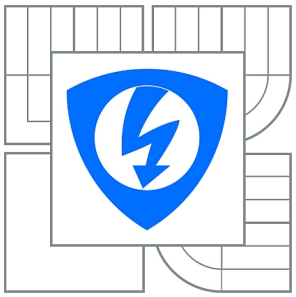
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

DYNAMICKÁ REKONFIGURACE S ATMEL FPSLIC

DYNAMIC RECONFIGURATION WITH ATMEL FPSLIC

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

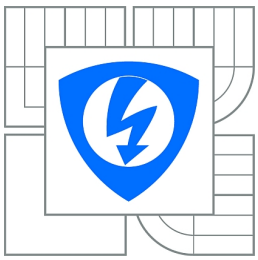
AUTOR PRÁCE — Bc. MARTIN JANČÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ZBYNĚK FEDRA, Ph.D.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student:	Bc. Martin Jančík	***ID:***	78512
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Dynamická rekonfigurace s Atmel FPSLIC

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s platformou Atmel FSSLIC a vývojovým kitem ATSTK 594. Prostudujte způsob programování daného zařízení jak pro část FPGA (jazyk VHDL) tak pro AVR (assembler, C). Vytvořte jednoduchou aplikaci využívající obě části platformy a ověřte předávání hodnot a provázání programu mezi platformami. Prostudujte možnost dynamické rekonfigurace hradlové části.

Pokuste se realizovat jednoduchou ukázku rekonfigurace. Otestujte možnosti a omezení dané platformu a sestavte manuál pro využití dynamické rekonfigurace na dané platformě.

DOPORUČENÁ LITERATURA:

[1] AT94K series field programmable system level integrated circuits. Atmel, 2002.

[2] SEKANINA, L. Evolvable components. From theory to hardware implementations. Berlin: Springer, 2004.

Termín zadání:	8.2.2010	***Termín odevzdání:***	21.5.2010

Vedoucí práce: Ing. Zbyněk Fedra, Ph.D.

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Bc. Martin Jančík
Bytem: Podolí 336, Podolí, 664 03
Narozen/a (datum a místo): 2. dubna 1986 v Brně

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

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

Čl. 1

Specifikace školního díla

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

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

   Název VŠKP: Dynamická rekonfigurace s Atmel FPSLIC
   Vedoucí/ školitel VŠKP: Ing. Zbyněk Fedra, Ph.D.
   Ústav: Ústav radioelektroniky
   Datum obhajoby VŠKP: _____

   VŠKP odevzdal autor nabyvateli*:

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

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

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

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

* hodící se zaškrtněte

Článek 2

Udělení licenčního oprávnění

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

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

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

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

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

Článek 3

Závěrečná ustanovení

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

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

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

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

V Brně dne: 21. května 2010

…….. ……

Nabyvatel Autor

ABSTRAKT

Tato práce popisuje platformu Atmel FPSLIC, která je tvořeny pomocí hradlových polí FPGA a mikrořadiče AVR. Je zde i popsán vývojový kit STK594 a jeho možnosti programování, ať hradlového pole FPGA, tak také mikrořadiče AVR. Také je popsán samotný obvod AT94K. Tento obvod je možné programovat jazykem VHDL (pole FPGA) nebo pomocí assembleru a jazyka C pro mikrořadič. Toto vše umožňuje sloučit do jednoho výstupního souboru pomocí programu System Designer, který obsahuje sadu softwarových nástrojů pro dané programové jazyky a generování celého obvodu. Práce dále popisuje jednoduchou aplikaci pro obě části platformy. Také je zde věnováno popisu dynamické rekonfigurace hradlové části tohoto obvodu.

KLÍČOVÁ SLOVA

ATMEL FPSLIC, Dynamická rekonfigurace, AT94K, VHDL, assembler

ABSTRACT

This study describes the platform Atmel FPSLIC, which is created by means of the logic arrays FPGA and the micro-sequencer controller AVR.

The developmental kit STK594 is described here as well, with its programming possibilities, as for the logic arrays FPGA, as for the micro-sequencers AVR. Also the separate circuit AT94K is described there. This circuit can be programmed by the language VHDL (the field FPGA), or by means of the assembler and language C for the micro-sequencer. All this can be integrated into the one output file by means of program System Designer, comprising a set of software tools for given programming languages and for generation of the whole circuit. Furthermore, the study describes a simple application for the both platform parts. Also the description of the dynamic reconfiguration of the circuit gate part is included.

KEYWORDS

ATMEL FPSLIC, Dynamic reconfiguration, AT94K, VHDL, assembler

JANČÍK. M. Dynamická rekonfigurace s Atmel FPSLIC. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2010. 42 s. Diplomová práce. Vedoucí práce: Ing. Zbyněk Fedra, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Dynamická rekonfigurace s Atmel FPSLIC jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu semestrální práce Ing. Zbyňku Fedrovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Tato práce se zabývá problematikou dynamické rekonfigurace hradlových polí FPGA(Field Programmable Gate Array), které jsou integrovány na jednom čipu společně s mikrořadičem AVR. Firma Atmel, která se danou problematikou zabývá vytvořila platformu Atmel FPSLIC a této platformě je v této práci věnováno. Zkratka FPSLIC znamená Field Programmable System Level Integrated Circuits, což by se dalo volně přeložit jako obvody s integrovanými hradlovými poli. Tato práce je tvořena na vývojovém kitu STK 594, kde je umístěn jeden obvod AT94K. Obvody AT94K a AT 94S (S značí verzi secure neboli zabezpečený) jsou tvořeny několika základními bloky. Jedním z bloků je programovatelné hradlové pole FPGA, další část obvodu je 8 bitový MCU rodiny AVR a paměť typu SRAM. Obvod je doplněno o množství periférií, díky kterým lze vytvářet různé sofistikované zapojení.

Obecně lze říci, že aplikace postavené na dynamické rekonfiguraci mají složitější strukturu než klasické statické aplikace. Úkolem dynamické rekonfigurace není jen vykonávání předepsaných operací, ale také řízení vlastní rekonfigurace a ošetření dalších jevů, které vzniknou danou rekonfigurací. Jde hlavně o správu veškerých rozhraní a stavových informací částí obvodu, jež nejsou uvnitř FPGA pole. O tuto činnost se většinou stará řídící obvod, v tomto případě mikrokontrolér AVR. Tato část obecně může být umístěna přímo na jednom čipu, jako je v tomto případě, nebo také připojený externě.

V této práci je dále popsán daný obvod AT94K a jeho periférie. Vývojový systém, System Designer, pomocí kterého jsou tvořeny zdrojové kódy pro obě části této platformy. Výsledná aplikace bude zkonstruovaná v jazyce VHDL(Very high speed circuits Hardware Description Language) a programovacím jazyce assembler.

Dále se práce bude zabývat praktickou ukázkou dynamické rekonfigurace, vlastnostmi při předávání informací mezi jednotlivými platformami. Výstupem celého snažení v této práci bude jednoduchá ukázka dynamické rekonfigurace společně s testováním a zkoušením jednotlivých omezení platformy FPSLIC.

Možná panuje i otázka proč využít tuto platformu. Není příliš vhodná pro rychlé zpracování velkých objemů dat. Avšak výhodou je, že téměř o 50% sníží oblast využití na desce plošného spoje. Mezi další výhody patří způsob programování, který je ISP(In System Programmable) což značné ulehčuje případný upgrade zdrojových kódů v cílovém obvodě. Navíc disponuje flexibilním rozhraní a také simulace pomocí softwarových nástrojů i případné simulace na vývojových deskách.

1 DYNAMICKÁ REKONFIGURACE

Rozeznáváme několik hledisek podle nichž můžeme rozdělit rekonfigurace v obvodech FPGA. Rekonfigurace se dá rozdělit na dynamickou a statickou.

→ Statické rekonfigurace znamená přeprogramování obvodu během času kdy obvod nepracuje tedy nevykonává žádné instrukce.

→ Dynamickou rekonfigurací rozumíme přeprogramování obvodu při právě vykonávané operaci, nebo alespoň část obvodu provádí svoji činnost.

Další možností rozdělení dynamické rekonfigurace je způsob provedené rekonfigurace. První možností je úplná rekonfigurace, kdy se mění konfigurace celého obvodu najednou. Druhou možností je částečná rekonfigurace, kdy se nemění celá funkce obvodu, ale jen jeho část bez toho, že by byl ovlivněn zbytek obvodu. Rekonfiguraci lze s výhodou použít v systémech kde není možné provést odstranění poruch pomocí obsluhy přímo v místě kde se nachází zařízení.

Využití rekonfigurace lze členit do čtyř základních oblastí:

• upgrade firmware

• zvyšování spolehlivosti

• zvyšování funkční hustoty

• sebeadaptaci systému na základě vnějších podnětů

Typickým zástupcem statické a úplné rekonfigurace je upgrade firmware obvodu. Systém ještě za běhu programu načte novou konfiguraci do konfigurační paměti. Obvod po vypnutí a následném zapnutí načte z paměti již novou konfiguraci celého zařízení.

Nejvíce využívanou rekonfigurací je zvyšování funkční hustoty. Zjednodušeně řečeno jde o provedení složitějšího výpočtu na menší ploše čipu. Myšlenka, na které je toto postaveno je rozdělení aplikace na menší celky, které se postupně střídají a provádějí výpočty pouze když jsou potřeba. K tomuto účelu lze použít částečnou nebo i úplnou rekonfiguraci obvodu. Navrhování tohoto způsobu rekonfigurace je složité, a také nemusí být vhodný pro všechny aplikace. Výhodou je po tomto návrhu menší spotřeba obvodu a rozměrů celého výsledného systému.

Metoda sebeadaptace je velmi populární. Jde o metodu kdy obvod reaguje na vstupní podměty, vyhodnocuje vstupní informace a podle určitých kritérií generuje výstupy a zároveň provádí optimalizaci vnitřní struktury. Tohoto způsobu se hojně využívá v oblasti kompresních algoritmů.

1.1 Princip dynamické rekonfigurace

Většinou rekonfigurovatelné aplikace jsou o poznání složitější než obyčejné statické aplikace. Úkolem není jen provedení potřebných výpočtů, ale zároveň je nutné řídit vlastní rekonfiguraci a ošetření další jevů vzniklých při rekonfiguraci. Prakticky jde o správu rozhraní FPGA/AVR a stavových informací uvnitř FPGA obvodu. Při spouštění systému nejprve dojde k nahrání úvodní konfigurace z permanentní paměti a úvodním spuštění obvodu. Kontrolér sleduje systém a na základě předem daných podmětů provádí výměnu dynamických modulů. Je-li splněna podmínka pro rekonfiguraci, konfigurační kontrolér (AVR procesor) zastaví činnost FPGA pole, zařídí uložení vnitřního stavu, uloží výstupní hodnoty a navrátí část FPGA obsazenou daným modulem do základního stavu. Pro nahrávání dynamického modulu je situace obdobná. AVR procesor nejprve modifikuje potřebné části FPGA obvodu konfiguračními daty daného modulu pomocí uložených stavových informací. Veškeré operace potřebují přesné časování, aby nedocházelo k nechtěným modifikacím aplikací.

Hlavním konceptem rekonfigurační technologie je užití dynamických logických modulů, které nemusí být vždy přímo v obvodu FPGA. Tyto moduly jsou však nahrávány, případně stahovány v čase, vždy kdy je nutné nahrát odlišné dynamické moduly, pro změnu konfigurace.

Nástroje většinou umožňují dva způsoby dynamické rekonfigurace:

- moduly se stejnými rozhraními, jaké jsou používány v předchozí konfiguraci FPGA obvod - supermakra
- a moduly, které jsou staženy z obvodu a uvolní tak místo nahrání nové konfigurace – dynamické moduly

Nahrávání a stahování dynamických modulů má časové omezení, proto jsou specifikovány potřebné informace v dynamic constraints file(DCF) založené na podobném formátu vytvořeném v Berkley BLIF(Berkley logic interchange format) formát. Nástroj Figaro potom načítá DCF soubory a také EDIF netlisty, tak že korespondují s hlavním designem. Podmínkou pro nahrávání i stahování bitstreamu je správné řízení pomocí hodinového signálu a kontrolních signálů.

S dynamickými moduly je zacházeno jak s normálními makry s novou charakteristikou. Mohou být nahrávány a stahovány na pevné místo v FPGA poli bez jakéhokoliv narušení ostatních designů, jež se už nacházejí v hradlové části pole FPGA.

Supermakra mají jednu zvláštnost. Ačkoliv funkce jiných modulů je odlišná, jejich vstupní a výstupní porty jsou stejné. Totéž platí i o umístění v tom samém místě, na dynamicky nekonfigurovatelnou část pole. Supermakro je ze základu několik maker vložených do jednoho bloku tzv. blackboxu. Tento blok pak obsahuje několik funkcí. V daný čas je vždy používána jen funkce, která je potřeba.

V potřebný čas je nahrána funkce supermakra a nebo funkce dynamického modulu do FPGA.

Supermakro je výhodnější než užití dynamického modulu. Pro umístění dynamických modulů je nutné hledat vždy správné vhodné místo pro umístění. Tento proces pro supermakro je ojedinělý, stačí nalézt pouze jednou vhodné umístění a

následně si toto umístění rezervovat pro další použití. Jakmile je jednou umístěn dynamický modul, je nutné pro vyhledání vhodné pozice hodně času, dynamická rekonfigurace není tak „pružná“. Umístění dynamických modulů není vždy na stejné místo v hradlovém poli.

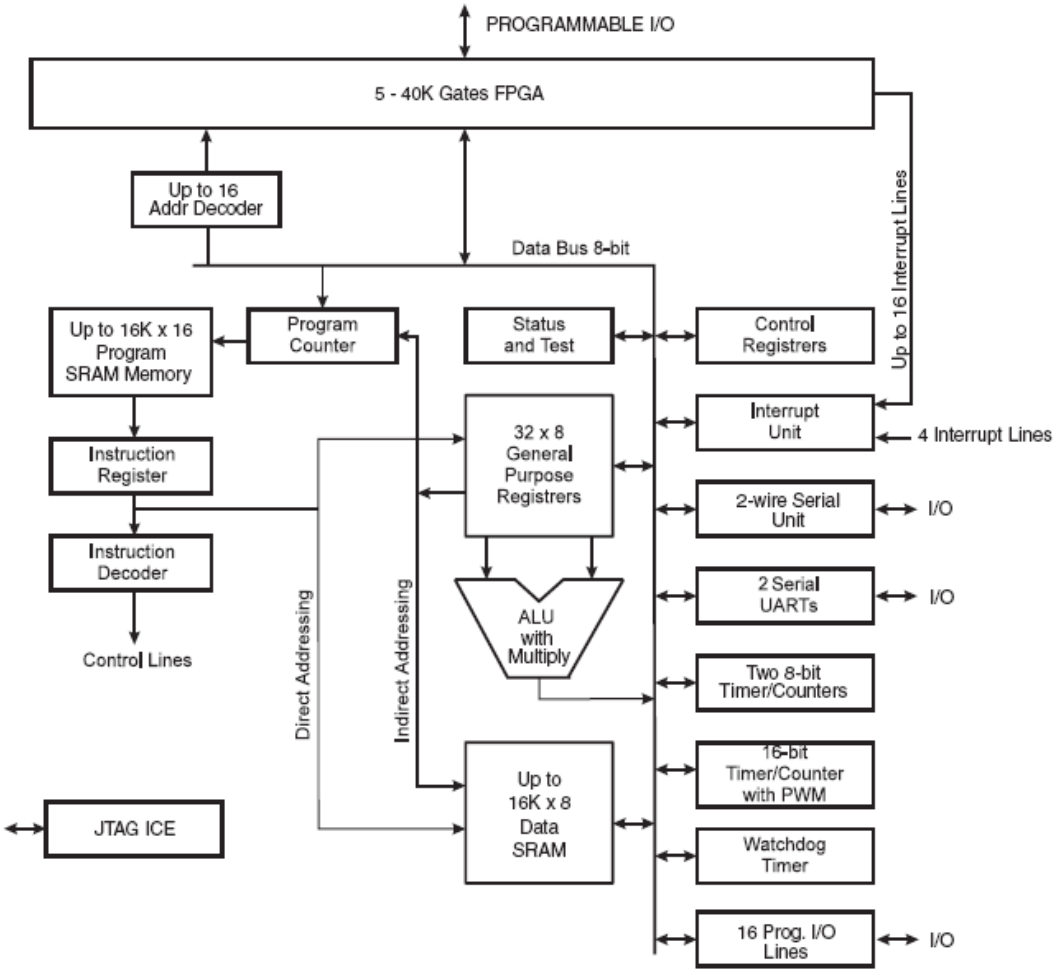
2 OBVOD AT94K

Obvod Atmel AT94K je složen z několika částí jak již bylo zmíněno. Na Obrázku 1.1 je zobrazena bloková struktura toho obvodu. Z blokového schématu je patrné, že obvod je složen ze dvou hlavních částí. Jednou dominantou je programovatelné pole FPGA, spojené s blokovou strukturou jádra riscového mikrořadič AVR. Tento řadič je mírně modifikován oproti obyčejnému AVR mikrokontroléru. U tohoto obvodu je použito hradlové pole AT40K s 5 až 40 tisíci hradel jež je také patentem firmy Atmel.

