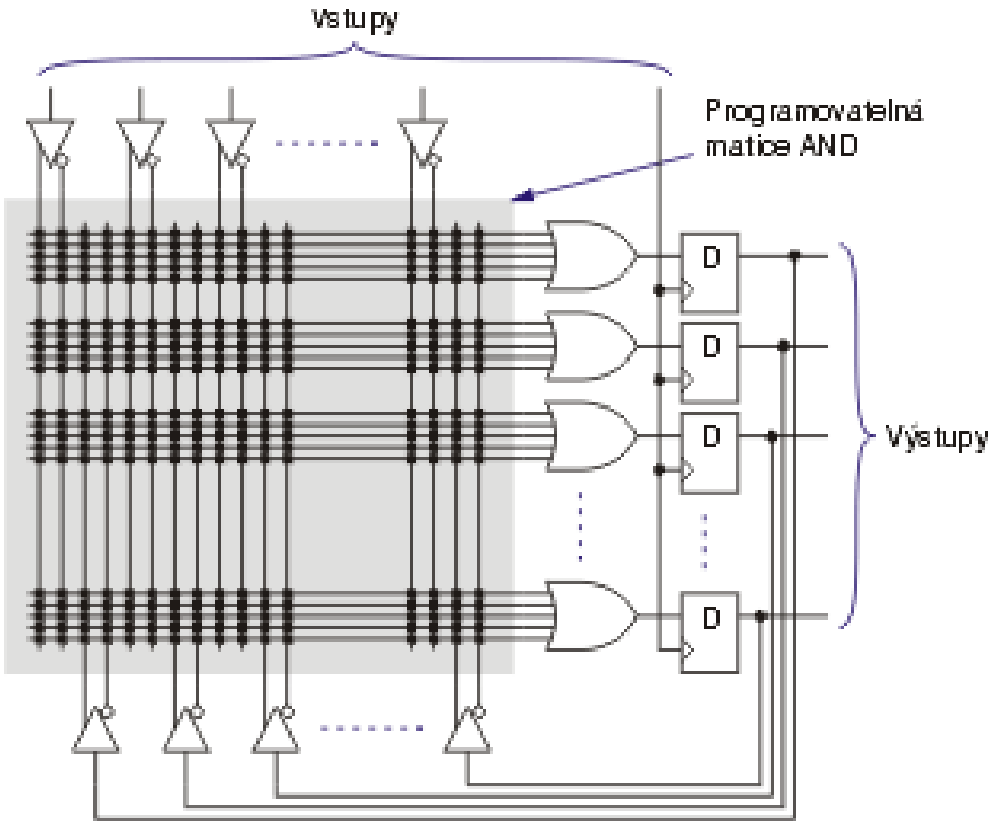
Obvody PLD mohou pracovat při různých napájecích napětích, proto bude programátor vybaven přizpůsobovacím obvodem, pro libovolné napájení cílového PLD.

Jelikož se obsah konfigurační paměti ve většině FPGA obvodů smaže se ztrátou napájecího napětí, je potřeba do FPGA nahrát konfiguraci po každém připojení napájecího napětí. Programátor proto bude mít datovou paměť typu FLASH, aby bylo možné cílové FPGA opakovaně konfigurovat bez nutnosti přenosu konfiguračních dat z PC.

Nedílnou součástí programátoru je software, a to jak část přímo v mikrokontroleru programátoru tak i část pro PC. Cílem je jednoduché a rychlé přeprogramování daného FPGA/CPLD obvodu. Konfigurační data jsou uloženy v souboru .bit, ze kterého je potřeba je načíst. Rámcovou předlohou je originální software od firmy Xilinx k vývojové desce s CPLD CoolRunner-II.

1 ŘEŠENÍ STUDENTSKÉ PRÁCE

1.1 Struktura CPLD a FPGA



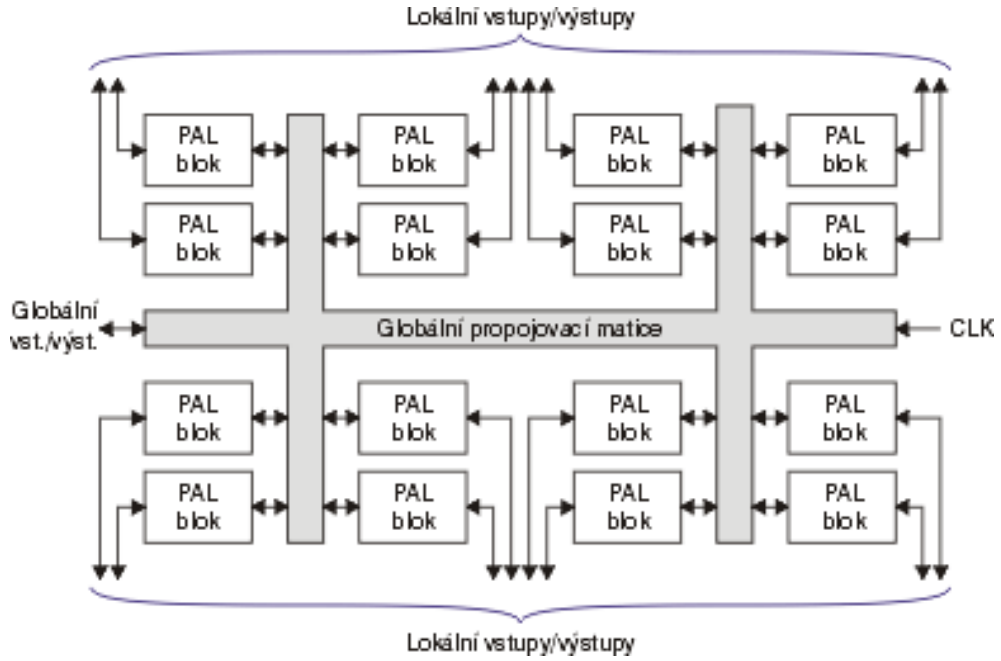
Obr. 1.1: Struktura PAL [1]

Programovatelné hradlové pole (většinou se používá zkrácený termín hradlové pole) nebo také PLD (zkr. programmable logic device) je číslicový integrovaný obvod s velmi obecnou strukturou hradel, registrů a spínačů. Do obvodu je nutné nahrát konfigurační soubor, který definuje chování jednotlivých částí a určí propojení těchto dílčích částí mezi sebou. V podstatě se jedná o obvod, kterému můžeme nastavit jeho vnitřní strukturu konfiguračním souborem. V malých výrobních sériích mohou nahradit obvody ASIC[1], neboť umožňují dosáhnout podobných výsledků při použití sériově vyráběných obvodů.

Obvody PLD se dělí do 3 skupin: SPLD, CPLD a FPGA.

Nejjednodušší a nejstarší obvody SPLD (simple programmable logic device) obsahují matici AND-OR hradel a klopné obvody typu D. Mají řádově deset až dvacet

[1] application-specific integrated circuit – zákaznické IO



Obr. 1.2: Struktura CPLD [1]

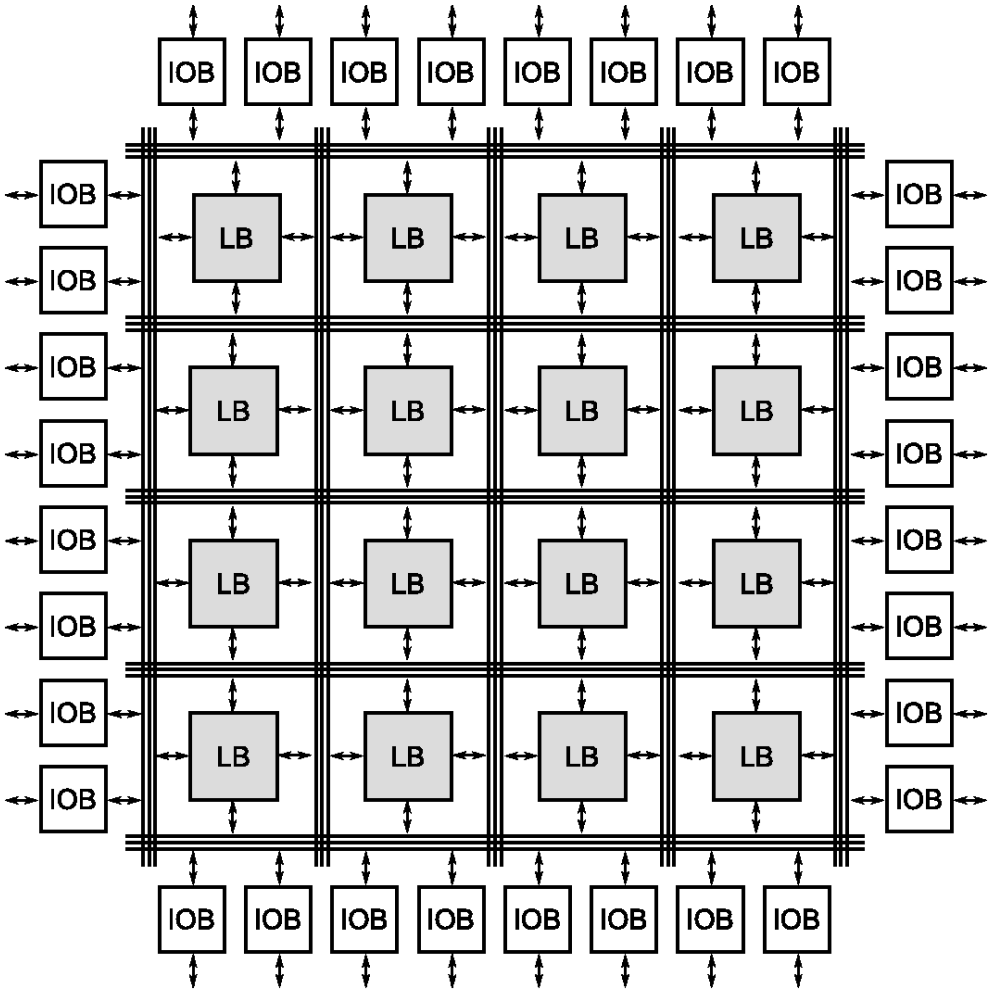
vstupů a kolem osmi výstupů. Nejčastěji se používají jako adresní dekodér na paralelní sběrnici. Konfigurace je uložena ve vnitřní paměti EEPROM. Dnes se již téměř nepoužívají. Základní struktura je zobrazena na obr. 1.1.