Další možností rozšíření jsou sériové konfigurační paměti ATFSxx nebo obvodové řady Secure AT94S. Paměť řady secure nabízí daleko větší bezpečnost proti reverznímu engeneeringu proti vyčtení zdrojového programového kódu z FPGA a AVR.

Obvod AT94K je vhodný pro realizaci jednoduchých úloh, které nevyžadují rychlé zpracování ani propustnosti velkého množství dat.

Pro tuto platformu je zveřejněno i několik knihoven pro jednodušší práci jako například pro ovládání LCD displeje, ADPCM modulaci, převod modulací od PCM k PWM a jiné.



Obrázek 2-1: Architektura obvodu Atmel AT94K FPSLIC, (převzato z [1]).

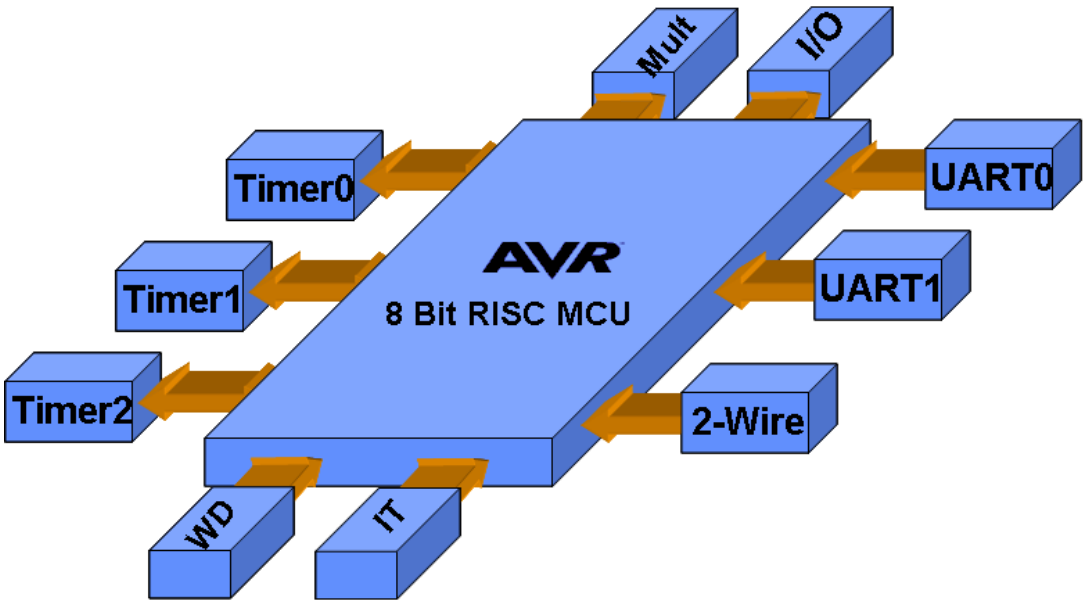
2.1 Jádro mikrořadiče AVR a jeho periférie

V obvodech FPSLIC je začleněno AVR jádro riscového 8 bitového mikrořadiče. Tento mikrořadič je vytvořen technologií CMOS low-power, pro co nejmenší spotřebu dosahuje výpočetního výkonu až 1 MIPS na 1 MHz. AVR jádro založené na rozšířené architektuře RISC kombinuje bohatý instrukční soubor i s instrukcemi trvajícími pouze jeden hodinový takt s 32 pracovními registry. Tyto registry jsou spojeny s aritmeticko logickou jednotkou (ALU), která dovoluje ovládat další dva nezávislé registry, které jsou přístupné pro instrukce trvající pouze jeden hodinový cyklus.

Jádro lze provozovat ve třech napájecích módech. V Idle módu je CPU zastaven. Jakmile pamět SRAM, čítač/časovač nebo přerušení jsou vyvolány potom CPU pokračuje v práci. V módu Power-down se uloží obsahy registrů a zastaví se činnost oscilátoru, který vyřadí ostatní funkce na čipu a čeká, dokud nedojde k přerušení nebo hardwarovému restartu. V posledním módu Power-save oscilátor pokračuje v práci, dovoluje uživateli udržovat čítač v základním stavu, jakmile dojde k signálu ke klidu jádro AVR se uspí.

Jádro obsahuje(zobrazeno na obrázku 2-2):

- obvod čítače watchdog s interním oscilátorem (WD)
- oscilátor s vnitřním časovacím obvodem
- 16 bitový čítač/časovač (Timer1)
- 2x 8 bitový čítač/časovač (Timer0, Timer2)
- hardwarovou násobičku (Mult)
- dvě jednotky UART (UART0, UART1)
- rozhraní 2-Wire
- I/O port D (8 bitový)
- I/O port E

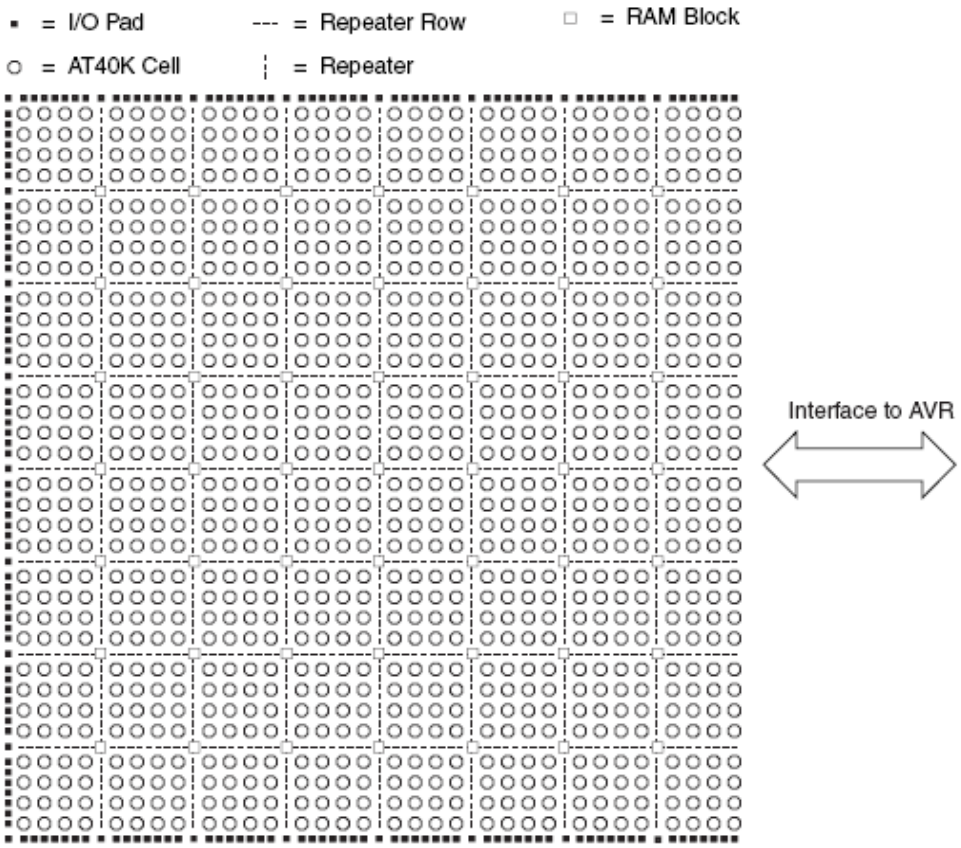


Obrázek 2-2: Struktura AVR části obvodu a periferie.

2.2 Jádro FPGA

V obvodech FPSLIC je použito obvodu AT40K. Do tohoto obvodu FPGA může být implementovány různé aplikace. Mezi nejpoužívanější aplikace, které se implementují jsou například adaptivní FIR (Finite Impulse Response) filtry, rychlá fourierova transformace, diskrétní kosinova transformace, dále se zde používají různé metody pro kompresi signálů, šifrování a jiné. V těchto obvodech je využito rychlé statické paměti SRAM s dobou přístupu cca 10ns. Toto programovatelné pole umožňuje i rychlý přístup do jednotlivých buněk obvodu a také podporu vektorové násobičky.

Srdcem celé platformy FPSLIC je symetrické programovatelné pole stejných buněk. V tomto poli je po každých čtyřech buňkách opakovač sběrnice. Ke každému vertikálním opakovači je přiložena jedna buňka paměti RAM. Tato paměť může být konfigurována jako jednoportová nebo dvouportová a může pracovat v synchronním nebo asynchronním režimu. Na Obrázku 2-3 je vidět FPGA sít buněk, které jsou součástí obvodu. Tyto plány polí mají 3 přístupové sběrnice: Lokální sběrnice – každý segment spojuje 4 buňky, které jsou připojeny na opakovače. Dále jsou to dvě expresní sběrnice, které spojují 8 buněk do jednoho segmentu a vyřadí opakovač signálu.



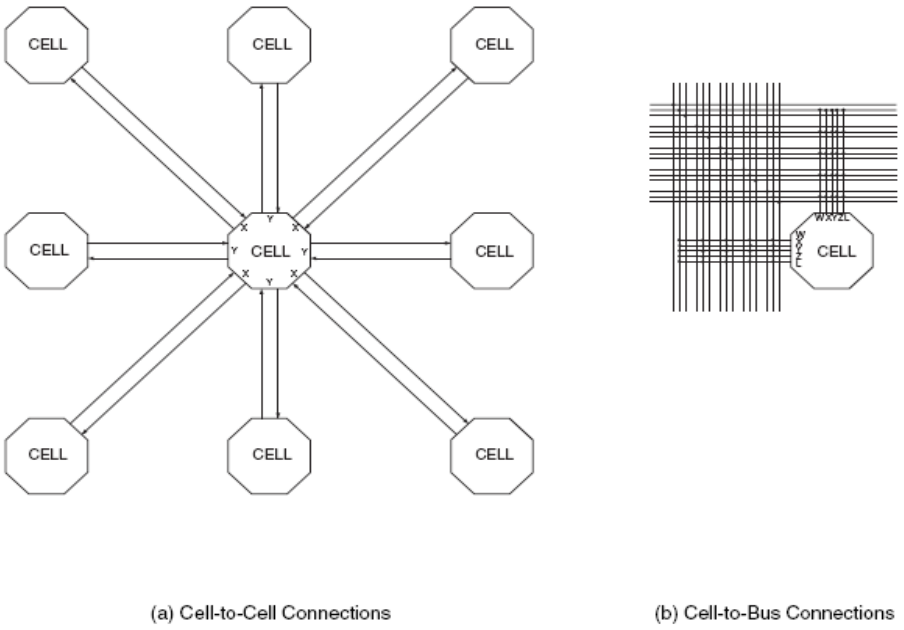
Obrázek 2-3: Pracovní síť buněk obvodu AT40k, (převzato z [2]).

V obvodech FPSLIC jsou dva druhy operační paměti RAM. Jedna tzv. „FreeRAM“ distribuovaná programovatelným polem FPGA a jedna SRAM paměť sdílená pro programovatelné pole i AVR mikrokontrolér.V poli FPGA je 32 x 4 dvouportová paměti RAM, která je připojena ke každému sektoru buněk.

V tomto obvodě je osm časovacích sběrnic. Šest z osmi sběrnic je spojeno a přivedeno na jeden globální časovací vstup, zbylé dvě sběrnice jsou řízeny hodinovým signálem řízeným jádrem AVR mikrokontroléru.

2.2.1 Struktura buněk FPSLIC

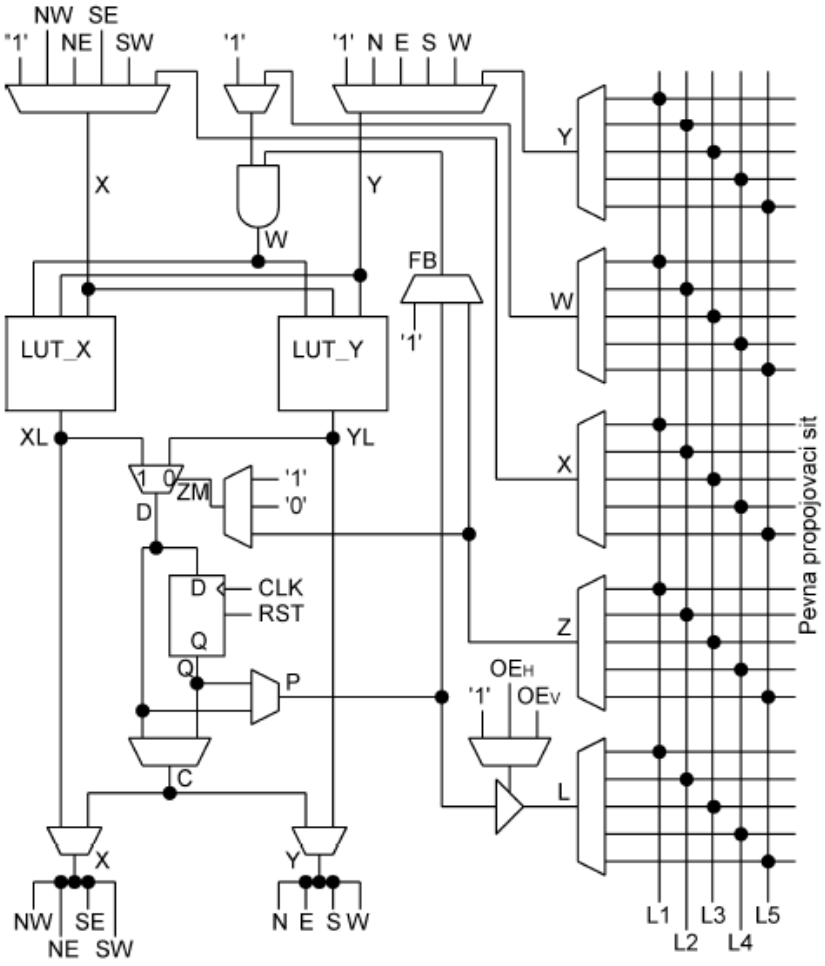
Na Obrázku 2-3 je zobrazena pracovní síť buněk v obvodu. V této kapitole je podrobněji popsána struktura těchto buněk.



Obrázek 2-4: Struktura propojení jednotlivých buněk, (převzato z [2]).

Pro zjednodušení popisu jsou zde v popisu vynechány anomálie u krajních buněk. Mezi každými buňkami jsou dva horizontální spoje. Jeden spoj slouží jako vstupní a druhý jako výstupní. Toto ukazuje Obrázek 2-4 na pravé straně. Na levé straně Obrázku 2-4 jsou zobrazeny vertikální sběrnice a horizontální sběrnice. V místě kde dochází ke křížení těchto sběrnic se nachází každá jednotlivá buňka. Sběrnice má šířku pěti bitů.

Z Obrázku 2-5 je patrné, že výstup signálu je zapojen vždy, záleží pouze na příjemci (přijímající buňce), jestli tento signál použije. Zvolí-li přijímací buňka, záleží, z které buňky použije výstupní hodnotu. Přijímat signál ze sousedních buněk je možné i ze dvou buněk současně. Je nutné dodržet pravidlo, že jeden přijímaný signál musí být namapován na vstup X a ten druhý na vstup Y. V případě signálu ze sousedních buněk nelze na již nadefinovaném mapování nic měnit. Zda přijde signál na vstup Y a nebo na vstup X je pevně stanoveno vlastní polohou sousedních buněk. Pro jednodušší popis propojení jednotlivých buněk bude využito světových stran. Spojení přímo ze severu na jih či východu na západ je trvale napojeno na vstup Y. Přímé propojení ze severovýchodu, jihovýchodu, jihozápadu a severozápadu jsou pak mapovány na vstup X. Toto propojení je zobrazeno na Obrázku 2-4.

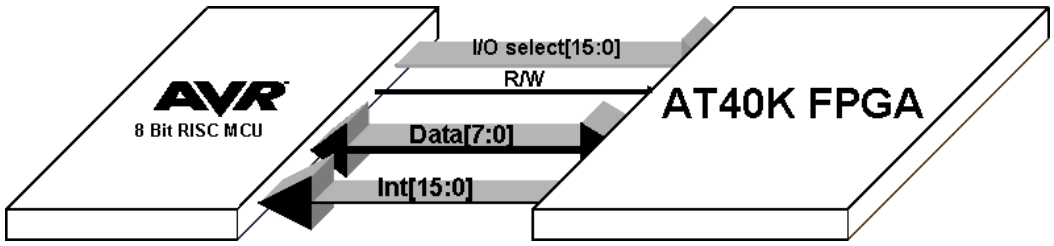


Obrázek 2-5: Zjednodušená struktura jedné buňky obvodu FPSLIC, (převzato z [5]).