Klasické obvody SPLD mají velmi omezené prostředky, takže umožňují realizovat pouze jednodušší funkce. Proto výrobci začali sdružovat více těchto obvodů na jednom čipu spolu s nutnými prostředky pro propojení. Tyto obvody se většinou označují jako CPLD, což znamená Complex Programmable Logic Device.

Každý výrobce CPLD používá trochu jinou interní strukturu obvodů, ale většinou vychází z obr. 1.2. CPLD od různých výrobců se obvykle liší v provedení bloků vlastní programovatelné logiky, i když většinou vychází z klasické struktury SPLD. Obvody CPLD dnes používají nejčastěji paměťové prvky typu FLASH – elektricky programovatelné i mazatelné, s mazáním celé paměti najednou. Často mají samotné prvky struktury obvodu PLD charakter SRAM a konfigurační paměť typu FLASH je integrována na čipu, což nemusí být uživateli na první pohled vůbec zřejmé. Tak jsou uspořádány například obvody řady XC 9500XL firmy Xilinx. Při připojení napájecího napětí se automaticky rozběhne konfigurační proces, který obvod naprogramuje v čase, který je pro uživatele téměř nepostřehnutelný.

Obvody typu FPGA (Field Programmable Gate Array) mají z programovatelných obvodů nejobecnější strukturu a obsahují nejvíce logiky. Obecná struktura je na obr. 1.3. Současné největší obvody FPGA obsahují až 6 milionů ekvivalentních hradel (typické dvouvstupové hradlo NAND). Mezi jejich hlavní přednosti patří mno-

hem pokročilejší programovatelná propojovací struktura, umožňující spojovat jednotlivé funkční bloky "uvnitř" integrovaného obvodu; na rozdíl od SPLD a CPLD, kde každá makrobuňka měla napevno připojen jeden I/O pin, takže vytvářením kaskády makrobuněk, jsme se připravovali o použitelné vývody z obvodu.



Obr. 1.3: Struktura FPGA [1]

Bloky označené IOB (Input/Output Block) představují vstupně-výstupní obvody pro každý vstupně-výstupní pin FPGA. V podstatě se jedná o makrobuňky použité v obvodech SPLD i CPLD. Tyto bloky obvykle obsahují registr, budič, multiplexer a ochranné obvody. Na vhodném místě je programovatelný invertor, aby mohl příslušný vývod pracovat v pozitivní i negativní logice.

Bloky LB (Logic Block) představují vlastní programovatelné logické bloky. Jsou tvořeny polem AND-OR hradel, nebo look-up tabulkou. Lze je řadit kaskádně s mnohými zpětnými vazbami.

Všechny bloky mohou být různě propojeny globální propojovací maticí.

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

Konfigurace obvodů FPGA je uložena ve vnitřní paměti RAM. Po vypnutí napájení se obsah paměti ztratí, takže při každém zapnutí musíme do obvodu znovu nahrát konfigurační data.

Zaměříme se na konkrétní rodinu FPGA firmy Xilinx – Spartan-3. Jedná se o velmi komplexní obvody, které mohou tvořit velmi složité logické funkce. Obvody z rodiny Spartan-3 sestávají z 5 základních typů bloků.

- Configurable Logic Block (CLB) – Nastavitelný logický blok, obsahuje pravdivostní tabulku v RAM pro vykonávání logických operací a paměťové buňky, které mohou být použity jako klopné obvody nebo latche. CLB lze nastavit na vykonávání logických operací nebo pro uchování informací.
- Input/Output Block (IOB) – Vstupně-výstupní blok, řídí přesun dat mezi vývody obvodu a vnitřní logikou. Každý IOB umožňuje obousměrný přenos dat a podporuje i třístavovou funkčnost. Každý vývod obvodu lze nastavit do jednoho z 26 režimů přenosu signálu. IOB dále obsahuje DDR registry a číslicově řízenou impedanci pro jednoduché impedanční přizpůsobení.
- Block RAM – Blok paměti RAM.
- Bloky násobiček – hardwarové násobičky
- Digital Clock Manager (DCM) – Správce hodin umožnuje upravovat hodinový signál.

Jednotlivé dílčí bloky jsou pospojovány programovatelnou propojovací strukturou, umožnující připojit několik bloků na společný propoj.

1.2 Metody konfigurace

Veškeré funkční bloky i spojovací matice jsou řízeny pomocí tzv. "CCL" (CMOS Configuration Latch). Po zapnutí napájení je nutné tyto paměťové buňky "naplnit" konfiguračními daty. Existuje několik způsobů nakonfigurování.

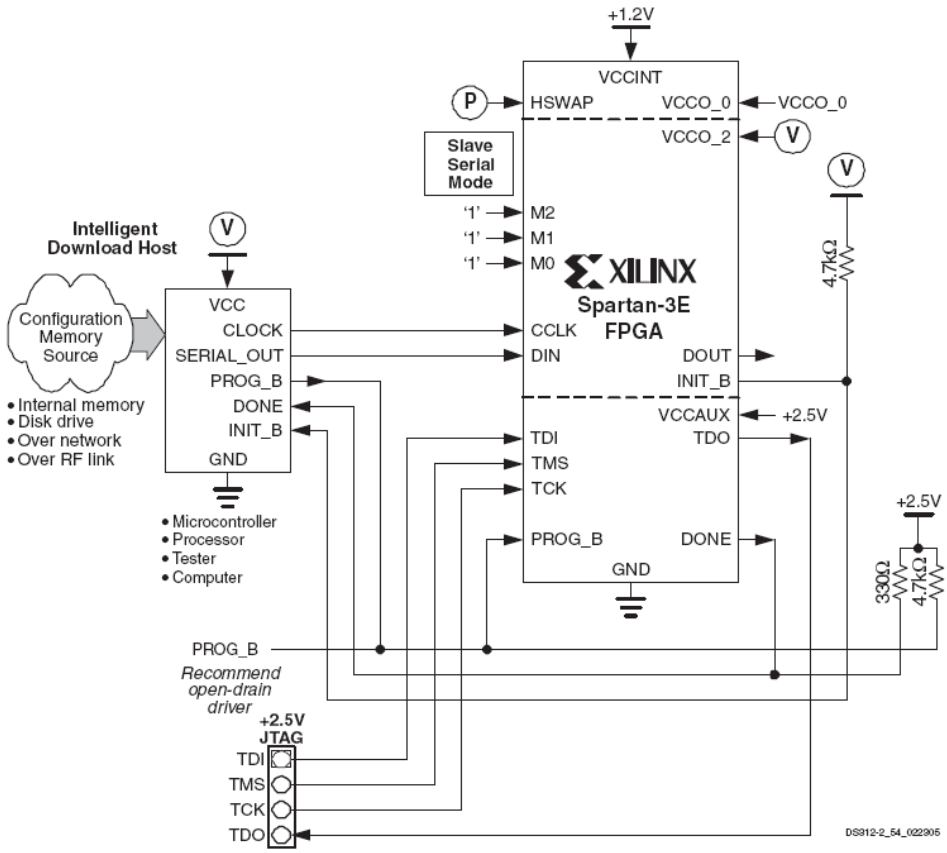
- Master Serial mode
- Master Parallel mode (BPI)
- Slave Serial mode

- Slave Parallel mode (SelectMAP)
- Master SPI
- JTAG mode

Pro naše účely je výhodnější, když bude přenos řízen mikroprocesorem. Proto se režimům "master" nebudeme příliš věnovat.