Na Obrázku 2-5 je zobrazena struktura buňky. Nejdůležitějším místem celé buňky jsou dva 3vstupové jednotky LUT , označené LUTX a LUTY (přidané písmeno X nebo Y je řízeno podle toho, který signál daných jednotek LUT budí). Obě jednotky LUT mají vstupní vodiče X, Y a W a generují libovolnou funkci těchto 3 proměnných na signálech XL, resp. YL. Zapojení do nejčastěji používané funkce 4 vstupové jednotky LUT se děje pomocí multiplexoru ZM, který lze ovládat čtvrtým signálem Z, místo statického přiřazení některým z konfiguračních bitů, jak je to u všech ostatních multiplexorů v tomto schématu[5].

Každá buňka umí pracovat v několika režimech. Například v režimu aritmetické, režimu čítače a dalších. Více je uvedeno v dokumentaci k obvodu [2].

2.3 AVR/FPGA rozhraní a systém řízení



Obrázek 2-6: AVR/FPGA rozhraní, (převzato z [2]).

Komunikace mezi mikrokontrolérem a hradlovým polem je tvořena několika sběrnicemi viz obrázek 2-6. Hradlové pole generuje až šestnáct přerušení. Obousměrná komunikace je tvořena jednou osmi bitovou datovou sběrnicí. Šestnáct vstupně/výstupních vodičů řídí přístup do čtyř pamětí. V paměti SRAM jsou uloženy programové instrukce pro mikrokontrolér AVR. Tato paměť je ještě rozdělena na dvě části. K jedné části přistupuje AVR mikrořadič a k druhé části má přístup FPGA pole.

Pro řízení obvodu máme několik možností. První možností řízení je vyvolání pomocí externího restartu. V obvodu AT94K jsou hned dva resetovaní vstupy. Oba dva jsou řízeny pomocí nízké úrovně. Jeden restartovací vstup smaže celou konfiguraci FPGA z paměti SRAM a vynuluje i systémový kontrolní registr. Další vstup slouží pro restartování AVR mikrořadiče.

Tento obvod umí pracovat ve čtyřech režimech. Dva z těchto režimů jsou rezervované pro tovární testování. Jeden z těchto módů se nazývá „Master Serial“ a druhý „ Slave cascade“.

AT94K má 8 bitový stavový registr, kde se ukazují informace a stavech celého systému. V tento registr neslouží jen pro informace, ale i pro nastavení a povolení periférií a také restů, přerušení apod.

- obě části platformy sdílí duální port pro přístup do paměti SRAM s přístupem 15ns
- až 16 přerušení od FPGA do jádra AVR
- až 16 adresných linek přímo do jádra FPGA
- 8 bitová datová propojovací linka z FPGA do AVR

2.4 RAM

Obvody FPSLIC v sobě obsahují dva druhy pamětí RAM. Jedním druhem je volná paměť RAM přímo v jádru FPGA pole jak je zobrazeno na obrázku 2-3. Paměťové bloky jsou rozprostřeny v celém hradlovém poli. Druhou pamětí je paměť typu SRAM sdílená mezi jádrem AVR a hradlovým polem FPGA.

Paměť SRAM je vytvořena tak, aby do ní mohly přistupovat obě části, s přístupovou dobou 15ns. Tuto paměť i využívá jádro AVR pro uložení jeho programové instrukce hodnoty do důležitých registrů. Obvody AT94K10 a AT94K40 má rozdělenou paměť SRAM do tří bloků. V jednom bloku 10kB x 16 je vyhrazeno pro

programovou paměť. Druhý blok paměti o velikosti 4kB x 8 je vyhrazeno pro datovou část. V posledním třetím bloku je konfigurovatelná paměť, která rozdělena na 6kB x 16 popřípadě 12kB x 8, která může být využita pro uložení programových instrukcí nebo dat, dle potřeby. Obvod AT94K05 má paměť SRAM rozdělenou poněkud odlišně. Opět je rozdělena do tří bloků, na paměti programovou, datovou a konfigurovatelnou, pouze v jiném poměru. Pro programovou paměť je určeno 4kB x 16, pro datovou paměť 4kB x 8 a zbytek paměti o velikosti 2kB x 16 popřípadě 4kB x 8 pro konfigurovatelnou paměť.

2.5 Pracovní módy obvodu AT94K

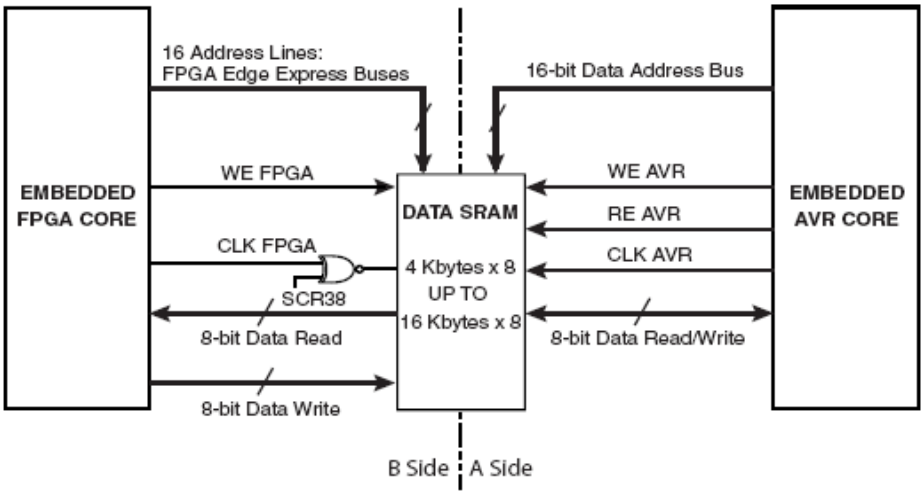
Tento obvod podporuje několik způsobů podle jakých umí pracovat. Jedním pracovním režimem je FPGA Frame Mode a druhý režim se nazývá AVR Cache Mode.

2.5.1 FPGA Frame Mode

FPGA obvod používá logicky přístup do paměti SRAM přes rozhraní B viz obrázek 2-7 . Tato paměť má dvě rozhraní, jedno je přímo spojeno s částí FPGA a jedno rozhraní je spojeno s procesorem AVR. Na straně B(rozhraní FPGA – SRAM) je možné zakázat čtení pomocí registru SCR 63. Kontrola celého rozhraní je implementována ve vývojovém prostředí System Designeru v Interface Conection.

Rozhraní A zprostředkovává komunikaci s pamětí SRAM a AVR jádrem obvodu. Propojení je vytvořeno 16 bitovou adresní sběrnicí s 8 bitovou datovou sběrnicí pro čtení nebo zápis. Vše je prováděno synchronně s časováním AVR procesoru. Jde-li o čtení nebo zápis je voleno pomocí dalších řídících signálů.

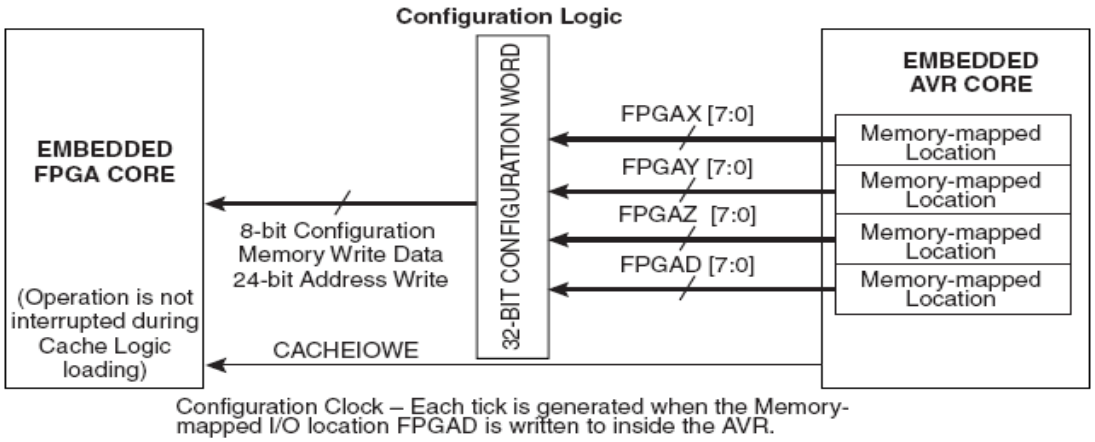
B rozhraní je spojeno s částí FPGA. Opět pro adresaci je použita 16 bitová adresní sběrnice (FPGA Edge Express)a dvě sběrnice pro datovou komunikaci. Jedna 8 bitová sběrnice je pro načítání dat z paměti SRAM a druhá, taktéž 8 bitová, pro zápis dat z hradlového pole do paměti SRAM s jedním řídícím registrem. Přístup do paměti je omezen v tzv. debug modu. Data z AVR jsou mapována do části FPGA na stejné adresy.



Obrázek 2-7: Bloková struktura FPGA Frame Modu (převzato z [2]).

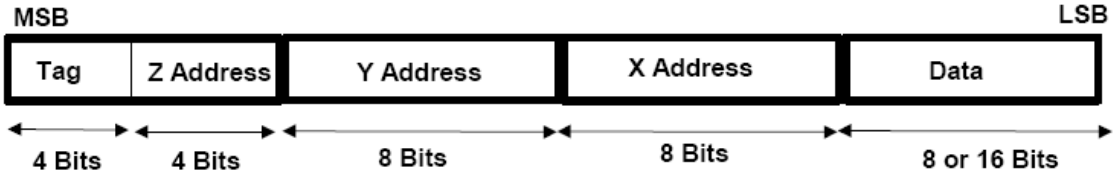
2.5.2 AVR Cache Mode

V tomto módu je možné přímo provádět modifikaci FPGA části obvodu za chodu jiné aplikace. AVR procesor má možnost přímo přistoupit do konfigurační paměti SRAM části FPGA, kde je uložen bitstream. Během běžného downloadu není možné používat tuto metodu. Port používaný pro cache mode jsou přímo ve vstupně výstupních registrech obvodu procesoru. Celá rekonfigurace je řízena pomocí několika registrů. Registry FPGAX, FPGAY, FPGAZ určují adresu bitstreamu uvnitř FPGA. Tyto registry jsou namapovány do posledních registrů viz obrázek 2-10. Registr FPGAD je uložen v jádře AVR a kontroluje případně zapisuje data. O hodnoty registrů FPGAZ a FPGAD je nutné požádat přímo výrobce obvodu firmu Atmel. Dostupnost hodnot těchto registrů je pouze pod smlouvou NDA (Non-disclosure Agreement). Tímto výrobce chrání know-how programátorů uložené v hradlové části. Struktura komunikace je na obrázku 2-8. Pomocí registrů FPGAX, FPGAY, FPGAZ a FPGAD se vytvoří 32 popřípadě 40 bitové konfigurační slovo, jehož struktura je označena na obrázku 2-9. Toto konfigurační slovo je pak následně presentováno dále s řídícím impulsem CACHEIOWE.

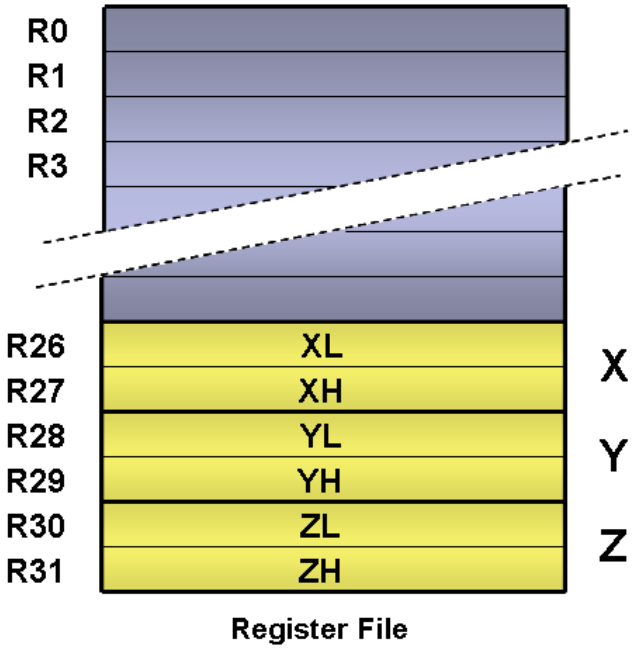


Obrázek 2-8: Bloková struktura AVR Cache Modu, (převzato z [2]).

Na obrázku 2-9 je zobrazen komunikační rámec v Cache pracovním režimu obvodu FPSLIC. Tento rámec je složen ze 24 bitové adresy a 8 nebo 16 bitové datové struktury. Rámce je přenesen pomocí jednoho hodinového cyklu.



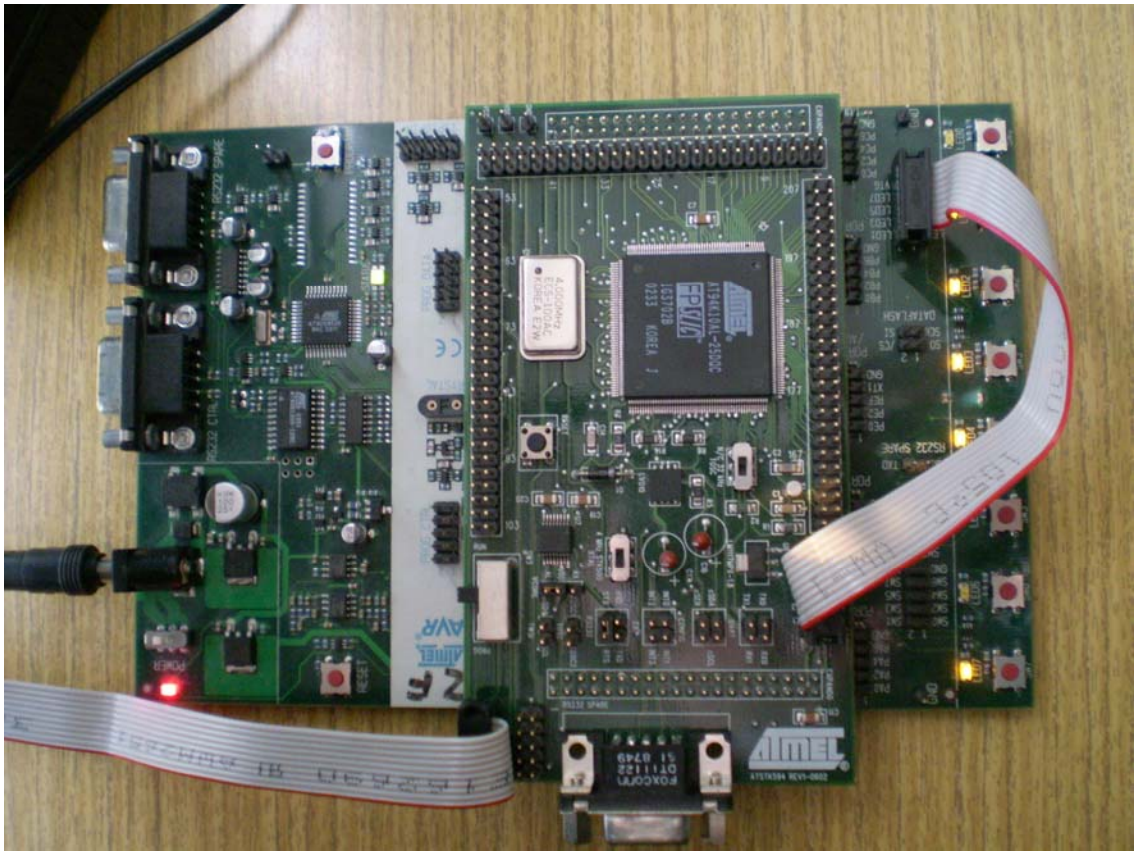
Obrázek 2-9: Struktura rámce v Cache modu obvodu FPSLIC, (převzato z [7]).



Obrázek 2-10: Struktura registrů AVR jádra obvodu, (převzato z [7]).

3 VÝVOJOVÝ KIT STK 594

Tento vývojový kit je vytvořen jako nadstavba na vývojovou desku STK 500, která byla vytvořena pro testování aplikací osmi bitových mikrokontrolerů Atmel AVR. Samotná deska se připojuje pomocí konektorů k vývojovému kitu STK500. Celý kit je napájen pomocí desky STK500 viz obrázek 3-1. Pro připojení se využívá propojení pomocí rozšiřujícího rozhraní, které je na desce STK 500. Je důležité dodržet orientaci desky, jak zobrazuje obrázek 3-1.



Obrázek 3-1: Vývojový kit STK 594

3.1 Rozhraní vývojového kitu STK594

Jedním z rozhraní jsou vstupně výstupní porty. Porty A až D jsou přímo na základní desce STK 500. Po zasunutí vývojového kitu je téměř nemožné se připojit jakékoliv konektory ke spodní desce STK500. Nejsou duplikovány přímo na vývojovém kitu STK594. Port E je dostupný pouze na kitu STK594.

Další rozhraní na vývojovém kitu je rozhraní pro programování pomocí ISP. Přes toto rozhraní se programují paměti SRAM u obvodu FPGA a AVR. Do zařízení se nahraje bitstream, kde jsou uloženy potřebné informace generované ze System Designeru.

Tento obvod umožňuje i programování a ladění pomocí rozhraní JTAG. Toto rozhraní je přímo integrováno v obvodě AT94K. Pro ladění a programování je nutné

použít speciální programovací kabel určení pro rozhraní JTAG.

Tento vývojový kit má dvě rozhraní UART. Vývody toho rozhraní jsou přístupné jako pinové pole, a také k jednotce UART je možno se připojovat pomocí klasického RS 232 C konektoru 9 pinového konektoru cannon.

Rovněž nechybí ani rozhraní TWSI (Two-Wire Serial Interface). Také nechybí připojovací rozhraní pro externí přerušení, možnosti nastavení hodnoty jádra obvodu AT94K a jiné.Součástí vývojového kitu jsou vstupy pomocí nichž lze vyvolat externí přerušení.

3.2 Ovládání a nastavení vývojového kitu STK594

Nezbytnou součástí vývojového kitu je také tlačítko pro resetování obvodu. Po jeho stisku je celé zařízení znovuspuštěno a je opět nahrána inicializace z konfigurační paměti. Druhou možností restartování je možnost restartovat procesor AVR přímo na základní desce kitu STK500. Tento spínač je označován jako AVRRESET. Po jeho stisku začíná běh programu procesoru od adresy $0000.

Důležitý je i přepínač, pomocí kterého je nastavováno zda má obvod AT94K pracovat nebo má-li být programován. Na vývojovém kitu jsou umístěny ještě další dva přepínače pomocí, kterých jsou nastavovány oscilátory. Jeden z nich umožňuje nastavení oscilátoru kdy je používán oscilátor přímo na hlavní desce(STK500), nebo zda se má využít oscilátor na desce STK594. Druhý přepínač nastavuje vnitřní oscilátor TOSC.

Pro napájení obvodu AT94K je nutné napájení 3,0 – 3,6 V. Pro připojení kitu STK594 je nutné připravit i vývojový kit STK500. Na tomto kitu je nutné nastavit napájecí napětí VTARGET na hodnotu udanou v datasheetu. Toto nastavení je prováděno pomocí propojek.

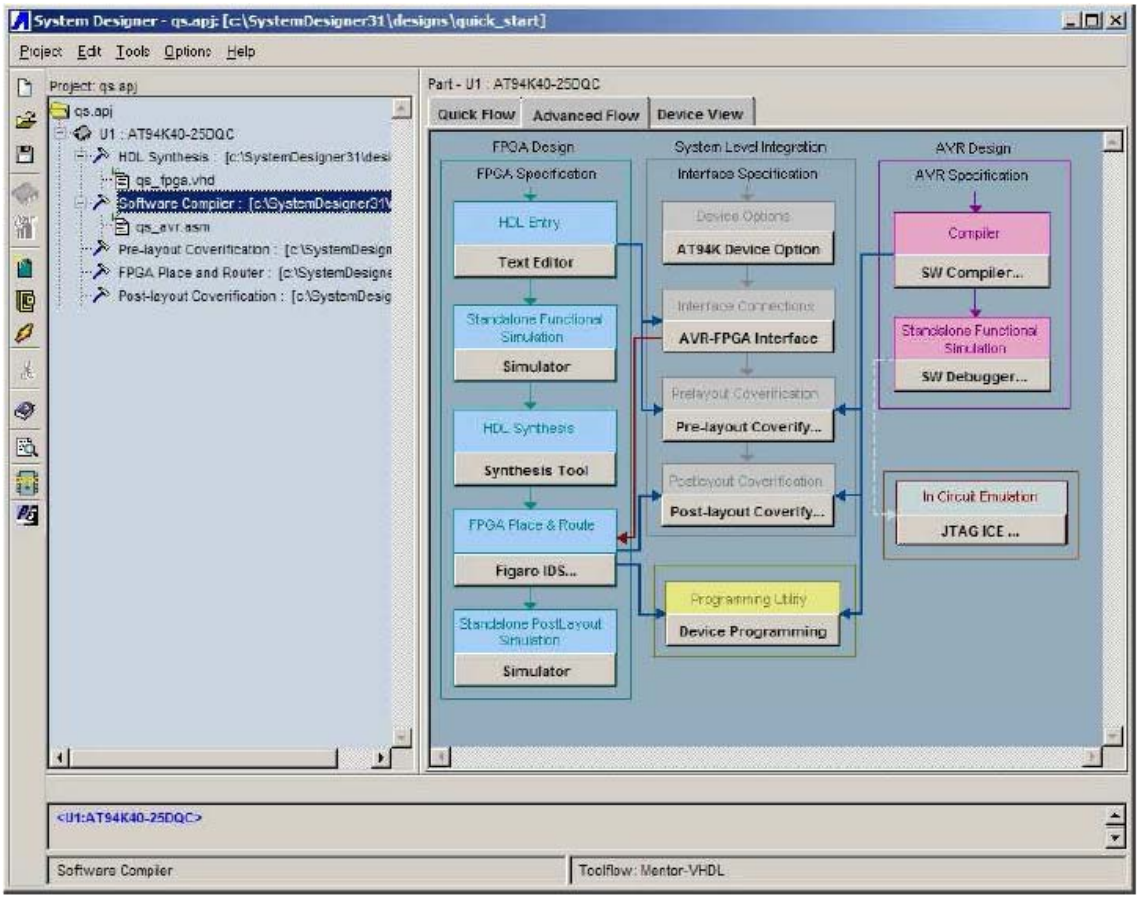
4 SYSTEM DESIGNER 3.1

V této kapitole bude popisován nástroj pro kompletní programování, syntézu a simulaci vytvořených zdrojových kódů pro obvody FPSLIC. Rovněž v tomto software je implementován nástroj pro programování obvodu AT94K.

Přístup k jednotlivým stupňům navrhování je možný pomocí tlačítek jak je patrné na obrázku 4-1 na pravé straně okna. Ovládání je možné také přes pravé tlačítko myši, které umožňuje někdy větší nabídku možností práce.

Ve spodní části okna je stavový řádek pomocí kterého je čitelné co se právě provádí, jsou zde zapisovány chyby a informace o provedených operacích, ukládá se z tohoto řádku tzv. logovací soubor.

Na levé okna na obrázku 4-1 je projektový manažer se zdrojovými kódy, pomocí něj lze také ovládat celý proces tvorby bitstream. Nevýhodou tohoto vývojového prostředí, že nemá podporu příkazové řádky pro všechny aplikace, a tak znemožňuje vytvořit automatické vytváření aplikací a následné nahrávání přímo do cílového obvodu.



Obrázek 4-1: Vývojové prostředí System Designer.

Systémové požadavky:

• Jednotka CD-ROM

• 250 MB paměti na pevném disku

• minimálně 128 MB operační paměti

• paralelní port

• síťová karta (pro zabezpečení licenční politiky)

• OS Windows 95/98/2000/Me/XP Windows NT 4.0

4.1 Nástroje pro programování AVR jádra

Bohužel vývoj nového software byl firmou Atmel pozastaven. Chybí i možnost provozování na novějším operačním systému Windows, poslední podporovaná verze Windows jsou Windows XP. Pro implementaci na jinou platformu, např. Linux popřípadě Apple Mac OS, neexistuje podpora od výrobce.

4.1.1 Wavrasm

Jedná se o nástroj, který pracuje pouze s programovacím jazykem assembler. Jde nejen o kompilátor z tohoto jazyka, ale i obsahuje textový editor, který umožňuje tvořit zdrojový kód. Tento nástroj vytvořila firma Atmel. Autorem je Tan Silliksaar. Grafické rozhranní není příliš propracované a umožňuje jen jednoduché úpravy zdrojových kódů a prohledávání textu.

Umožňuje vytvoření několika druhů výstupních formátů. Jedná se o formát všeobecný, formát Motorola S a nejpoužívanější formát Intel.

4.1.2 AVR Studio

Další nástroj pro programování jádra AVR obvodu AT94K a jiných osmi bitových mikrokontrolérů firmy Atmel.

Tento nástroj umožňuje programování pomocí jazyka C a také pomocí assembleru. Jako překladače je možné použít populárního open source gcc překladače. Platforma FPSLIC podporuje primárně dva komerční překladače. Jde o překladač IAR Systems C compiler a o překladač ImageCraft C compiler AVR.

Navíc ve vývojovém prostředí System Designer slouží jako simulátor zdrojových kódů pro část obvodu s mikroprocesorem. Rovněž je používán i při společné simulaci v Pre – Layout coverifikaci. Při simulaci je možné kontrolovat obsahy registrů, co je uloženo v paměti, kontrolovat stavový registr a periférie obvodu. Je možné sledovat výsledný kód v disassembleru. Simulátor umožňuje krokovat ve zdrojovém textu řádek po řádku, také krokovat přes jednotlivé funkce, případně z těchto funkcí přímo odejít. Samozřejmě nechybí možnost kdykoliv simulaci přerušit a možnost modifikovat obsahy registrů a tak i vyvolávat jednotlivé přerušení.

AVR Studio je dobře propracovaný nástroj, ať pro tvorbu zdrojových kódů nebo simulaci. Bohužel System Designer 3.1 využívá starší verzi tohoto programu. Implementace novější verze je velice problematická, ne-li nereálná. Propojení ostatních nástrojů a AVR Studia není příliš dokonalá. Simulaci je možné provádět jen ze zdrojového kódu v jazyku symbolických adres.

4.1.3 JTAG ICE

Tato platforma rovněž umožňuje simulaci a programování pomocí rozhranní JTAG. Rozhraní JTAG je specifikováno normou IEEE 1149.1. Toto rozhraní umožňuje komunikovat s platformou FPSLIC během ladění programu. Umožňuje skenování vnitřních periférií jako jsou přerušení, watchdog, zjišťovat obsahy registrů a jejich modifikaci apod. Výhodou této metody je možnost krokování programu přímo na čipu procesoru. Mikrokontrolér je možné přímo ovládat z AVR Studia. Paměť kódu programu lze zapisovat a číst přímo z aplikace (jedná se o speciální instrukce mikroprocesoru).

4.2 Nástroje pro programování FPGA části obvodu

4.2.1 HDL Planner

Jedná se o nástroj pomocí, kterého je vytvářen zdrojový kód v jazyku Mentor -VHDL a nebo Mentor -Verilog.

Při vytváření kódu má několik pomocných nástrojů, pomocí kterých si můžeme například definovat vstupní i výstupní proměnné, definovat si architekturu a jiné. Tento nástroj rovněž zvýrazňuje klíčová slova ve zdrojovém kódu. Také podporuje vkládání různých úseků vytvořených zdrojových kódů jako například vkládat procesy, příkaz case, různý druhy cyklů … Podporuje možnost vytváření vlastních maker. Pro rychlejší práci s velmi rozlehlými zdrojovými kódy jsou vytvořeny některé klávesové zkratky a ikony pro rychlejší přechody v textu.

4.2.2 ModelSim ATMEL 5.6

Jedná se nástroj od firmy Model Technology 2003 upraven pro firmu Atmel a potřeby pro programování Atmel FPSLIC. Je určen pro simulaci zdrojových kódů v jazyce VHDL nebo Verilog.

Bohužel je toto licencovaný nástroj a je nutné zakoupení licence pro práci s tímto simulátorem. Firma Mentro Graphics uvolnila jednu studentskou verzi programu ModelSim, pomocí, které by bylo možné simulovat výsledky napsaný ve VHDL jazyku popřípadě Verilogu. Avšak se nepodařilo tento nástroj importovat do vývojového systému System Designer. Licenci se nepodařilo také přenést. Licence je vázána na MAC adresu síťové karty.

4.2.3 Leonardo Spektrum

Tato část vývojového systému se stará o syntézu návrhu společně s Precision RTL Synthesis. Program bez problémů zvládá syntézu jazyku VHDL i Verilog. Tento program zvládá syntézu obvodů typu CPLD, FPGA i obvodů ASIC.

Rovněž ovládá i časovou analýzu a s licencí umožňuje náhled do vytvořených schémat zapojení. Pomocí tohoto nástroje jsou definovány i vývody s lze je měnit. Výstupním souborem jsou soubory s koncovkou *.edif (netlist) a soubor s příponou *.pin s konfigurací využití vývodů obvodu. Software je licencovaný firmou Mentor Graphics.

4.2.4 Precision RTL Synthesis

Jedná se o nástroj pro syntézu návrhu od firma Mentor Graphics. Syntézou rozumíme proces konverze systému z vyšší úrovně abstrakce na nižší úroveň abstrakce. V případě syntézy RTL hovoříme o konverzi kódu RTL HDL zapsaného pomocí syntetizovatelné podmnožiny jazyka do seznamu logických prvků a jejich vzájemného propojení. Bohužel neexistuje možnost vyzkoušení tohoto licencovaného nástroje, ani žádné trial verze programu není možné používat. Pro používání je nutné zakoupení licence. Nástroj je také výrobkem firmy Mentro Graphics. Ve vývojovém systému pro FPSLIC je použita verze 2006.a.

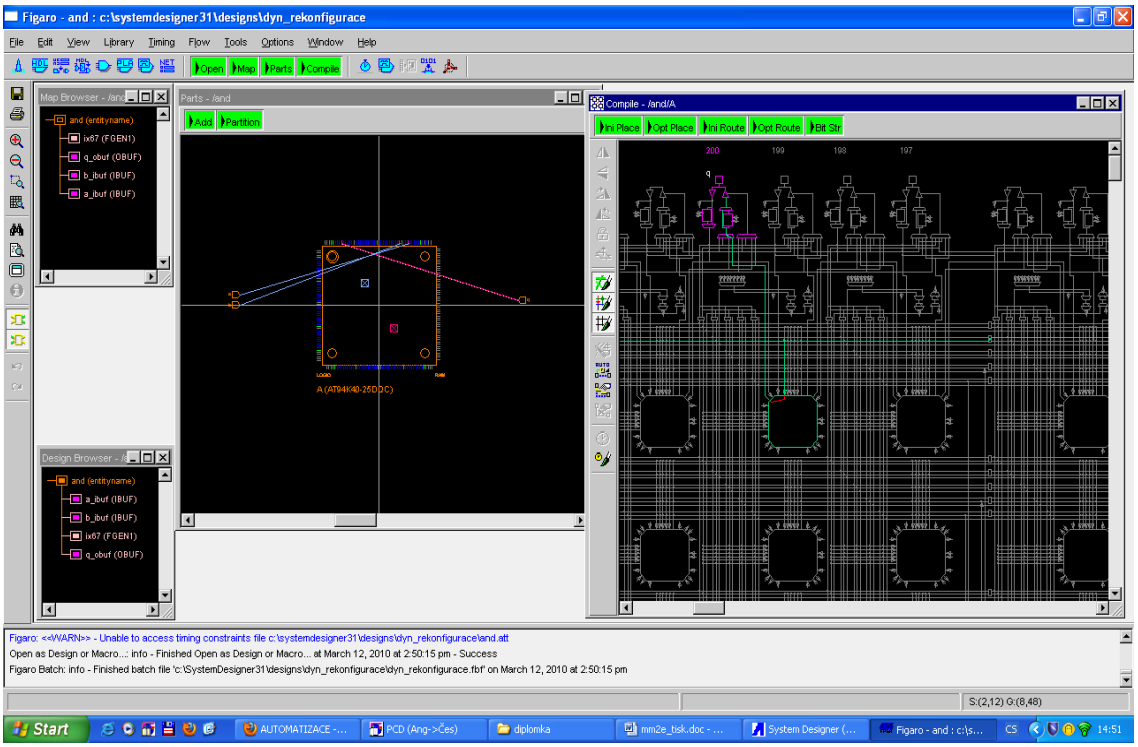
4.2.5 FIGARO IDS

Jedná se o nástroj pro tvorbu bitstreamů, které se po té nahrávají do cílového obvodu FPGA. Pracuje se vstupy netlistů typu EDIF. Také umožňuje vytváření vlastních maker, které lze implementovat do cílového obvodu. Umožňuje také „mapování“ do cílových obvodů. Mapováním je naznačena funkce procesu optimalizace logického designu a adaptování do obvodu s ATMEL architekturou. Zároveň se stará o to, aby byla fyzická implementace v pořádku. Práce s tímto nástrojem je náročná a skrývá spousty nebezpečí při špatné práci. Bohužel některé nestandardní chování lze eliminovat jen tím, že se dobře pročte manuál a následně se pustit do práce s tímto nástrojem. Pracovní okno toho nástroje je zobrazeno na obrázku 4-2.