1.2.1 Slave Serial mode

Konfigurační data jsou do FPGA nahrávána sériově. Rychlost přenosu je řízena hodinovým signálem, který musíme přivádět zvenku na pin CCLK. Jedná se o nejjednodušší metodu konfigurace. Nevýhodou může být rychlost, neboť se data nahrávají "bit po bitu". Na obrázku 1.4 je uvedeno zapojení FPGA k mikrokontroleru, který řídí konfiguraci.



Obr. 1.4: Zapojení při módu konfigurace Slave Serial [2]

Řídíci mikrokontroler zahájí konfiguraci impulzem na pinu **PROG_B**. Kontrolou, zda je na pinu **INIT_B** log. 1, zjístíme, zda je FPGA připraveno na nová konfigurační data. Mikrokontroler poté do FPGA synchroně nahraje data a kontrolou pinu **DONE** zkontroluje uspěšný přenos. Pokus se při konfiguraci objeví log.

FPGA pin	Popis
M[2:0]	výběr programovacího módu – zde 111
DIN	vstup konfiguračních dat
CCLK	vstup hodinového signálu
INIT_B	výstup – indikuje inicializaci FPGA
DONE	výstup – indikuje zda je FPGA úspěšně nakonfigurováno
PROG_B	vstup – umožňuje provést novou konfiguraci již běžícího FPGA

Tab. 1.1: Použité piny při konfiguraci v Slave Serial módu

0 na pinu **INIT_B**, došlo k chybě a celou proceduru je třeba opakovat.

Konfigurační proces potřebuje více hodinových cyklů, než je počet bitů v konfiguračním souboru. Další hodinové impulzy slouží k inicializaci FPGA, zvláště pokud je zvoleno čekání na zasynchronizování DCM na vstupní hodinový signál.

Režim Slave Serial zvolíme slovem 111 na pinech **M[2:0]**. Tyto piny jsou čteny při vzestupné hraně signálu **INIT_B**. Po nakonfigurování FPGA, když výstup **DONE** indikuje úspěšný přenos, lze piny **M[2:0]** používat jako běžné IO piny. Stejně tak vstup **HSWAP**, který v době konfigurace zapíná vnitřní pull-up rezistory, je po nakonfigurování možné používat jako běžný IO pin.

Počet pinů potřebných ke konfiguraci je tedy 5. Většina mikrokontrolerů obsahuje hardwarově realizované SPI rozhraní, které můžeme použít pro odesílání dat, což značně urychlí vlastní konfiguraci.

1.2.2 Master Serial mode

Master Serial Mode je velmi podobný předchozímu režimu konfigurace. Jediným rozdílem je, že hodinový signál je generován samotným FPGA a obvod s konfiguračními daty se musí přizpůsobit. Pin **CCLK** je v tomto případě výstup. Základní kmitočet hodinového signálu je 6MHz.

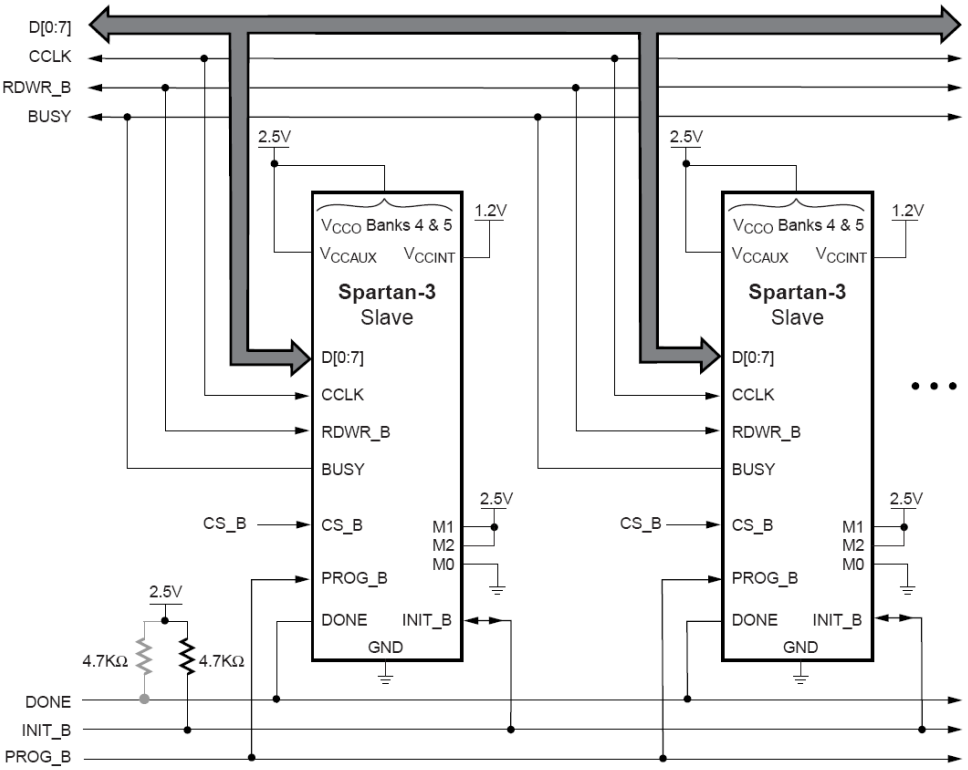
Metoda je velmi výhodná při nahrávání konfigurace z externí paměti Platform Flash. FPGA si samo řídí přenos a všechny Platform Flash firmy Xilinx mají sériové rozhraní.

Pro konfiguraci z mikrokontroleru však tato metoda není vhodná. Je jednodušší hodinový signál generovat mikrokontrolerem, než se přizpůsobovat již existujícímu hodinovému signálu z FPGA.

1.2.3 Slave Parallel mode (SelectMap)

V tomto režimu jsou data do FPGA přenášenena po celých bajtech. K přenosu dat se využívají piny D0 – D7. Tato metoda se vyznačuje větší rychlostí na rozdíl od

sériové metody, je však potřeba více vodičů mezi FPGA a konfiguračním obvodem. Tím pádem se buď ochudíme o několik pinů z FPGA, nebo musíme řešit přepínání mezi konfiguračním mikrokontrolerem a výslednou aplikací.



Obr. 1.5: Zapojení při paralelním módu konfigurace [2]

Jak je vidět z tab. 1.2, je potřeba velké množství konfiguračních vývodů. Jelikož jsou data přenášena paralelně, je přenos z principu 8x rychlejší při použití stejného kmitočtu CCLK. Počet programovacích pinů je zde 15.

1.2.4 Master Parallel mode

Opět velmi podobné Slave Serial módu, avšak s paralelním přenosem dat. Novější FPGA tento režim neumožňují. Master Parallel mód je u nich nahrazen Master BPI módem. Pro naše účely je to opět nevhodná metoda, neboť si FPGA samo generuje hodinový signál.

1.2.5 JTAG mode

Programování probíhá normovaným rozhraním dle IEEE 1149.1/1532. Obvody mají vyhrazené 4 vývody pro JTAG TAP (Test Access Point). Toto rozhraní je aktivní po celou dobu činnosti FPGA. Přepnutí konfiguračního módu na JTAG (M[2:0] =

FPGA pin	Popis
M[2:0]	výběr programovacího módu
CCLK	vstup hodinového signálu
INIT_B	výstup – indikuje inicializaci FPGA
DONE	výstup – indikuje, zda je FPGA úspěšně nakonfigurováno
PROG_B	vstup – umožňuje provést novou konfiguraci již běžícího FPGA
RDWR_B	vstup – musí být v log. 0
CS_B	vstup – chip select
BUSY	výstup – FPGA není schopno příjmout další data
D[7:0]	paralelní vstup konfiguračních dat

Tab. 1.2: Použité piny při konfiguraci v Slave Parallel módu

FPGA pin	Popis
TDI	vstup dat
TDO	výstup dat
TCK	vstup hodinového signálu
TMS	vstup řídícího signálu

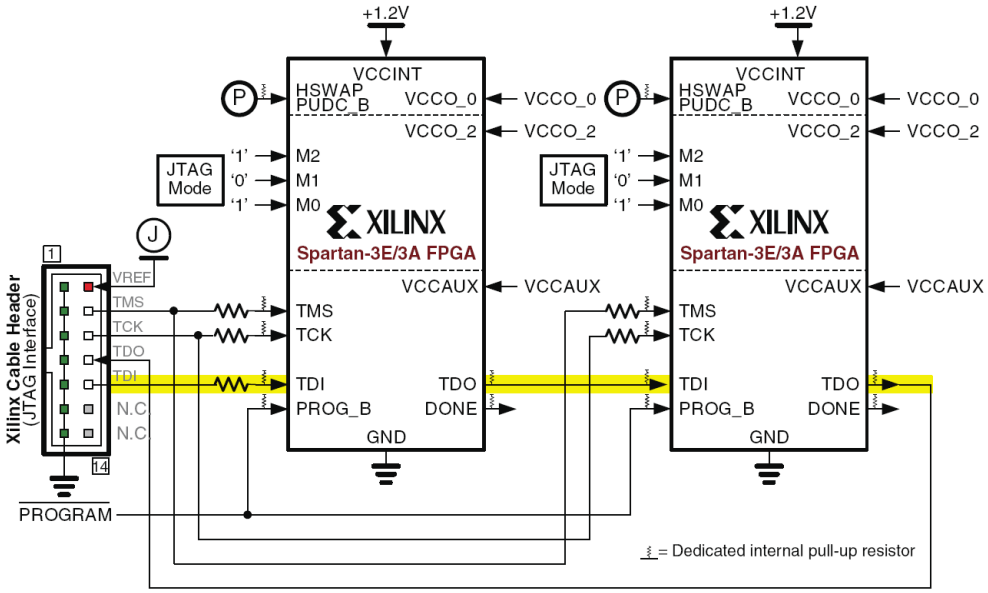
Tab. 1.3: Piny rozhraní JTAG

101) pouze způsobí, že FPGA po zapnutí čeká na konfiguraci. Rozhraním JTAG lze kdykoli za běhu aplikace FPGA znovu rekonfigurovat. Také Platform Flash paměti jsou programovány tímto rozhraním.

Jedná se o sériové synchronní rozhraní. Vstup hodinového signálu je na vývodu TCK, který je generován konfiguračním mikrokontrolerem. Vstupní vývod TMS slouží k výběru stavu rozhraní JTAG. Rozhoduje, zda zapisujeme instrukci nebo příkaz. Vždy při vzestupné hraně hodinového signálu nastává změna stavu JTAG rozhraní podle stavového diagramu (obr. 1.7).

Vlastní data do obvodu vstupují vstupem TDI a vystupují výstupem TDO. Jak budou data interpretována záleží na momentálním stavu JTAG rozhraní. JTAG je koncipován pro konfiguraci několika obvodů, které jsou spojené vývody TDI a TDO kaskádně. Hodinový signál TCK a řídící signál TMS je všem obvodům společný. Datové vývody TDI a TDO musí uzavírat smyčku mezi všemi programovanými obvody a řídícím mikrokontrolerem. Schéma připojení JTAG rozhraní je na obr. 1.6.

Do každého nastavovaného obvodu je nejdříve nahrána instrukce, co s obvodem chceme dělat. Základní možnosti jsou BYPASS (propouští data z TDI do TDO přímo), přímé řízení I/O vývodů nebo nahrání konfiguračních dat. Poté teprve



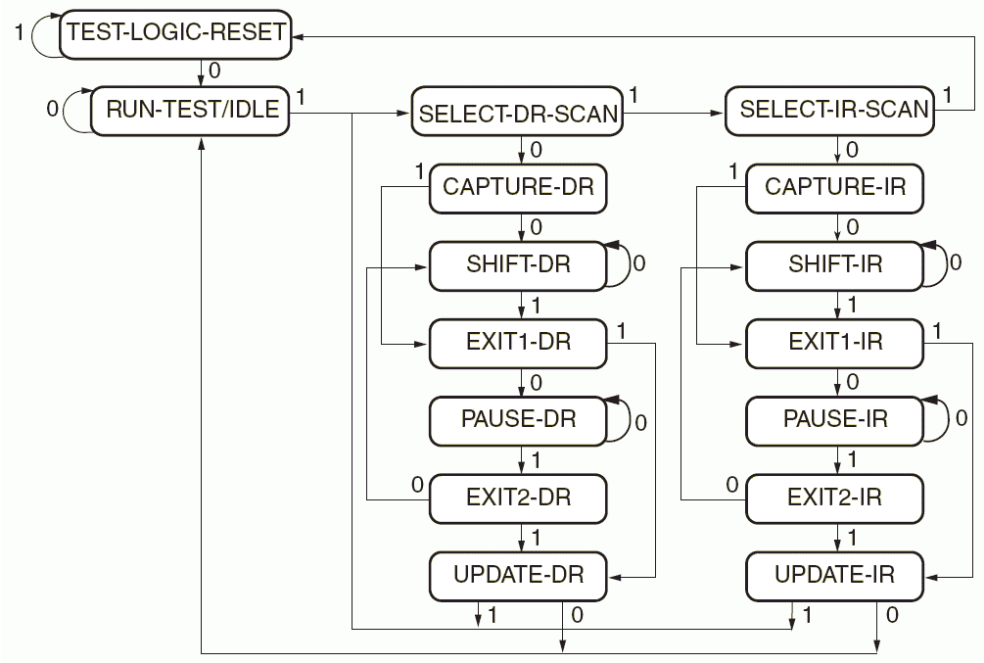
Obr. 1.6: Zapojení přes JTAG [2]

můžeme jednotlivým obvodům poslat data.

Tato metoda je značně komplikovanější pro řídící software v mikrokontroleru, na druhé straně přináší možnost konfigurovat FPGA i programovat externí Platform Flash paměť pomocí jednoho hadwarově jednoduchého rozhraní. Softwarový nástroj pro generování konfiguračních souborů od společnosti Xilinx – iMPACT, vytvoří soubor s daty pro FPGA i s odpovídajícími instrukcemi pro JTAG rozhraní. Samotné programování konfiguračních dat je tedy velmi podobná sériovému programování. Konfigurační soubor neobsahuje prostá data, ale příkazy ve speciálním makrojazyce. Konfigurační mikrokontroler musí tyto příkazy interpretovat.

1.3 Konfigurační soubory

Syntézou zdrojových souborů s definicí vnitřní struktury FPGA obdržíme soubor .bit. Ten obsahuje informace, jak má být dané FPGA nakonfigurované, není však určen pro přímé odeslání do FPGA. Součástí vývojového prostředí pro Xilinx ISE je softwarový nástroj *iMPACT*. Slouží k vytvoření vlastního konfiguračního souboru, který můžeme přímo nahrát do daného FPGA. Můžeme si zvolit, jakou metodu konfigurace použijeme, a podle toho obdržíme vygenerovaný datový soubor, který stačí do FPGA danou metodou odeslat. Pokud si zvolíme konfiguraci přes JTAG rozhraní, vygenerovaný soubor obsahuje kromě konfiguračních dat i příkazy pro uvedení JTAG rozhraní do správného módu činnosti. Formáty souborů pro JTAG existují dva. Jeden je textový, druhý je binární a úspornější.



Obr. 1.7: Stavy rozhraní JTAG [2]

1.3.1 Soubory SVF

SVF (Serial Vector Format) je textový soubor, obsahující příkazy pro odeslání přes JTAG rozhraní. Má velmi jednoduchou syntaxi (viz 3). Obsahuje příkaz a parametry oddělené mezerami a ukončené středníkem. Jednotlivé příkazy určují, do jakého stavu je nutné uvést JTAG rozhraní a jaká data máme odeslat. Jedná se v podstatě o jakýsi makrojazyk, který musí zpracovávat interpreter. Příklad souboru je uveden v tab. 1.5.

Příkazy HIR, HDR, TIR a TDR se využívají, pokud máme přes JTAG spojených kaskádně více obvodů.

1.3.2 Soubory XSVF

XSVF je binární podoba SVF souboru. Každý příkaz je nahrazen jednobytovým kódem a všechny parametry jsou uloženy v binární podobě. Soubor ve formátu XSVF je znatelně menší (asi 4x) než odpovídající textový SVF soubor. V aplikační poznámce xapp058 (viz [6]) firmy Xilinx je uveden zdrojový kód v jazyce C pro interpretaci XSVF souborů. Lze jej využít pro vytvoření XSVF interpreteru přímo v mikrokontroleru. Binární formát je vhodnější nejenom z úspory místa, ale i z důvodu snadnější realizace interpreteru.