Nástroj Figaro umí také zpracovávat vstupní data ViewLogic netlisty ve verzi 4.1 . Jde o formát dat s koncovkou *.WIR. Je-li tento netlist složený z mnoha oddělených souborů, jsou tyto soubory čteny pouze po jednom. ViewLogic je nástroj od firmy Synopsys. Tento nástroj produkuje tři výstupní soubory ve třech složkách. Tyto složky jsou nazývané SCH, SYM a WIR. Pro správné načtení nástrojem Figaro je dobré vytvořit složku do které je celý design nebo makro vloženo. Po té Figaro načítá automaticky soubory WIR a také soubory SYM, které také potřebuje pro správné načtení.

Pomocí programu Figaro, který umožňuje zpracovávání souborů pro statickou část vyvíjeného designu. Je zde možné vytvářet „mapování“ a zároveň vytvářet propojovací cesty mezi jednotlivými bloky PFGA obvodu. Tvorbou dynamických částí designu, jež jsou někdy nazývány dynamickými moduly, je nutné deklarovat ve VHDL entitách.

Entity jsou specifikovány jako VHDL příklady ve statické části. Každý příklad koresponduje s jedním tzv. d-modulem(dynamický modul). Každý tento modul je reprezentován jedním EDIF souborem.



Obrázek 4-2: Figaro IDS, vývoj designu.

Supermakro je složené z několika dynamických modulů se vstupy, které jsou zapojeny paralelně. O výstupní se stará zpravidla multiplexer. Každé supermakro je specifikováno jedním VHDL příkladem a je reprezentováno několika EDIF soubory, které mají definované vstupní/výstupní porty.

Statická část spolu s dynamickými moduly je instalována v tzv. blackboxu. Dalším krokem při tvorbě aplikace je přiřazení designu do knihovny, ve které budou uloženy všechny dynamické moduly. Následně je tato knihovna umístěna do statické části, je nalezeno fixní umístění a orientace pro každý dynamický modul. Výstupem z tohoto nástroje je několik bitstreamů, jeden pro statickou část a dva pro každý dynamický modul – jeden je pro nahrávání a jeden pro uložení.

4.2.6 Co-verification

Tento nástroj poskytuje možnosti simulace designu FPGA a AVR mikrokontrolér zároveň. V praxi to znamená, že se spustí zároveň dva simulátory se sdíleným rozhraní a tím se dosahuje možnosti simulace kompletního zdrojového kódu. Jen vždy na vstupním rozhraní může používat jen jeden simulátor. Spouští se zvláště simulátor pro část obvodu AVR, AVR Studio. Druhý simulátor se používá pro hradlové pole jedná se o nástroj ModelSim.

4.3 System Level Integration

V tomto vývojovém prostředí se stará o možnosti nastavení rozhraní mezi jádrem AVR a FPGA.

4.3.1 Device Options

Zde je možná nastavit velikost použité paměti SRAM pro celý obvod. Také se zde nastavuje zdroj hodinového signálu. Možnosti jsou tři. Buď použít AVR systémový čas, hodiny z čítače a také možnost ovládání pomocí watchdog obvodu. Rovněž se zde nastavuje na jakou hranu signálu budou zpracovávána data z paměti SRAM.

4.3.2 Interface Conection

Zde se nastavují propojení mezi jádrem AVR a FPGA částí. Pro tuto činnost je nutné mít v celém studiu nahraný zdrojový kód pro obvod FPGA a jazyce VHDL popřípadě Verilog. Po té jak je umístěn tento kód je možné definovat rozhraní pro komunikaci. Zde se také vybírá nejvyšší level entity. Po té je možné připojit signály z jádra AVR do této nejvyšší entity. V tomto úseku zpracování se také generuje vzorový kód pro testování tzv. testbench. Výstupem tohoto bloku je jednoduchý textový soubor s příponou *.ict.

4.3.3 Pre-layout coverify

V této části se stará vývojové prostředí o dokončení spojení a jeho následnou simulaci, už s oběma spuštěnými simulátory. Myšleno oběma simulátory jsou simulátor ModelSim a AVR Studio. V této fázi zpracování je možnost editovat vygenerovaný testovací soubor. Po případné editaci těchto testovacích vektorů je možné i vybrat hlavní testovanou entitu. Při simulaci běží oba simulátory zároveň, avšak pouze jeden z nich je možné ovládat a kontrolovat. Tohoto se dociluje tím, že se v AVR Studiu spustí režim simulace s automatickým krokováním, a následně se v ModelSim simulátoru pustí simulace a je možné převzít kontrolu nad AVR Studiem a v něm puštěným simulátorem. Simulace je zastavena z AVR Studia pomocí tlačítka „break“, a následným uzavřením AVR Studia a posléze simulátoru ModelSim. Tuto simulaci se nedoporučuje přeskočit. Jak je známo simulace obvodů a jejich funkčnosti patří mezi

nejdůležitější krok při navrhování celého funkčního systému.

4.3.4 Program Utility

Zde se nahrává tzv. Bitstream do cílového zařízení. Jako vstupních dat využívá bitstream generovaný z části FPGA a dále zdrojový kód ve formátu hexa souboru. Tato utilita umí generovat soubor pro celý systém FPSLIC a také pouze pro jednotlivé bloky, je tím myšleno jen pro jádro mikrokontrolér AVR a nebo obvodu FPGA.

Výstupem této utility je už přímo programování cílového obvodu. K tomuto kroku se používá Configurator Programming System tzv. CPS. Po té je možné nahrát už soubor do testovací desky popřípadě do jiného zařízení používané pro platformu FPSLIC.

4.4 Ostatní nástroje

V tomto vývojovém prostředí jsou integrovány i jiné nástroje, které nejsou součástí „stromu“ pro vývoj celé aplikace. Je možné je spouštět pomocí rychlých ikon na levé straně vývojového prostředí, nebo pomocí nabídky programu.

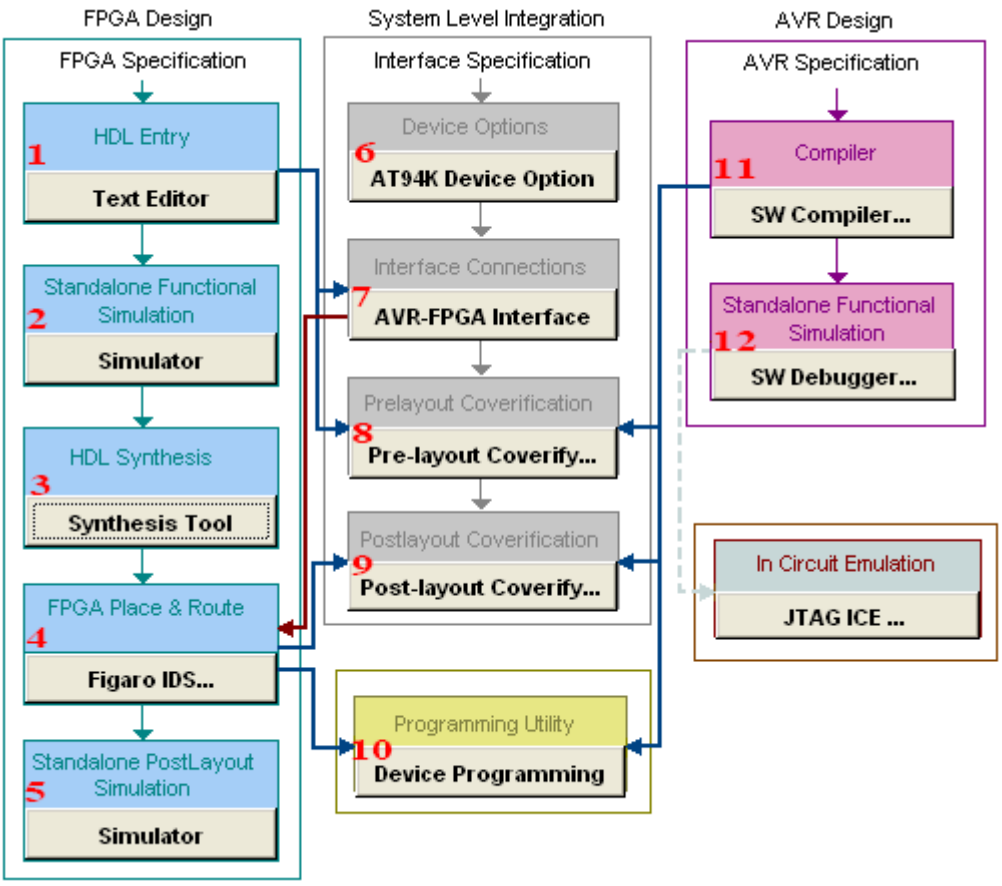
4.4.1 EasyPlanner

EasyPlanner je grafické uživatelské rozhraní pro spravování nástrojů pro komponenty z knihovny Atmel. Umožňuje i grafické možnosti sestavení designu v jazyce vhdl. Toto sestavení je jednoduché pomocí funkce drag & drop, tzn. jednoduché přetažení komponent z nabídkového stromu do pracovního okna. Každá takováto komponenta je zpravidla doprovázena dvěma soubory. Jeden soubor ukládá informace a provedených změnách tzv. log soubor a druhý soubor, který je ve formátu html o detailech komponenty. Tento nástroj také umožňuje třídit si logické komponenty do knihoven a řídit tak i přístup k takto již předem vytvořeným knihovnám. Mimo třídění EasyPlanner umožňuje uživatelům si definovat logické komponenty. Rovněž je zde možné ukládání a vytváření designu. Umí taktéž generovat tzv. netlisty. Obsahuje v sobě i několik IP(intellectual propety) jader, které je možné využívat.

5 PRAKTICKÉ PROGRAMOVÁNÍ OBVODU AT94K

System Designer je plně sofistikovaný vývojový systém, ve kterém jsou některé nástroje volně šiřitelé a některé nástroje jsou licencované, většinou jde o nástroje firmy Metor Graphics.

5.1 Vývoj aplikace v jazyce VHDL



Obrázek 5-1: Vývoj aplikace v SystemDesigneru.

Při psaní zdrojového kódu ať pro jazyk VHDL nebo Verilog není zde žádný problém. Více je uvedeno v kapitole 4.2.1. Výstupním souborem tohoto procesu je soubor s příponou *.vhd popřípadě *.v.

Problém ovšem nastává při následné simulaci, což je znázorněno na obrázku 5-1 v bodě 2. Bohužel na ústavu radioelektroniky nebyla zakoupena licence pro práci

s nástroji Mentor Graphics ve vývojovém prostředí System Designer. Před několika měsíci bylo zažádáno o vygenerování nové licence, pro tyto nástroje. Odpovědí od firmy Atmel, která se zabývá licencí k vývojovému prostředí System Designer, bylo pouze to, že mají momentálně nějaký problém(zřejmě při kooperaci s firmou Mentor Graphics) při generování licencí. Slíbili zároveň, že po odstranění problému bude poskytnuta licence. Tento proces proběhl asi před rokem a půl, stále bez odpovědi. Snaha o převzetí licence z jiné verze simulátoru ModelSim nebyla úspěšná. Byla vyzkoušena verze simulátoru ModelSim jež je využívána ve spolupráci s nástrojem Xilinx ISE jež je používán na ústavu radioeletroniky. Spolupráce nebyla možná pouze se objevili další chybové hlášení. V těchto hlášeních se objevili problémy s licencí tohoto nástroje. Firma Mentor Graphics poskytuje studentskou verzi simulátoru ModelSim PE. Bohužel ani zde pokusy o převzetí licence nebyly úspěšné. Simulovat VHDL kód je možné, ale už nefunguje spolupráce se System designerem a následná možnost simulace souběžně pro jádro AVR a FPGA pole, jež by mělo být možné.

Po simulaci zdrojových kódů má následovat syntéza HDL, čemuž odpovídá bod 3 na obrázku 5-1. Bohužel syntetizační nástroj Leonardo Spectrum je také licencovaný. Všechny snahy se obejít bez tohoto nástroje vždy ztroskotali. Bylo vyzkoušeno generování netlistu pomocí externího nástroje EASY Planner, který umožňuje také generování netlistu s příponou *.edif. Vyzkoušeno bylo i nahrání souboru z přímo vygenerovaného souboru, který nebyl tvořen v HDL Planneru ale přímo vytvořen nástrojem EASY Planner.

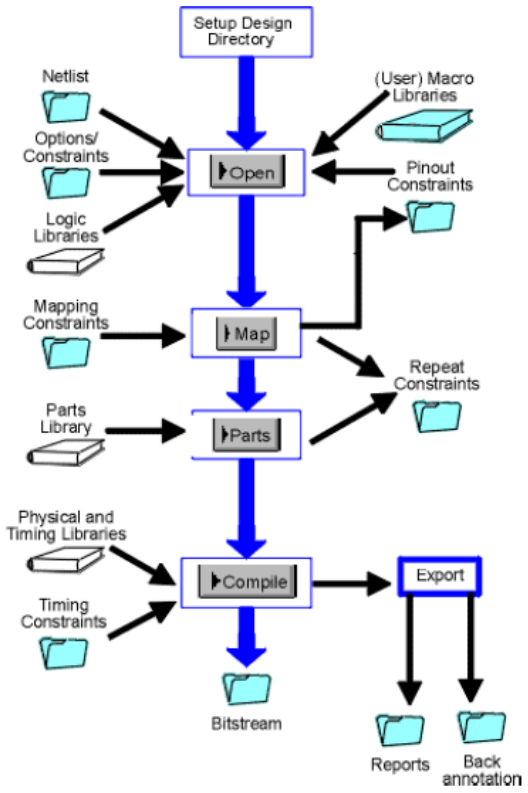
Při vytváření aplikace, která využívá obě části obvodu AT94K (obrázek 5-1 bod 7) veškeré snahy skončily při načítání interface mezi částí obvodu FPGA a AVR jádra. Toto rozhraní má jednoduchou definici a bylo vyzkoušeno jej ručně napsat. Bohužel se nepodařilo vytvořit správný textový soubor, který by byl korektně načten. Nebyla nikde nalezena žádná dokumentace, která by blíže popisovala vlastnosti rozhraní jádra procesoru AVR. V části obvodu s FPGA hradly je možné si nadefinovat jakoukoliv proměnnou, kterou je pak nutné svázat s pevně definovaným rozhraním. Jedná se o různá přerušení, signál hodinového času či o adresní sběrnice.

Dále ve vývoji aplikace dle struktury následuje FPGA Place&Route viz bod 4 na obrázku 5-1. Zde je v System Designeru používán nástroj Figaro. Tento nástroj funguje bez zaplacených licencí. Bylo vyzkoušeno generování maker, které bylo bez problémů. Jak je patrné z obrázku 5-2, vstupují zde soubory z několika nástrojů. Blíže vývoj designu specifikuje obrázek 5-2. Pomocí tlačítka OPEN jsou načítány nástrojem vstupní data, netlist, uživatelsky definovaná makra, soupis vstupních vývodů a logické knihovny s designy. Následuje namapování designu do obvodu. Po umístění designu do obvodu je možné jej ještě lehce editovat a posledním krokem je už jen možná kompilace a výsledek je již zmiňovaný bitstream popřípadě makro.

Nástroje pro tvorbu rozhraní jádra AVR a FPGA a také z nástroje pro syntézu Leonardo Spectrum, popřípadě Precision RTL Synthesis vytvářejí vstupní soubory. Vstupní data mají příponu *.ict pro definici interface. Dalším vstupním souborem jsou datové souboru s koncovkou *.edf jako tzv. netlisty a poslední soubor s příponou *.pin, kde jsou nadefinované vstupní a výstupní vývody na pouzdru obvodu FPSLIC.

Další možností vstupu jsou netlisty s příponami *.wir, jde o netlist generovaný programem WievLogic a také soubor s příponou *.CDB. Koncovka *.CDB označuje makro. Poslední možností vstupu je soubor s příponou *.fgd, což jsou soubory

generované přímo nástrojem Figaro, pro tvorbu maker a supermaker.



Obrázek 5-2: Vývoj designu pro dynamickou rekonfiguraci v nástroji Figaro(převzato z [10]).

Výstupem z tohoto procesu vývoje aplikace by měl být bitstream přímo nahrávaný do FPGA paměti. Příponu má tento soubor *.bst. Na stránkách výrobce je přístupný jeden typ zkoušecího bitstreamu, avšak není určen pro vývojový kit STK594, ale pro verzi STK94. Také je možné z nástroje Figaro spouštět Postlayout coverifikaci, která rovněž používá simulátor ModelSim.

Na obrázku 5-1 je v 5 bodě opět možná post layout simulace. Rovněž je používán simulátor ModelSim firmy Mentor Graphics.

5.2 Systémová integrace rozhraní a simulace

Vývoj aplikace také pokračuje uprostřed obrázku 5-1. v bodech 6 až 9. V bodě šest je možné nastavit si různé vlastnosti chování obvodu. Nastavuje se zde jak bude segmentovaná paměť SRAM a nastavení spouštěcích hodin. Jestli bude paměť fungovat na náběžnou nebo sestupnou hranu, a zda bude používáno časování z obvodu FPGA nebo od hodinového signálu jádra AVR procesoru.

V bodě 7 na obrázku 5-1 je vytvářen pomocí nástrojů interface mezi AVR a FPGA částmi obvodu. Do této procedury vstupují soubory z HDL Planneru. Nevýhodou je, že je zde opět spouštěn simulátor, který potřebuje zakoupit licenci. Během zkoumání System Designeru se nepodařilo vypnout spouštění simulátoru. Protože při každém

spouštění se běh aplikace zastaví s chybovou hláškou. Bohužel se nepodařilo nikdy soubor s koncovkou *.ict vygenerovat. Tento soubor má celkem jednoduchý formát, proto bylo vyzkoušeno si toto rozhranní ručně vytvořit. Dále je uveden jednoduchý příklad tohoto soboru.

```
AVR_IO_Selects ('load'='IOSELA0' )
AVR_Controls ('aWE'='FIOWEA' )
FPGA_Interrupts ('RCO'='INTA0')
Data_from_AVR ('d(0)'='ADINA0' 'd(1)'='ADINA1'   'd(3)'='ADINA3')
AVR_Side_Clocks ('clock'='GCLK5' )
```

Syntaxe je vcelku jednoduchá. Úvodní slovo definuje o jaký typ rozhranní se jedná například `AVR_IO_SELECTS` ukazuje na části pro řízení vstupních a výstupních pinů. Následuje mezera a v závorce je pomocí znaku a názvu 'load' uvozena proměnná definovaná v části vývoje designu v jazyce VHDL. Následuje rovná se a definice, s kterými vstupy popřípadě výstupy má být spojeno v jádru mikrořadiče AVR. V tomto případě jde o `'IOSELA0'`, které vybírá jeden ze vstupně výstupních pinů zmíněných v kapitole 2.3. Opět následuje mezera a uzavření závorek. Proměnné jsou u jádra AVR přesně definované a nelze je libovolně měnit, jako je v případě části při vývoji aplikace pro FPGA část obvodu. Po každé definici jednoho signálu je vždy pokračováno na dalším řádku. V případě, že je používána více bitová sběrnice jako v případě řádku s `Data_from_AVR`, je mezi jednotlivými signály vytvořena mezera.

V bodě 8 a 9 na obrázku 5-1 pokračuje vývoj aplikace Prelayout a Postlayout coverifikaci. Toto bylo také zkoušeno. Opět je zde používán simulátor, kde je nutné zakoupit licenci a proto se nepodařilo provést tyto činnosti.

5.3 Vývoj aplikace pro jádro AVR

V tomto směru nepanuje žádný problém s licenční politikou. Nástroje, které jsou používány jsou vyvíjeny firmou ATMEL. Jak bylo popsáno v kapitole 4.1. Prakticky bylo vyzkoušeno jádro AVR procesoru jednoduchým programem. Tento program byl napsán v jazyce symbolických adres, viz příloha A.1. Program pracoval s LED diodami, které rozsvěcoval v podobě binárního čítače. Byli do obvodu implementovány 2 čítače, jeden vzestupný a jeden sestupný. Oba čítače byli spouštěny pomocí obsluhy externího přerušení INT0 a INT1. Vše bylo nahráno na vývojovou desku STK594. Byl použit port E jako výstupní. Na jeho výstupech byli připojeny LED diody pomocí deseti žilového propojovacího kabelu.

5.3.1 Programování pomocí assembleru

Pro programováni v jazyce symbolických adres je možné využívat nástroje Wavrasm popřípadě AVR Studio. Při programování v jazyce symbolických adres je nutné vložit soubor AT94KDEF.INC. Je nutné nadefinovat i cílové zařízení, vstupně výstupní registry a také je nutné alokovat paměť, ukazatele na data HX a HL. Dobré je také ošetřit i veškerá možná přerušení, které mohou nastat v obvodě AT94K. Úvod

zdrojového kódu může vypadat následovně:

```
.include "AT94KDEF.INC"   ;
.device AT94K40           ; definice ciloveho obvodu
.def XL   = r26           ; definice ukazatele pro data
.def XH   = r27           ; definice ukazatele pro data
```

Následovat by mělo ošetření všech přerušení, které mohou nastat v obvodu viz příloha se zdrojovými kódy A.1.

5.3.2 Programování v jazyce C

Jednou možností jak programovat aplikace pro AVR jádro je možné využití volně šiřitelného a velmi známého kompilátoru avr-gcc. Instaluje se společně s celým balíkem programů ve WinAVR.

Pro práci s obvodem AT94K a jeho programování je nutné vložit do zdrojového kódu hlavičkový soubor IOAT94K.H. Je nutné také nadefinovat vektory přerušení z hradlového pole FPGA. Nastavuje se také bitový přístup pro speciální funkční registry. Nutné je také nadefinovat porty a jejich hlavním účelem.

Další možnosti mimo používání volně šiřitelných programů je možno používat IAR C jazyk a také ICC C jazyk. Nástroje pro práci s jazykem C a vývojem aplikace je nutné zakoupení licence. Na CD dodávané s vývojovým prostředí bylo dodávány i zkušební 30 denní verze těchto programů.

5.4 Programování obvodu AT94K na vývojovém kitu STK594

Pro nahrávání bitstreamu do cílového obvodu je využíváno speciálního hardwaru. Před vlastním programováním je nutné připojit k paralelnímu portu počítače 25 pinový konektor s programátorem ATDH2225. Výstup programátoru je známý, kdy dochází při programování pomocí ISP rozhranní s 10 vývody. Tento kabel je přímo připojen k vývojovému kitu STK594 ke konektoru ISP. Tento konektor nemá osazeny všechny jeho pozice, jedna není osazena a pomáhá tak k jednoduššímu připojení kabelu a jeho záměně.

Vlastní programování je prováděno pomocí nástroje v System Designeru, kde je nutné si připravené zdrojové soubory načíst a následně spojit a nahrát do cílového obvodu. Před samotným programováním je dobré zkontrolovat nastavení registrů obvodu viz kapitola kapitola 4.3.1.

Následně lze už přijít k programování. Před vlastní činností je nutné nejprve přepnout krajní přepínač na vývojové desce STK594 do polohy *PROG*. Následně je možné nahrávat program do obvodu. Po úspěšném nahrání se tento přepínač vrátí opět do polohy *RUN*. Následně pro běh aplikace je nutné ještě zmáčknout tlačítko *RESET* na desce STK594. Tímto se nahraje do paměti obvodu konfigurace a zařízení již pracuje.

6 ALTERNATIVNÍ NÁSTROJE PRO VÝVOJ APLIKACE PRO FPSLIC

Po neúspěšných pokusech s vývojovým prostředím System Desinger bylo vyzkoušeno najít nějakou jinou vhodnou alternativu pro práci s obvody FPSLIC a jejich hradlovou část. Velmi zajímavý byl projekt Slipway and Abits. Pro simulace bylo nalezeno hned několik možných alternativ i studentské verze komerčních simulátorů například ModelSim PE od firmy Mentor Graphics. Při nezdarech i s alternativními nástroji byla alespoň snaha vytvořit projekt a z něho si vygenerovat vstupní data (netlist *.EDIF), který byl následně zkoušen zpracovávat v nástroji Figaro.

Bylo rovněž požádáno firmu Mentor Graphics o uvolnění evaluation licence nástrojů Precision RTL Synthesis popřípadě Leonardo Spectrum. Bohužel pro studentské účely firma poskytnula firma pouze nástroje ModelSim PE studentskou edici, SystemVision trial verzi a nástroj PADS v evaluation verzi. Také bylo zažádáno o studentskou licenci nástroje firmy Synopsys, konkrétně o nástroj Synplicity, který také umožňuje syntézu návrhu v jazyce VHDL. Bohužel ani tato firma neposkytuje studentskou verzi tohoto programu.

6.1 Nástroje pro přímý návrh pro část FPGA

6.1.1 Slipway and Abits

Pro vývoj aplikací na této platformě byl na universitě v Berkeley spuštěn projekt pro vývoj nástroje pro práci s obvody FPSLIC. Celý projekt se jmenuje „Slipway and Abits“.

Slipway – jde o levný a jednoduchý vývojový kit pro experimentování s dynamickou rekonfigurací. Je možné jej jednoduše sestrojit, protože používá snadno dostupné součástky a není nutné žádné speciální vybavení. Tento vývojový kit je složen z obvodu AT94K, stavových LED diod, napájecího obvodu a obvodu FTDI 232R, který slouží jako USB rozhraní pro komunikaci s počítačem. Napájení je provedeno přes USB. Součástí desky je i 24 MHz krystal pro řízení obvodu FPSLIC. Pomocí USB lze resetovat jen jádro AVR popřípadě celý obvod. Rovněž zde probíhá i inicializace a nahrávání programu do zařízení. Pro komunikaci je využíváno jednotky UART přímo na čipu AVR mikrořadiče, který je pomocí obvodu FTDI převedeno na standart USB s maximální rychlostí 1,5 Mbit/s. Celý kit je řízen pomocí software napsaném v jazyce Java. Je zde využito knihoven Libusb a Libftdi.

Po inicializaci ovládací zařízení odešle globální resetovaní impuls do FPGA pole a následně je nastaveno časování a je uložen program do mikrokontroléru. Tento základní program je načten do jádra AVR čeká na příkazy, které jsou přes obvod FTDI odesílány do jednotky UART.

Následně je už vývojový kit připraven pro nahrávání nového designu pro

dynamickou rekonfigurace.

Abits – jde o knihovnu, která je napsána v jazyce Java. Tato knihovna umožňuje veškerou manipulaci s bistreamy. Manipulací se rozumí tvorba, editace, syntaktická analýza a částečná rekonfigurace. Celá knihovna je pod licencí open-source a nepoužívá žádné patentované nástroje. Je využíváno nástrojů Icarus Verilog a BU EDIF. Knihovna Abits je napsána tak, že podporuje jakékoliv obvody u kterých je známa jejich dimenze a podporují konfigurační slova ve tvaru jež bylo zmíněno v kapitole 2.4.2.

Rozhraní je definováno velice jednoduše viz následující zdrojový kód:

```
public interface FpslicInterface {
/** device width, in cells */
public int getWidth();
/** device height, in cells */
public int getHeight();
/** writes data byte "d" at coordinates x,y,z */
public int mode4(int z, int y, int x, int d);
}
```

Bylo vytvořeno i nízkoúrovňové rozhraní FpslicInterface API. Tyto API reprezentují proměnné zdroje, které jsou v zařízení, například buňky, sektory nebo vstupně výstupní bloky viz kapitola 2.2. Tyto API jsou následně přeloženy do konfiguračních slov, které jsou využívány v Cache modu obvodu FPSLIC.

Vývoj tohoto projektu je patrně zastaven, protože poslední aktualizace webových stránek je několik let stará. Bohužel se nepodařilo danou aplikaci zkompilovat. Kompilace byla prováděna na několika architekturách a několika distribucích operačního systému Linux. Nejprve byla kompilace zkoušena s distribucí Mandriva 2010 a kompilátoru gcc několika verzí. Jedná se o verze 4.4, 4.2 a starší verzi 3.5. Verze Javy, které byli postupně instalovány jsou verze 1.5.0 a novější 1.6.0. Pro kompilaci bylo také nutné doinstalovat aplikace swig a curl. Dále byla kompilace zkoušena na serveru FIT VUT Brno, na němž je nainstalována distribuce CentOS release 5.3 (final), kde byl nainstalován kompilátor gcc 4.3.4 a verze Javy 1.6.0. Dále byla kompilace zkoušena na 64 bitové verzi distribuce Fedora. Ani pro 64 bitovou verzi se nepodařilo zkompilovat soubor. Problém byl v napsaných funkcích zrdojového kódu. Z makefilu je zřejmé, že aplikace byla vytvořena pro jinou 32 bitovou architekturu. Soubor makefile pro kompilaci využívá hojně stahování software přímo z online přístupných souborů různých nástrojů. Jedná se o nástroje libusb, avr-gcc a libc a edif parser a další. Program edif parser je také vytvořen na univerzitě v Berkeley. Bohužel se nepodařilo spojit s autorem, který by mohl poskytnout informace, které by odstranili problém při kompilaci zdrojových souborů.

Alternativy pro simulátory jazyku VHDL nebyl problém najít, jenže to není nejpodstatnější problém. Nebylo nikde možné vygenerovat tzv. bitstream a nebo jakkoliv s ním pracovat. Příklady simulátorů s open source licencí jsou například: GTK Wave, Simbus a jiné.

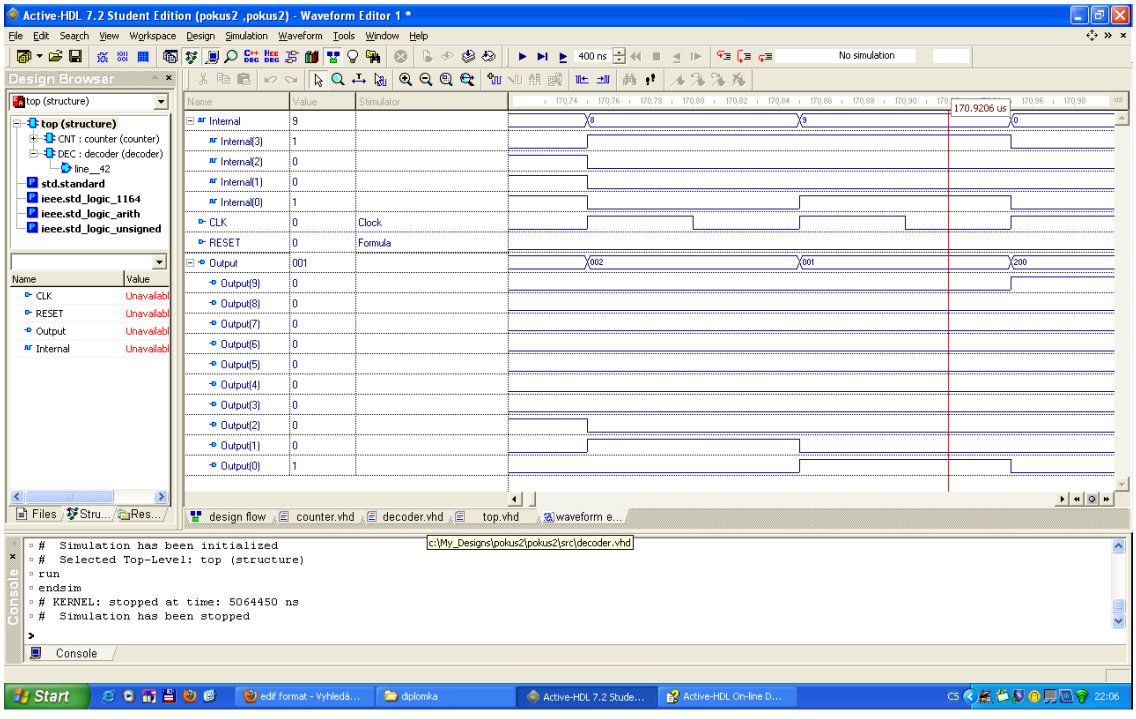
6.1.2 Active-HDL

Další nástroj, který má podporu tvorby designů pro obvody FPSLIC, ale jen pro vývoj pro hradlové pole. Jedná se o nástroj firmy ALDEC. Tato firma uvolnila i studentskou verzi, kde se dá licence obnovovat i několikrát bez problémů. Komerční licenci a její demo verze nebyla firmou poskytnuta s odkazem na zmiňovanou studentskou verzi.