Příkaz	Popis
STATE	uvede JTAG rozhraní do definovaného stavu
ENDIR	definuje stav, kde se má skončit po instrukci SIR
ENDDR	definuje stav, kde se má skončit po instrukci SDR
HIR	definuje hlavičkum, která se bude odesílat před každou SIR instrukcí
HDR	definuje hlavičkum, která se bude odesílat před každou SDR instrukcí
SIR	odeslání dat do IR registru
SDR	odeslání dat do DR redistru
RUNTEST	přepne JTAG do stavu běh (RUNSTATE) na definovanou dobu
! nebo //	komentář

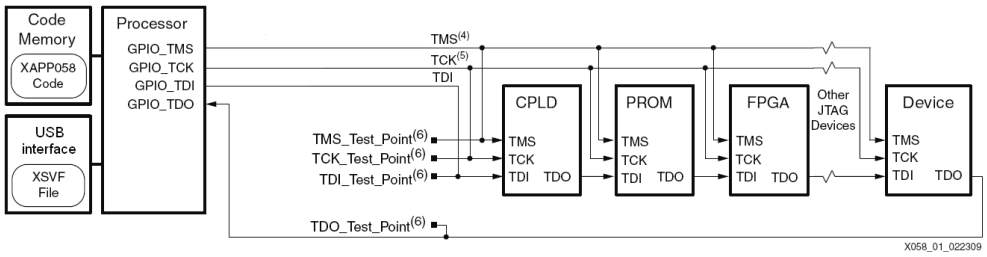
Tab. 1.4: Některé příkazy SVF souboru

```
TRST OFF; !Disable Test Reset line
ENDIR IDLE; !End IR scans in IDLE
ENDDR IDLE; !End DR scans in IDLE
HIR 8 TDI (00); !8-bit IR header
HDR 16 TDI (FFFF) TDO (FFFF) MASK (FFFF);!16-bit DR header
TIR 16 TDI (0000); !16-bit IR trailer
TDR 8 TDI (12); !16-bit DR trailer
SIR 8 TDI (41); !8-bit IR scan
SDR 32 TDI (ABCD1234) TDO (11112222); !32-bit DR scan
STATE DRPAUSE; !Go to stable state DRPAUSE
RUNTEST 100 TCK ENDSTATE IRPAUSE; !RUNBIST for 100 TCKs
```

Tab. 1.5: Příklad SVF souboru

1.4 Shrnutí programovacích metod

Analýzou konfiguračních metod se dostaneme k závěru, že paralelní metody potřebují velký počet konfiguračních pinů, jak na straně FPGA tak i u mikrokontroleru. Jako vhodné se jeví použít sériové nahrávání nebo JTAG rozhraní. Jelikož se FLASH paměti pro FPGA (Platform Flash) dají jednoduše konfigurovat přes JTAG rozhraní, lze JTAG označit za nejvhodnější. Samotná Platform Flash může do FPGA konfigurační data nahrávat metodou Master Serial Mode.



Obr. 1.8: Schéma připojení mikrokontroleru přes JTAG rozhraní [2]

1.5 Návrh hardwaru

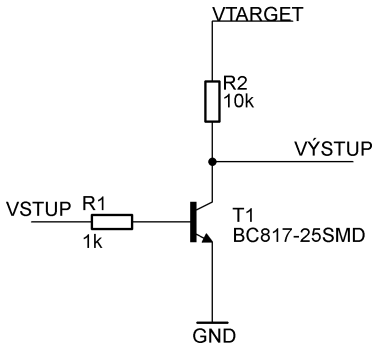
1.5.1 Volba mikrokontroleru

Mezi počítačem a vlastním FPGA obvodem je potřeba umístit mikrokontroler, který bude interpretovat příkazy z PC a posílat data do FPGA. Dále je nutné vyřešit připojení mikrokontroleru k PC.

Na straně PC jsou 2 rozhraní, připadající v úvahu – Sériový port a USB. Jelikož je však sériový port na ústupu a některé počítače jej již vůbec nemají, je vhodnější rozhraní USB. Na druhou stranu je USB mnohem složitější než jednoduchý sériový port. Máme dvě možnosti jak mikrokontroler připojit přes USB. První je použít mikrokontroler s integrovaným USB řadičem. Je zde však problém se špatnou dostupností takovýchto mikrokontrolerů. Většinou je složité je naprogramovat a musí se pro tyto mikrokontrolery vytvořit vlastní ovladače pro Windows. I když výrobci jako Microchip, Atmel a Freescale vyrábí celou řadu mikrokontrolerů s integrovaným USB rozhraním, nejsou tyto obvody v nabídce tuzemských prodejců součástek. V nabídce obchodů GM, GES ani SOS tyto mikrokontrolery nenajdeme. Možnost nechat si poslat vzorky zdarma (Free samples) je vhodná k získání 1-3 kusů. Už při 10 kusech je však velký problém s objednáním, proto obvody takto sehnatelné nelze považovat za dostupné.

Druhá možnost je použít hotové převodíky "USB to serial". Zde máme většinu práce hotovou od výrobce. Převodník připojíme k mikrokontroleru sériovou linkou. Jelikož skoro každý mikrokontroler má dnes implementované rozhraní UART, není s připojením žádný problém. Na straně PC jsou již hotové ovladače, které lze jednoduše stáhnout z internetu.

Nejznámějším výrobcem převodníků *USB to serial* je firma FTDI. Do jejich sortimentu patří vynikající obvod FT232RL, který navazuje na dříve velmi populární FT232BL. Obvod FT232RL se vyznačuje minimem potřebných externích součástek a nepříliš velkým pouzdrem – TSSOP28. Z vnějšku stačí připojit pouze 2 rezistory a 1 kondenzátor. Vše ostatní obsahuje obvod uvnitř, například interní oscilátor nebo konfigurační EEPROM. Dokonce obsahuje i lineární stabilizátor s výstupem 3,3V.



Obr. 1.9: Obvod napěťového přizpůsobení

Hlavní výhodou je však dostupnost. Můžeme jej najít v nabídce GM Electronics i SOS Electronics. Ovladače pro Windows jsou na Microsoft Update, takže operační systém si jej dokáže z internetu stáhnout sám.

Pokud se rozhodneme pro řešení s externím převodníkem na USB, můžeme použít téměř libovolný mikrokontroler. Nejlepšího poměru cena/výkon/dostupnost dosahují obvody ATmega firmy Atmel. Mikrokontrolery z těchto rodin mají integrovaný UART, který můžeme použít k připojení k PC a disponují i hardwarovým SPI rozhraním, které můžeme využít pro sériové posílání dat.

Pro programátor bude postačovat nejmenší mikrokontroler s této rodiny – ATmega8. Obsahuje 8kB interní FLASH programové paměti, 1kB operační RAM paměti, rozhraní USART i SPI a integrovaný RC oscilátor s volitelným kmitočtem 1, 2, 4 nebo 8 MHz.

Rozhraní SPI lze výhodně použít při odesílání velkého počtu bitů. Rychlost přenosu tak bude srovnatelná s hodinovým kmitočtem oscilátoru. Při užití vnitřního RC oscilárotu 8 MHz, bude dosaženo rychlosti skoro 8Mbit/s.

1.5.2 Napěťové přizpůsobení

Obvody FPGA používají napětí pro I/O vývody nejčastěji 3,3V, 2,5V nebo 1,8V. Navrhovaný programátor by měl zvládnout pracovat při jakémkoli napájení cílového FPGA. Napájení převodníku FT232RL i mikrokontroleru je však nejméně 3,3V. Je proto nutné realizovat napěťové přizpůsobení mezi mikrokontroler a cílové obvody.

Při větším počtu datových linek by bylo vhodnější použít specializovaný obvod. Nám však postačí jednoduché zapojení z diskrétních součástek.

Jelikož jsou linky rozhraní jednosměrné, stačí k napěťovému přizpůsobení zapojení tranzistoru se společným emitorem viz obr. 1.9.

1.5.3 Externí paměť dat

Obvody FPGA a CPLD mají konfigurační paměť zpravidla tvořenou paměťovými buňkami RAM, které se po odpojení napájení smažou. Programátor je proto vybaven externí pamětí typu FLASH, která umožní do programovaného obvodu uložit konfigurační data vždy po připojení napájení k FPGA, aniž by bylo potřeba data opakovaně posílat z PC.

Použita je paměť 45DB041B-SU, která se v prodejní siti GM Electronic prodává za 41,- Kč. Její kapacita je 4Mb, což postačuje pro velkou část PLD obvodů. Paměť komunikuje rozhraním SPI, které je přímo spojeno s hardwarovým SPI modulem v mikrokontroleru.

1.6 Software pro mikrokontroler

1.6.1 Překladač a programovací jazyk

Při výběru programovacího jazyka pro napsání softwaru pro mikrokontroler jsem se rozhodl pro **C**. K mikrokontrolerům řady ATmega existuje freewarový kompilátor WinAVR, se kterým mám celkem dobré zkušenosti. Není však úplně bez chyby a většinou je nutné nezapínat příliš silné optimalizace. Zpravidla postačuje volba -o1. Zapnutím lepších optimalizací se výsledný kód zmenší jen o málo, ale za to dost často dojde k narušení funkce celého programu.

Samotná firma Atmel bezplatně poskytuje vývojové prostředí "AVR Studio"[7] obsahující simulátor, pro snadné ladění aplikací.

1.6.2 Uspořádání programu

Celý program je pro přehlednost rozdělen do několika souborů. Nejdůležitější z nich jsou uvedeny v tab. 1.6. Ke každému souboru s koncovkou **.c** samozřejmě náleží i příslušný soubor s hlavičkami funkcí **.h**. Některé soubory jsou převzaty z [6]. Jsou však upraveny a optimalizovány pro efektivní překlad a minimalizaci zabíraného místa. Převzaté funkce byly napsány obecně pro implementaci do různých mikrokontrolerů. Proto byly části, které nejsou v našem případě potřeba, promazány a přepsány.