Pro tento nástroj jsou i přímo vytvořeny soubory, pomocí kterých lze importovat i knihovny, které jsou používané nástroji od firmy Atmel a to zejména Figaro IDS. Tyto knihovny lze používat a lze vytvářet makra i IP již vytvořené v nástroji Figaro. Byla použita verze 7.2.1644 Student Edition. Pracovní okno nástroje je zobrazeno na obrázku 6-1. Na obrázku je zřejmé, jak vypadá celý tento nástroj. Obsahuje stejné pracovní okna jako většina podobných programů. Nechybí v dolní části konsole, v které se zobrazují informace a chybové hlášení. Na levé straně je okno projektu pro snadnou orientaci ve vyvíjeném designu. Většinu okna programu zaujímá pracovní plocha, kde se provádí tvorba veškerých zdrojových kódů.

Bohužel tento nástroj opět k syntéze návrhu obvodu používá nástroje třetích stran. Opět pro obvody AT94K je používán nástroje od firmy Mentor Graphics, ať se jedná o Leonardo Spektrum 2005 popřípadě Precision RTL 2006a Synthesis a také od firmy Synplicity jako například Synplify Pro a jiné. Mimo těchto nástrojů tento program podporuje další nástroje pro syntézu obvodů. Jedná o nástroje firem Actel, Altera, Lattice, Xilinx.

V tomto programu je možné i provádět simulace vytvářených designů, ať jednoduché simulace pomocí Waveform nástroje, popřípadě je možné si vytvořit vlastní testbench soubory. Tvorba zdrojových kódů s tímto nástrojem je jednoduchá, nechybí ani podpora generování vstupních a výstupních proměnných.



Obrázek 6-1: Active-HDL s ukázkou simulace pomocí Waveform.

Tento nástroj také umožňuje vytváření stavových diagramů, blokových schémat a nechybí ani podpora jazyka C pro vývoj kódu pro obvody hradlových polí. Nechybí ani podpora práce s tímto nástrojem pomocí Perl skriptů i Tcl skriptů, a tak zrychlit a zefektivnit vývoj složitějších designů.

Prakticky bylo vyzkoušeno vytvořit jednoduchý projekt s jednoduchým čítačem a dekodérem 1 z 10. K tomuto byl vytvořen i hlavní modul. V této studentské verzi chyběl modul pomocí, kterého je možné si jednoduše generovat testbench soubory. Jak již bylo zmíněno, nebylo možné provést syntézu návrhu designu a nepodařilo se vygenerovat výstupní soubory.

6.1.3 Návrhový systém pro obvod AT40K

Protože obvod AT94K je složen z hradlového pole stejného typu jako pro obvod AT40K vznikl nápad vyzkoušet alespoň jednoduchý design pro část FPGA obvodu AT94K. Pomocí této možnosti by byli vytvořeny dva samostatné bloky s různou činností. Jeden blok mohl mít třeba implementovaný například čítač(hradlové pole) a v druhém bloku by mohla být jiná aplikace v procesoru AVR, například ovládání LCD displeje a načítání dat jednotkou UART.

Software pro obvod AT40K je také nástroj Figaro, dokonce i stejná verze. Opět je nutné provést syntézu návrhu a následně načíst do tohoto nástroje. Ovšem to byl problém i při používání stejného nástroje v návrhovém prostředí System Designer. Rovněž byli zkoušeny všechny varianty, jako byli použity pro nástroj Figaro spouštěný přímo ze System Designeru. Běh aplikací byl vždy ukončen chybovými hláškami jako v předešlém případě. Tento krok nevedl k žádnému dalšímu pokroku ve vývoji designu.

6.2 Nástroje pro návrh pro jádro AVR

Během hledání vhodného nástroje s podporou obvodu AT94K byl nalezen zajímavý nástroj pro tvorbu a simulaci pro část mikrokontroléru.

6.2.1 AtmanAvr

Jedná se o nástroj podobný zmiňovanému AVR Studiu. Jedná se o IDE vývojové prostředí s podporou jazyky C a C++. Podporuje mikrokotroléry od firmy Atmel, mezi nimiž je i obvod AT94K. Pro kompilaci zdrojových kódů používá volně šiřitelný gcc kompilátor. Umožňuje i modulární programování. Malou nevýhodou je, že tento nástroj není volně šiřitelný. Je poskytován pod licencí shareware a bezplatně funguje jen po dobu 30 dnů.

Výhodou tohoto programu je jednodušší práce s periferiemi, kdy se pomocí průvodce nastaví veškeré periferií od vstupních pinů po přerušení, jednotek UART. Existuje zde i podpora LCD displejů, kde se pomocí jednoduchého průvodce vše připraví. Používání LCD displejů nebylo vyzkoušeno.

Tento nástroj obsahuje i pomocníka při práci s jednotlivými funkcemi, kdy se dá využívat nabídky a rychle se přepínat mezi jednotlivými funkcemi. Usnadňuje tak orientaci ve větších projektech.

6.3 Nástroje pro generování vstupních souborů pro Figaro IDS

Zde je vybráno několik programů, pomocí kterých bylo možné si vytvořit vstupní informace pro nástroj Figaro, který je používán pro generování bitstreamů do cílového obvodu, jak bylo popsáno v kapitole 4.2.5.

6.3.1 Ikarus verilog

Ikarus Verilog je simulační a syntetizační nástroj. Funguje také jako kompilátor, kdy kompiluje zdrojové kódy z jazyka Verilog(IEEE-1364) do cílového formátu pro různé architektury. Rovněž podporuje simulaci pomocí dávkových souborů. Dávkové soubory jsou tvořeny z příkazů vvp. Pro syntézu návrhu designu kompilátor generuje netlist v požadovaném formátu, například EDIF nebo XNF. Pro tyto jeho vlastnosti byl také použit v této práci. Hlavním cílovým operačním systémem byl Linux, avšak existují i mutace pro operační systémy Windows a Solaris.

Tento nástroj je ovládán pomocí příkazů z příkazového řádku. Grafické rozhranní chybí. Nástroj podporuje překlad z jazyka Verilog do VHDL. Při syntéze je možné definovat několik „cílových“ zařízení. Jednou možností je zařízení *null* – kde se provádí pouze kontrola syntaxe jazyka Verilog. Mezi další možnost patří definovaní cílového zařízení *vvp*, které je definováno standardně. Toto symbolické označení je vytvořeno pro simulační nástroj. Následně je simulace ovládána pomocí vpp příkazů. Dále je možné definovat jako výstupní formát jazyk VHDL pomocí přepínač *vhdl*. Tuto možnost ale podporují až novější verze od verze 0.9. Pomocí tohoto přepínače je generován soubor VHDL z jazyka Verilog. Další možností výstupního formátu je *xnf*, kde se jedná o Xilinx Netlist Format. Tento výstup lze přímo použít pro jiné nástroje od firmy Xilinx pro přímou implementaci do cílového obvodu. Tento formát je starší a tak autor doporučuje používání posledního „cílového“ zařízení a tím je *fpga*. Pomocí tohoto přepínače jsou generovány netlisty formátu EDIF. Podporované rodiny jsou od firmy Xilinx Virtex, Virtex II a nezávislý formát *LPM* od firma Altera. V tento nezávislý formát je rozdělen do tří specifikací.

Pro třídu *virtex* je používán výstupní formát EDIF 2.0.0, stejný jako pro obvod FPSLIC. Totéž platí i pro architekturu *virtex II*, kde jsou zahrnuty i obvody rodiny VIRTEX II Pro. Pro rodiny VIRTEX jsou již definovány knihovny, které se při syntéze používají.

Nástroj Ikarus Verilog, byl zkoušen na operačním systému Linux i Windows. Byla vyzkoušena jeho nejnovější verze 0.9.2. Bohužel s ní se nepodařilo generování netlistu ani na platformě Windows či Linux. Proto byla zkoušena starší verze, která s mírnými problémy generovala výstupní soubor. Jednalo se o verzi 0.8.4. Problémy je myšleno chybějící některé soubory, které se podařilo nakopírovat ještě z jiných verzí tohoto

nástroje. Tyto soubory byly důležité pro vlastní tvorbu výstupních souborů, kde byli definované vlastnosti cílových obvodů. Stačilo je jen zkopírovat z jiné verze a vše už fungovalo správně.

Pro možnost generování výstupního souboru byl zvolen tento nástroj, pomocí kterého bylo vyzkoušeno vygenerovat jednoduchý design. Bylo vyzkoušeno použít několika přepínačů, které jsou uvedeny v textu výše. Byl navíc zkoušen i příkaz ve formátu

```
iverilog –parch=generic-edif –tfpga –S –O andgate.edf andgate.v
```

pomocí, kterého se měl vygenerovat netlist, který by nebyl závislý na nějaké rodinně obvodů. Toto snažení skončilo chybovým hlášením ANACHROISM, které nebylo možné nalézt v nápovědě a ani uvedené dokumentaci. V daném příkazu jednotlivé přepínače mají svůj význam. První přepínač *–parch* definuje architekturu obvodů. Přepínač *–tfpga* značí zvolené výstupní zařízení, v tomto případě univerzální FPGA. Tento přepínač lze i více specifikovat. *–S* značí příkaz, že se má provést syntéza. Přepínač *–O* definuje výstupní soubor, v tomto případě *andgate.edf* a poslední je udáván zdrojový soubor pro možná překlad.

Během generováni netlistů byly vyzkoušeny všechny definované architektury. Například architektura lpm, a oba druhy architektur virtex. Architektura lpm nebyla blíže specifikována na rozdíl od rodin obvodů virtex.

Tento soubor byl následně zkoušen načíst pomocí nástroje Figaro. Bohužel díky slabší podpoře rodin obvodů se to nepodařilo. Problém byl v propojování vnitřních jednotek s vstupními a výstupními porty.

6.3.2 EasyPlanner

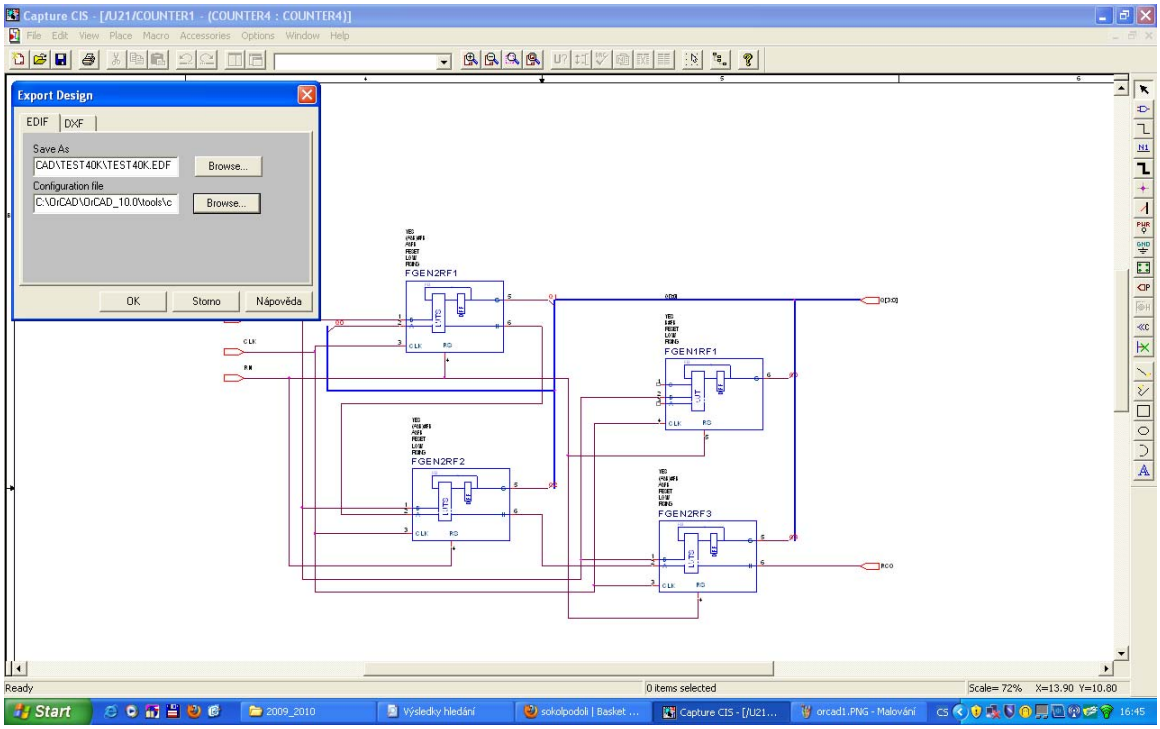
Tento nástroj byl popsán již v kapitole 4.4.1. Byl použit pro vytvoření jednoduché úlohy a následně byl vygenerován netlist jako vstupní data do nástroje Figaro. Načtení vstupních dat se nepodařilo, díky vlastní vnitřní chybě nástroje Figaro číslo 400. O jakou chybu se jedná nebylo možné se dopátrat v příslušné dokumentaci. Chyba byla odeslána firmě Atmel pomocí průvodce přímo v nástroji Figaro k prozkoumání. Bohužel bez jakékoliv odezvy.

6.3.3 OrCAD 10.0

V dnešní době je na trhu mnoho návrhových systémů, které lze použít pro návrh obvodů a simulaci. OrCAD je ucelený balík nástrojů pro návrh schémat obvodů včetně možnosti přechodu do jiných systémů, například pro návrh plošných spojů, simulace i návrh digitálních programovatelných obvodů. V této práci byl použit pro generování souborů netlist *.EDIF.

Z dokumentace nástroje Figaro bylo zjištěno, že lze využít i pro generování schémat s následným převodem do netlistu typu EDIF. Při testování různých možností byla stažena nejnovější verze nástroje Figaro IDS verze 7.6. Ve verzi 7.6 je přímo ikona

pro rychlé spouštění nástroje OrCAD. Bohužel se nepodařilo nikde najít možnost změnit výchozí cestu pro odkaz na spouštění nástroje OrCAD.Přestože nástroj OrCAD byl instalován na místo definované výrobcem, se jej nedařilo spouštět pomocí ikony pro spouštění. Nutné bylo spouštět tento nástroj ručně. V přiložené dokumentaci byli přiloženy i některé příklady designu, určené přímo pro nástroje OrCAD. Toho bylo využito a pomocí softwaru OrCAD 10.0 bylo vyzkoušeno načtení daných schémat a následné vygenerování netlistu EDIF jak je ukázáno na obrázku 6-2.



Obrázek 6-2: Generování netlistu EDIF nástrojem OrCAD 10.0.

Během generování netlistu bylo vyzkoušeno několik konfiguračních souborů, pomocí se převádělo schéma na již zmiňovaný výstup ve formátu EDIF. Každé generování netlistu bylo v pořádku, i bez upozornění. Primárně byl určen pro generování výstupu konfigurační soubor s názvem CAP2EDI.cfg. Ve všech případech načítání skončilo chybovou hláškou. Většinou byl problém při propojování sběrnic a následnému připojení k jejich vstupním, případně výstupním portům. Při dalším zkoumání byli nalezeny další příklady přímo v systému OrCAD, které umožnili další možnosti zkoušení. V programu jsou příklady pro obvody od firem Xilinx, Altera, Actel. Další možností jsou ukázky v „čistém“ jazyce VHDL, avšak při testování i jich se nepodařilo korektně načíst nástrojem Figaro. U jednoho příkladu byl i vytvořený výstupní soubor po syntéze. Při načítání se vyskytly další problémy. Další chybou při načítání byl problém s jednotkami LUT. Program hlásil chybu kdy master jednotka LUT nebyla definovaná v knihovně. V nápovědě bylo uvedeno jak tuto chybu odstranit. Bylo nutné zkontrolovat a nastavit, zda je správně definovaná knihovna, která byla v pořádku. Přestože byli vyzkoušeny i možnosti „čistého“ VHDL a generování netlistu, také vytvořený v příkladu firmou Candance.

6.3.4 Ruční vytvoření souboru EDIF

Po nezdarech pomocí programů vznikl nápad vytvořit netlist EDIF(Electronic Design Interchange Format) ručně.

Tento druh souborů vznikl pro předávání informací mezi různými typy CAD systémů a návrhových systémů pro desky a případné sestavení výrobků. Tento soubor je jednoduchý pro zpracování ostatními systémy a tak tvoří vhodné „rozhraní“ pro jiné systémy. Tím umožňuje jednodušší předávání potřebných informací. Syntaxe tohoto formátu je vytvořena pro snadné strojové zpracování. Tento formát je standardizován asociací EIA(Electronic Industries Alliance). V dnešní době je používána verze 4.0 a 3.0. Tyto verze jsou i standardizovány IEC(International Electrotechnical Commission). Obvody FPSLIC používají starší verzi tohoto standardu a to 2.0.