Obecné programové konstanty (např. rychlost oscilátoru) jsou umístěny v souboru **global.h**. Často používané datové typy jsou s krátkými názvy definovány v **common_defs.h**. Hlavní programový soubor **JTAG_prog.c** obsahuje inicializaci mikrokontroleru a funkci zpracovávající přicházející povely z PC.

soubor	popis
JTAG_prog.c	hlavní programový soubor
micro.c	softwarové JTAG rozhraní
ports.c	nízkoúrovňové funkce pro JTAG rozhraní
uart.c	obsluha hardwarového UARTu
lenval.c	datová struktura pro JTAG data
spi.c	funkce pro obsluhu SPI rozhraní
spiflash.c	funkce pro obsluhu datové paměti
global.h	definice globálních konstant
common_defs.h	definice obecných datových typů

Tab. 1.6: Seznam souborů se zdrojovými kódy pro mikrokontroler

1.6.3 Softwarový JTAG port

Základem pro vytvoření softwarového JTAG portu se stal program převzatý z [6]. Původní převzatý program byl napsán velmi zvláštním stylem a některé nízkoúrovňové funkce byly nefunkční. Proto si vyžádal mnoho úprav. Využity byly funkce pro přechod mezi jednotlivými stavy JTAG rozhraní a odesílání dat do IR a DR registrů.

Jelikož je JTAG rozhraní univerzální, mohou být do řetězce připojeny i obvody jiných výrobců. Tyto jiné obvody je při programování nutné uvést do režimu "bypass". V tomto režimu, který je popsán v normě JTAG, se obvod chová jako jednobitový posuvný registr, proto je nutné na začátek a konec přenášených dat vkládat potřebný počet dodatečných bitů.

Nejprve je nutné určit počet připojených obvodů. Zvolil jsem velmi jednoduchou metodu. Všechny obvody přivedu do standardizovaného režimu "bypass", kdy se každý obvod chová jako 1-bitový posuvný registr. Velkým problémem JTAGu je, že normou není specifikována délka instrukčního registru a u jednotlivých výrobců se liší. Naštěstí je instrukce "bypass" definována jako samé log. 1. Stačí tedy do IR poslat počet log. 1 odpovídající součtu délek jednotlivých IR všech uvažovaných obvodů. Pokud je řetězec IR kratší, přebytečné bity se z konce řetězce vrátí zpět do programátoru, jako kdyby to byl předchozí obsah IR.

Když máme všechny obvody v "bypass" režimu, stačí do datového řetězce poslat jednu log. 1 a za ní samé nuly. Počet hodinových cyklů, než se tato log. 1 objeví na výstupu z řetězce, odpovídá počtu obvodů v řetězci plus jedna.

Dále můžeme zahájit detekci konkrétního typu připojeného obvodu. Ta se provádí speciální instrukcí, která do datového registru vloží 32-bitové identifikační slovo, které obsahuje kód výrobce, kód součástky a číslo verze.

název funkce	**popis**
jtag_Initialize	Inicializace JTAG rozhraní
jtag_shift	vnitřní funkce, odesílá a přijímá JTAG data
jtag_Reset	uvádí JTAG rozhraní cílového obvodu do počátečního stavu
jtag_getID	zjistí identifikační řetězec cílového obvodu
jtag_CountDevices	zjistí počet připojených obvodů
jtag_Erase	smaže cílové CPLD
jtag_ProgramBegin	zahájí programování cílového CPLD
jtag_ProgramEnd1	ukončuje programování cílového CPLD
jtag_ProgramEnd2	ukončuje programování cílového CPLD
jtag_ProgramBlock	zapisuje vlastní data do CPLD
jtag_ReadBegin	zahajuje čtení dat
jtag_ReadBlock	přečtě blok dat z CPLD
jtag_ReadEnd	ukončuje čtení dat

Tab. 1.7: Funkce v souboru micro.c

Vlastní konfigurace se provádí pomocí sekvencí příkazů okopírovaných ze SVF souborů vygenerovaných iMPACTem. Jedná se o mazání, programování a čtení cílového obvodu.

Softwarová emulace JTAG rozhraní je umístěna v souboru **micro.c** a několik nízkoúrovňových funkcí je v souboru **ports.c**. Zde se nachází funkce pro identifikaci připojených obvodů (**jtag_getID a jtag_CountDevices**) i pro vlastní konfiguraci (**jtag_Erase, jtag_ProgramBegin, ...**). Všechny funkce jsou shrnuty v tab. 1.7.

Nejprve je nutné JTAG rozhraní inicializovat voláním funkce **jtag_Initialize**, které předáme ukazatele na vstupní a výstupní buffer. Funkcí **jtag_Reset** uvedeme cílový obvod do počátečního stavu (stav TEST-LOGIC-RESET na obr. 1.7).

Nyní lze zjistit počet připojených obvodů voláním funkce **jtag_CountDevices**. Ve výstupním bufferu jsou 4 byty, v nichž se nachází jen jedna log. 1. Podle její pozice můžeme určit počet připojených obvodů.

Funkcí **jtag_getID** vyčteme z připojeného obvodu jeho identifikační číslo, jež má podle specifikace vždy 32 bitů a uloží jej do výstupního bufferu.

Před zahájením programování je nutné smazat konfigurační paměť pomocí funkce **jtag_Erase**. Tato funkce nepotřebuje žádné parametry ani nevrací žádný výsledek.

Samotné programování se zahájí voláním **jtag_ProgramBegin**, čímž se CPLD přepne do programovacího režimu. Nyní se nahrají jednotlivé bloky dat pomocí funkce **jtag_ProgramBlock**. Před každým jejím voláním musíme do vstupního bufferu dát 172 bytů konfiguračních dat. Po zapsání všech konfiguračních dat je nutné zavolat funkci **jtag_ProgramEnd1** a poté pomocí **jtag_ProgramBlock**

název funkce	**popis**
uartInit	Inicializace UART rozhraní
uartSetBaudRate	nastaví komunikační rychlost
uartSetBuffers	nastaví vysílací a přijímací vyrovnávací paměť
uartGetRxByteCount	vrátí počet přijatých znaků
uartSetRxByteCount	zjistí identifikační řetězec cílového obvodu
uartRetTxStatus	vrátí informaci, zda dochází k odesílání či nikoli
uartSendTxBuffer	odešle daný počet bytů z vysílací vyrovnávací paměti

Tab. 1.8: Funkce v souboru uart.c

znovu zapsat posledních 172 bytů s modifikovaným ENABLE bitem. Nakonec se celá programovací procedura ukončí zavoláním **jtag_ProgramEnd2**, čimž se CPLD uvede do provozního stavu.

Z důvodu kontroly správnosti zapsání dat, lze konfigurační paměť přečíst s pomocí funkcí **jtag_ReadBegin**, **jtag_ReadBlock** a **jtag_ReadEnd**.
Funkci **jtag_ReadBlock** se předává parametr určující, kterou část konfigurační paměti chceme přečíst. Přečtený blok dat se uloží do výstupního bufferu.

1.6.4 Obsluha sériového portu

Mikrokontroler s PC komunikuje prostřednictvím hardwarového sériového portu. Funkce pro jeho snadnou obsluhu jsou umístěny v souboru **uart.c**. Jednotlivé funkce jsou shrnuty v tab. 1.8.

Sériové rozhraní se inicializuje funkcemi **uartInit** a **uartSetBaudRate**, které předáme komunikační rychlost v baudech, a **uartSetBuffers**, které předáme ukazatele na přijímací a vysílací buffer. V našem případě je rychlost nastavena na 500 000 baudů a buffery jsou shodné s buffery pro rozhraní JTAG. Přijímací buffer pro sériový port je zároveň odesílacím bufferem pro JTAG a opačně. Toto řešení jednak šetří potřebnou paměť a navíc odpadají přesuny dat mezi vyrovnávací pamětí pro JTAG a UART.

Příjem probíhá neustále, neboť se o něj stará hardwarové přerušení. Počet přijatých znaků lze zjistit z návratové hodnoty funkce **uartGetRxByteCount**. Pokud chceme, aby se data přijímala znovu na začátek vyrovnávací paměti, nastavíme pomocí funkce **uartSetRxByteCount** počet přijatých znaků na 0.

Vysílání se provádí funkcí **uartSendTxBuffer**, které předáme, kolik bytů z vysílacího bufferu se má odeslat. Pokud potřebujeme počkat na dokončení přenosu, lze stav odesílání zjistit z návratové hodnoty funkce **uartRetTxStatus**, kde hodnota *true* znamená probíhající přenos a hodnota *false* ukončený přenos.

název funkce	**popis**
spiflashInit	Inicializace SPI rozhraní a připojené paměti
spiflashGetID	Přečte z paměti identifikační slovo – ověří činnost komunikace
spiflashChipErase	Smaže celou paměť
spiflashRead	Přečte daný počet bytů z externí paměti
spiflashWrite	Uloží daný počet bytů do externí paměti

Tab. 1.9: Funkce v souboru spiflash.c

1.6.5 Externí paměť dat

Externí paměť je připojena přes rozhraní SPI. Ke snadné manipulaci slouží soubor **spiflash.c**, kde jsou umístěny funkce pro přímou práci s pamětí (viz tab. 1.9). Také je potřeba soubor **spi.c**, která obsahuje nízkoúrovňové funkce pro práci se SPI rozhraním. Oba soubory jsou převzaty z [8].

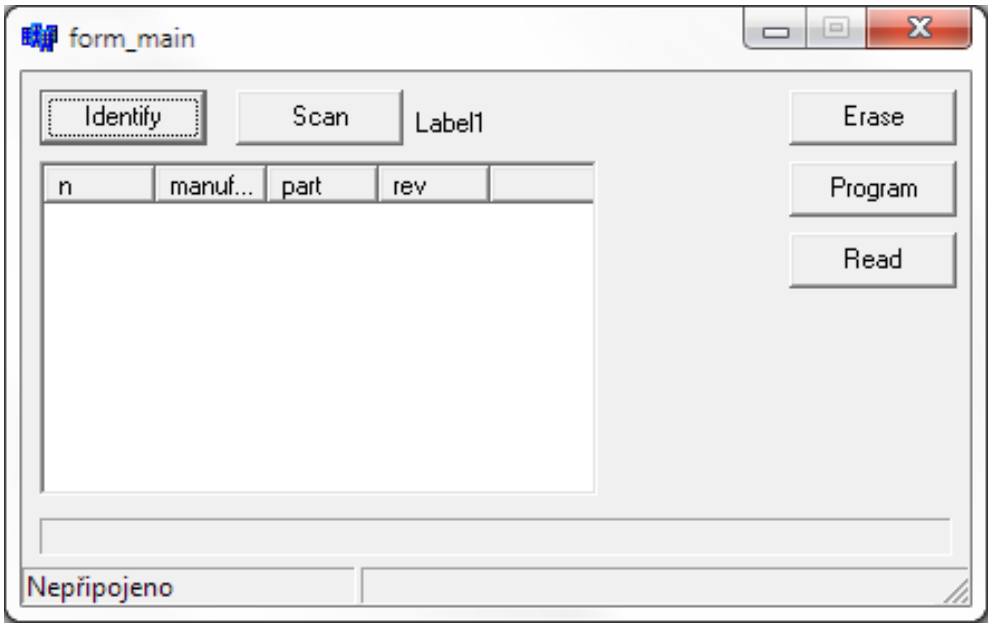
1.7 Software pro PC

1.7.1 Obecně

K softwaru uvnitř mikrokontroleru bylo potřeba vytvořit obpovídající "protikus" na stranu PC. Program je vytvořen ve vývojovém prostředí **C++ Builder** od firmy Borland. Toto vývojové prostředí je velmi intuitivní, urychluje programování a využívá dobře známý programovací jazyk C++. Ve srovnání s jinými vývojovými prostředími je efektivní, neboť minimalizuje čas strávený vytvářením částí programu, které se starají o podpůrné funkce, jako například vytvoření tlačítka ve formuláři, zpracování příchozích zpráv o stisknutém tlačítku, atd. Verze *Personal* je dokonce k dispozici po bezplatné registraci.

1.7.2 Grafické rozhraní

Rámcovou předlohou se stal ovládací program k jedné z vývojových desek od firmy Xilinx. Obslužný program se skládá z jediného okna. V něm jsou umístěna tlačítka pro zvolení konfiguračního souboru, indentifikaci,vymazání a nakonfigurování cílového FPGA. Dále okno obsahuje stavový řádek, kde se zobrazuje stav programátoru. Uprostřed se nachází seznam, který zobrazuje aktuální průběh vykonávání konfigurace. Názorná ukázka je uvedena na obr. 1.10.



Obr. 1.10: Okno programu pro PC

1.7.3 Komunikační protokol

Mezi Atmegou a programem v PC probíhá komunikace pomocí jednoduchého protokolu. Na začátku rámce je vždy znak určující příkaz, který má Atmega vykonat. Z důvodu jednoduchosti jsou všechny příkazy jednobytové, což velmi usnadňuje jejich zpracování v mikrokontroleru. Za příkazovým znakem se vyskytují, pokud to příkaz vyžaduje, data. Seznam příkazů je uveden v tab. 1.10.

Na začátku je potřeba spočítat počet zařízení na JTAG sběrnici. Pak zvolíme, se kterým zařízením budeme komunikovat. Poté již můžeme přečíst z obvodu identifikační kód, který nám určí přesný typ FPGA či CPLD.

Samotná konfigurace se provádí pomocí příkazu **erase** a skupiny příkazů **prog**. Po zahájení příkazem **prog begin** následuje série příkazů **prog data** vždy následovaných částí dat. Atmega po vykonání příkazu odešle do PC první znak příkazu, jako potvrzení úspěšného vykonání. Po tomto potvrzení můžeme posílat další příkaz. Po odeslání všech konfiguračních dat je nutné programování ukončit příkazem **prog end**.

Příkaz	**Název**	**Popis**
?	info	programátor odešle svůj identifikační řetězec
c	count	spočítá počet JTAG zařízení
i	identify	identifikuje připojené zařízení
e	erase	smaže PLD
p	prog_begin	začátek programování
q	prog_data	odesílá data do PLD
r	prog_end1	konec programování 1
s	prog_end2	konec programování 2
m	read_begin	zahájí čtecí režim
n	read_data	přečte blok dat z PLD
o	read_end	ukončí režim čtení

Tab. 1.10: Příkazy komunikačního protokolu

2 VÝSLEDKY STUDENTSKÉ PRÁCE

2.1 Výsledky

2.1.1 Testovací podmínky

Činnost zařízení byla vyzkoušena na vývojové desce s obvodem CPLD CoolRunner-II (XL2C256). Deska obsahuje originální programátor firmy Xilinx který sloužil jako rámcová předloha. Dále je tato deska vybavena konektorem pro připojení JTAG programátoru, přes který byl připojen navrhovaný programátor.

2.1.2 JTAG příkazy

V dokumentaci od firmy Xilinx jsou zmíněny jen základní JTAG příkazy definovány normou pro testování desek plošných spojů. Příkazy potřebné pro programování nejsou v žádné veřejně dostupné dokumentaci. Dle návodů firmy Xilinx se z konfiguračního souboru **.bit** má pomocí programu iMPACT vytvořit konfigurační skript pro JTAG programátor. Použití programu iMPACT je však nadbytečné a vygenerovaný konfigurační skript není příliš optimální. Proto bylo úkolem navrhnout programátor pracující přímo se soubory **.bit**.

Z vygenerovaného skriptu lze však zjistit, posloupnost příkazů při jednotlivých krocích konfigurace CPLD. Z těchto posloupností byl vytvořen program pro mikrokontroler Atmega, který má být schopen cílové PLD smazat, zapsat do něj konfigurační data a přečíst co je v PLD uloženo z důvodu kontroly správnosti zápisu.

2.1.3 Výsledná činnost

Navrhovaný programátor je schopen identifikovat připojený obvod a smazat jeho konfigurační paměť. Z doposud neznámého důvodu však programátor není schopen zapisovat a číst konfigurační data.

Při pokusu o zápis se do cílového CPLD nějaká data uloží, neodpovídají však datům v konfiguračním souboru. Hlavním důvodem je, že firma Xilinx neuvádí přesný postup programování přes JTAG rozhraní. Doporučený postup je použití programu *iMPACT* k převodu souboru **.bit** na soubor **.svf**. Přitom ale originální programátor načítá přímo soubor **.bit** a program *iMPACT* není nutné vůbec spouštět.

Z uvedeného vyplývá, že navrhovaný programátor je celkově nefunkční a může sloužit jen jako předloha pro další vývoj.

Jelikož nebyla zprovozněna ani základní funkce - přímá konfigurace PLD, v programátoru není implementována práce s externí datovou pamětí, ani možnost adresovat konkrétní obvod při zřetězeném připojení více obvodů přes JTAG rozhraní.

Z toho důvodu také není dokončen program pro PC. Přiložená verze je jen provizorní, jelikož nemá dokončenou grafickou úpravu a neumožňuje volbu konfiguračního souboru. Obsahuje však všechny náležitosti potřebné pro komunikaci s programátorem v současném stavu.

3 ZÁVĚR

Práce se věnuje návrhu jednoduchého programátoru založeném na mikrokontroleru pro konfiguraci obvodů PLD. Je popsána základní koncepce obvodů FPGA/CPLD a jednotlivé metody nahrávání konfiguračních dat do těchto obvodů. Jako nejoptimálnější z hlediska složitosti byla vybrána konfigurace pomocí JTAG rozhraní.

S ohledem na cenu a dostupnost byl vybrán mikrokontroler Atmega8 od firmy Atmel. Připojení přes USB rozhraní je řešeno externím převodníkem FT232RL.

Jelikož obvody PLD mohou pracovat při různých napájecích napětích, je mikrokontroler od konfigurovaného PLD oddělen obvodem napěťového přizpůsobení.

Navržený hardware splňuje zadané požadavky a lze jej bez problému využít.

Software v mikrokontroleru interpretuje příkazy zaslané z PC. Funkční je však pouze přečtení identifikačního slova PLD obvodu a smazání konfigurační paměti. Příkazy pro zápis a čtení konfigurační paměti se nevykonávají správně.

Navrhovaný programátor je ve své podstatě nefunkční a může sloužit jen jako předloha pro další vývoj.

LITERATURA

[1] Doc. Ing. Kolouch, J., CSc. *Programovatelné logické obvody a hradlová pole* [online]. [cit. 11-10-2009] Dostupné z URL: <www.urel.feec.vutbr.cz/k̃olouch/pld/>.

[2] Xilinx, Inc. *Xilinx UG332 – Spartan-3 Generation Configuration User Guide* [online]. [cit. 11-10-2009]. Dostupné z URL: <www.xilinx.com/support/documentation/user_guides/ug332.pdf>.

[3] Xilinx, Inc. *Datasheet DS099 – Spartan-3 FPGA Family data sheet* [online]. [cit. 09-11-2009]. Dostupné z URL: <www.xilinx.com/support/documentation/data_sheets/ds099.pdf>.

[4] ASSET InterTech, Inc. *Serial Vector Format Specification* [online]. [cit. 20-12-2009]. Dostupné z URL: <www.jtagtest.com/pdf/svf specification.pdf>.

[5] Xilinx, Inc. *Application note 503 – SVF and XSVF File Formats for Xilinx Devices* [online]. [cit. 20-12-2009]. Dostupné z URL: <www.xilinx.com/support/documentation/application_notes/xapp503.pdf>.

[6] Xilinx, Inc. *Application note 058 – Xilinx In-System Programming Using an Embedded Microcontroller* [online]. [cit. 11-10-2009]. Dostupné z URL: <www.xilinx.com/support/documentation/application_notes/xapp058.pdf>.

[7] Atmel, Corp. *AVR Studio - software* [online]. [cit. 18-05-2010]. Dostupné z URL: <www.atmel.com/dyn/products/tools_card_v2.asp?tool_id=2725>.

[8] Pascal Stang. *Procyon AVRlib* [online]. [cit. 20-03-2009]. Dostupné z URL: <http://hubbard.engr.scu.edu/embedded/avr/avrlib/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

PLD Programmable Logic Device - programovatelný logický obvod

SPLD Simple Programmable Logic Device - jednoduchý programovatelný logický obvod

CPLD Complex Programmable Logic Device - složitý programovatelný logický obvod

FPGA Field-programmable gate array - matice programovatelných hradlových polí

PC Personal computer - osobní počítač

ASIC Application-specific integrated circuit - zákaznické IO

LB Logic Block - logický obvod

JTAG Joint Test Action Group - rozhraní pro konfiguraci a ladění obvodů zapojených v koncové aplikaci

DCM Device Clock Manager - obvod pro distribuci hodinového signálu

IOB Input-Output Block - Vstupně-výstupní blok

USB Universal Serial Bus - Sériová sběrnice určená k připojení periferií k PC

SPI Serial Peripheral Interface - Synchronní sériové rozhraní

USART Universal Synchronous Asynchronous Receiver Transmitter - sériové rozhraní s mnoha režimy funkce

UART Universal Asynchronous Receiver Transmitter - jednodušší verze USARTu, neumožňuje synchronní režim

SVF Serial Vector Format - makrojazyk popisující proceduru programování JTAG rozhraním

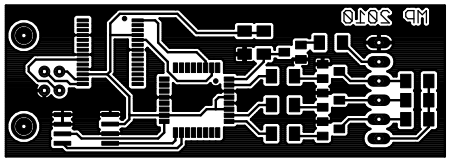
DR Data Register - datový registr rozhraní JTAG

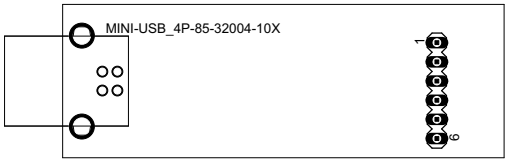
IR Instruction Register - instrukční registr rozhraní JTAG

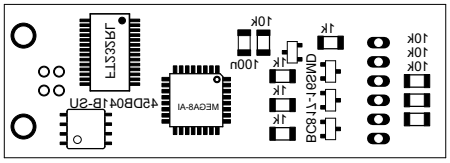
SEZNAM PŘÍLOH

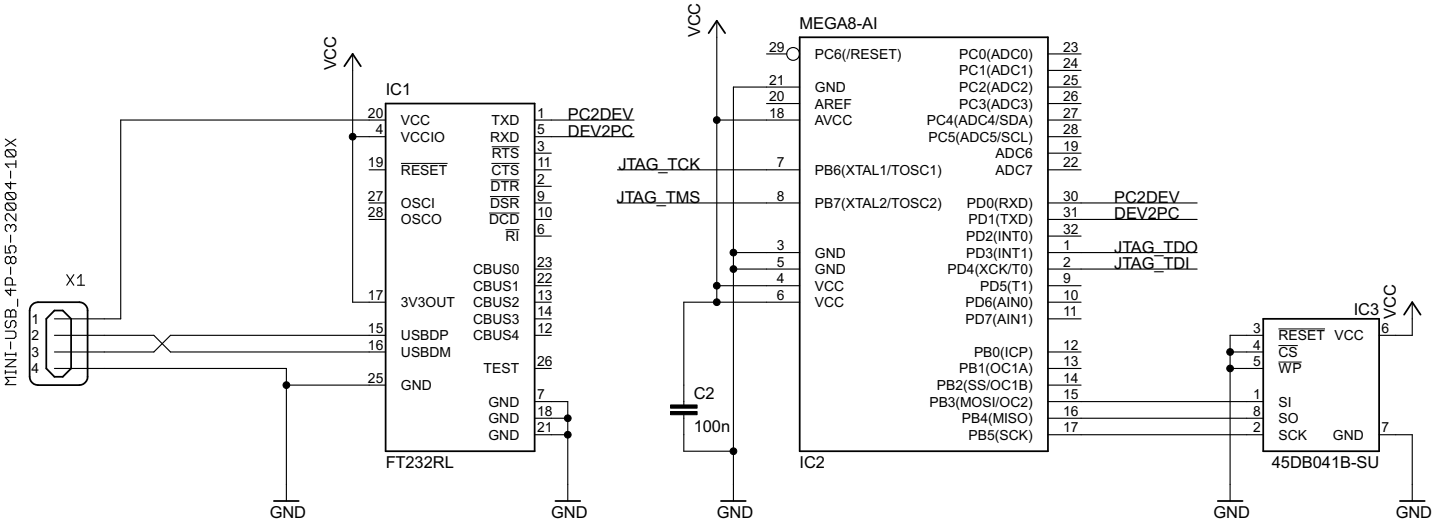
A SCHÉMATA

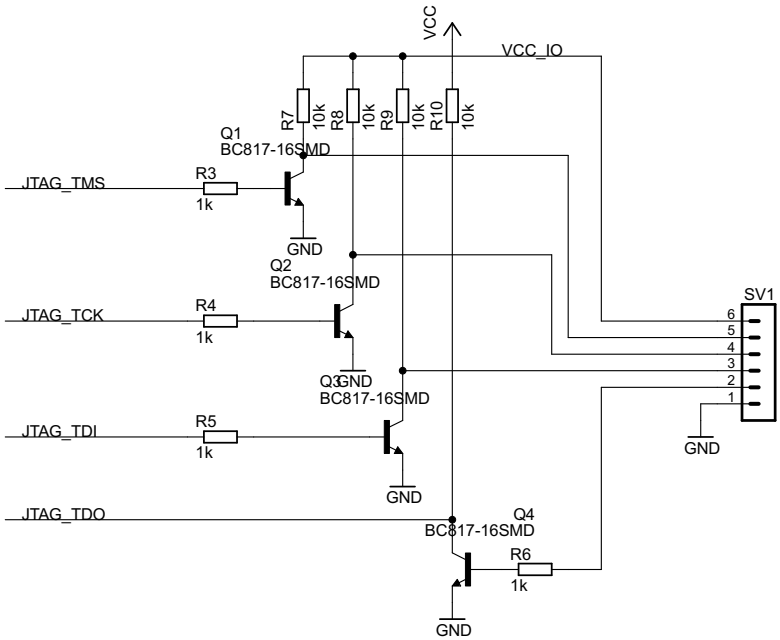
A.1 Schéma celého programátoru











B ZDROJOVÉ KÓDY PROGRAMŮ

B.1 Program pro mikrokontroler

viz přiložený disk CD složka **FW**

Kompilováno v AVR Studio 4.14 kompilátorem WinAVR-20090313.

B.2 Obslužný program pro počítač

viz přiložený disk CD složka **SW**

Kompilováno v Borland C++ Builder 2009.