Struktura souboru je naznačena níže. Na prvním řádku je napsáno jméno celého vyvíjeného designu. Následují stavové informace. V této části se specifikuje verze tohoto programu a úroveň EDIF souboru. Dále zde bývá uváděn autor, umístění tvorby souboru, použitý program pomocí, kterého byl vygenerován soubor, informace o poslední aktualizaci a změně souboru. Dalším bodem je umístění vytvářeného souboru. Po umístění následuje určení externí knihovny. Každá knihovna v souboru EDIF má v sobě umístěny buňky. Každá buňka může být popsána jedním nebo více formami, schématy nebo behaviorálním popisem a dalšími. V knihovnách může být i popsána technologie výroby. Důležitý je i popis rozhranní. Pro oddělení jednotlivých úrovní jsou použity závorky jak je patrné z ukázky.

```
(edif jmeno
        (status information)
        (design where-to-find-them)
        (external reference-libraries)
        (library name
              (technology defaults)
              (cell name
                    (viewmap map)
                    (view type name
                          (interface external)
                          (contents internal)
                    )
              )
        )
  ) [12]
```

Po prozkoumání popisu a několika vygenerovaných souborů různými nástroji bylo upuštěno od myšlenky vytvořit celý zdrojový kód ručně, ale jen zkusit editovat části, které dělali problémy při načítání nástrojem Figaro. Ani toto snažení nevedlo k žádnému závěru, kdy by byl načten korektně a mohl by být generován bitstream, který by se následně nahrál do obvodu. Jeden generovaný netlist soubor je přiložen v příloze A.2.

7 PRAKTICKÉ ZKOUŠENÍ DYNAMICKÉ REKONFIGURACE

Jak bylo popsáno v kapitole 2.5.2 je používán pro dynamickou rekonfigurace pracovní mód AVR Cache módu. Pro tuto činnost je nutné si zažádat firmu Atmel pro hodnoty do registrů FPGAZ a FPGAD. Bohužel firma Atmel ukončila podporu. Při veškerých pokusech o kontaktování podpory od této firmy byla odpověď stejná, že skončili podporu, nejprve podporu tohoto pracovního režimu a pak i celkovou podporu obvodů FPSLIC a nedoporučují vyvíjet další designy s tímto obvodem. Díky špatné podpoře nebylo možné vytvořit jakýkoliv design, který by pracoval v AVR Cache režimu.

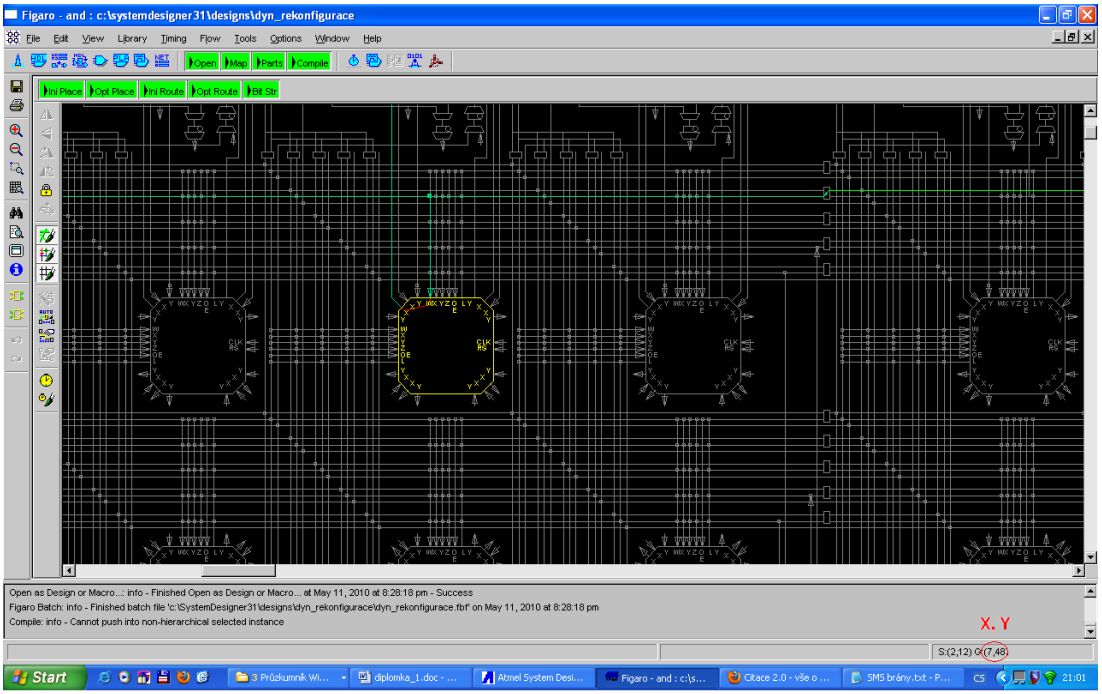
Dále pro práci v AVR Cache režimu je zapotřebí znát také hodnoty registrů FPGAX a FPGAY, které se zjistí v nástroji Figaro, jsou zobrazeny na obrázku 7-1 dole v pravém rohu pracovního okna.

7.1 Postup při návrhu designu pro dynamickou rekonfiguraci

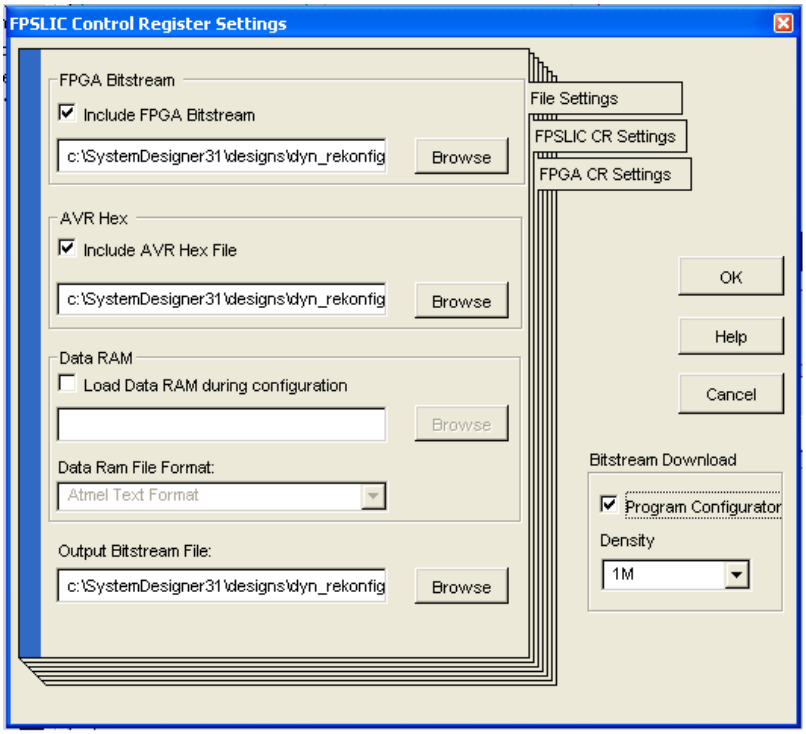
Návrh aplikace je vytvářen v několika větvích jak je zobrazeno na obrázku 5-1. Je nutné vytvořit zdrojové soubory pro část FPGA a část řízení mikrokontrolérem AVR. Dále je nutné specifikovat jejich rozhraní a způsoby komunikace. Po návrhu následuje většinou funkční simulace. Více podrobností o těchto procedurách jsou uvedeny v kapitolách 5.1, 5.2 a 5.3. Následně jsou vytvářeny pro část FPGA vstupní soubory pro nástroj Figaro. V tomto nástroji je nutné zjistit zmiňované hodnoty registrů FPGAX a FPGAY. Tyto hodnoty jsou zobrazeny na obrázku 7-1. Tyto hodnoty je nutné zmenšit o jedničku. Nástroj Figaro používá jako výchozí souřadnici buněk 1,1; avšak nástroje pro vytváření zdrojových kódů pro jádro AVR používá jako výchozí souřadnici 0,0.

Hodnoty všech registrů, určující odkud se má načítat daná konfigurace pro FPGA pole je nutné zadat přímo do zdrojového kódu pro mikrokontrolér, který se stará o řízení rekonfigurace. Následně se zdrojové kódy pro obě části platformy zkompilují a vytvoří se soubory, jak pro část FPGA a AVR. Tyto soubory jsou pak následně spojeny do jednoho bitstreamu, který je nahráván do cílového obvodu. Způsob nastavení je zobrazen na obrázku 7-2.

Je nutné použít zatrhávacích políček, pomocí kterých jsou vybrány vstupní soubory. Také je nutné zatrhnout políčko Program Configurator. Dále je nutné nastavit správně kontrolní registry FPSLIC. Nutné je nastavení povolení resetovacího vstupu jádra AVR a také umožnění přístupu do Cache paměti FPGA pole pomocí mikrokontroléru AVR. Ještě je nutné nastavit také zápis a čtení do SRAM paměti celého obvodu z hradlového pole. V záložce, jež je zobrazeno na obrázku 7-2, FPGA CR settings není nutné vykonávat žádné změny. V této záložce se nastavuje vlastní oscilátor včetně frekvence.

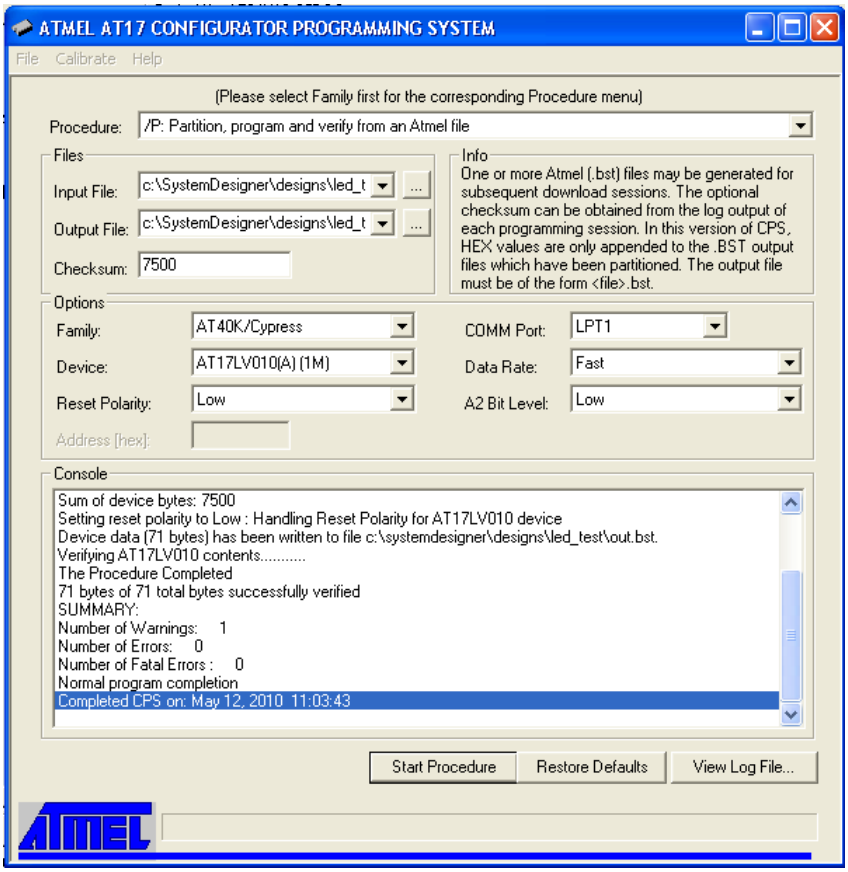


Obrázek 7-1: Nástroj Figaro IDS, označení registrů FPGAX, FPGAY.



Obrázek 7-2: Konfigurace a nahrávání bitstremu do obvodu FPSLIC

Nahrávání do cílového objektu je provedeno po stisku tlačítka OK, kdy je spouštěn další program, který je zobrazen na obrázku 7-3. Procedura nahrávání je automaticky spuštěna, případně může být spuštěna ručně.



Obrázek 7-3: Atmel AT17, konfigurační programovací systém pro FPSLIC

7.2 Ukázka dynamické rekonfigurace

Na stránkách výrobce se podařilo naleznout příklad, který umožňoval ukázku dynamické rekonfigurace. Tento příklad byl napsán pro obvod AT94K40, který je použit na vývojovém kitu STK94. Na vývojovém kitu je použit obvod s méně hradly AT94K10. Při postupu dle návodu pro vývojový kit STK94 bylo nutné trochu pozměnit některé kroky, avšak dílčí myšlenky byli použity. Následně byl nahraný bitstream do obvodu. Nepodařilo se jej spustit. Po tomto neúspěchu bylo vyzkoušeno editovat ručně vstupní soubor pro nástroj Figaro a definovat v něm ručně jiný cílový obvod a to verzi jen s 10 tisíci hradlovými poli AT94K10. Následovalo běžná rutinní činnost pro vygenerování bitstreamu, přes kompilaci a namapování jednotlivých buněk do obvodu. Po nahrání takto upraveného bitstreamu nebyla funkčnost jednoduché ukázky dynamické rekonfigurace. Ukázka měla za úkol překonfigurovat jedno dvoustupové hradlo AND na dvoustupové hradlo OR.

Otázkou je proč nefungovala ani tato jednoduchá rekonfigurace. Bylo nutné zažádat firmu Atmel pro hodnoty do již zmiňovaných registrů FPGAZ a FPGAD.

8 ZÁVĚR

V této práci je rozebrána platforma Atmel FPSLIC. Je zde popsán vývojový kit STK 594 a práce s ním.

V diplomové práci bylo zapracováno na porozumění celé platformě FPSLIC. Dále se autor seznamoval s vývojovým prostředím System Designer. Prakticky bylo vyzkoušena možnost tvorby a případné simulace pro část AVR jádra ať pomocí AVR Studia pro simulaci a programování v assembleru, nebo Wavrasm pro práci s jazykem symbolických adres. Další možnost pro AVR jádro ve formě JTAG nebylo zkoušeno.

Dále pro FPGA část platformy FPSLIC bylo zkoušeno vytvoření zdrojových kódů, které bylo bez problémů. Problém nastal při pokusu o simulaci vytvořeného zdrojového kódu. Simulátor Modelsim, jenž je součástí vývojového prostředí System Designer je licencován, s licencí omezenou na 60 dní. Daná licence není bohužel zakoupena a na tomto ztroskotalo další testování. Po komunikaci s výrobcem bylo zjištěno, že mají problém s generováním nových licencí. Slíbili zároveň, že dají vědět jakmile problém odstraní. Je tomu už cca rok a půl a výrobce se stále neozval. Snaha nějakým způsobem obejít a pak následné generování netlistu se nezdařilo.

Pokus o přímo vytvoření zdrojového kódu pomocí nástroje Figaro, kde bylo zjištěna možnost generování vlastních maker se nepodařilo znovu načíst jako design a následné vygenerování bitstreamu. Softwarový nástroj Figaro, prý umí zpracovávat i jiné netlisty než jen z nástroje Leonardo Spectrum například z programu ORCADu, což si myslím, že by velmi znesnadnilo možnost programování dynamické rekonfigurace. Znamenalo by to opustit programování z jazyka VHDL a vytvářet schémata v tomto nástroji.Tento postup byl také zkoumán, ale nepodařilo se korektně načíst ani vytvořené příklady přímo od výrobce.

Byli prozkoumány i nějaké jiné alternativy pro práci s bitstream, avšak nebylo možno je vyzkoušet, protože projekt zabývající se touto platformou byl zřejmě ukončen. O podpoře od výrobce Atmel také bylo zmiňováno v této práci. Postupně bylo zjištěno, že skončila podpora pracovního režimu pro dynamickou rekonfiguraci a v poslední době dokonce celá podpora platformy FPSLIC.

Problémy s ukončenou podporou firmy Atmel a nemožnost získání licence nástrojů pro práci s obvody AT94K znemožnilo splnit zadání diplomové práce. Pokus o vytvoření aplikací pro jednotlivé části obvodu se povedlo také jen pro část AVR procesoru.

LITERATURA

[1] ATMEL, AT94K05/10/40AL Summary[online]. 2008 [cit. 2009-05-01]. Dostupné na WWW: < http://www.atmel.com/dyn/resources/prod_documents/1138S.pdf>. 5s

[2] ATMEL, AT94K05/10/40AL[online]. 2008 [cit. 2009-05-01]. Dostupné na WWW:< http://www.atmel.com/dyn/resources/prod_documents/doc1138.pdf>. 204s

[3] Coufal, T. Když se řekne FPSLIC [online]. 2007 - [cit. 2009-05-01]. Dostupné na WWW:< http://hw.cz/teorie-praxe/art1978-kdyz-se-rekne-fpslic.html>.

[4] MATOUŠEK, Rudolf; DANĚK, Martin; KUBÁTOVÁ, Hana. Perspektivy dynamické rekonfigurace programovatelných polí FPGA. *Sdělovací technika* [online]. 2006, 2006, 4, [cit. 2010-05-11]. Dostupný z WWW: <http://www.stech.cz/sqlcache/04_06.pdf>.

[5] KVASNIČKA, Jiří. *Vysoce spolehlivý systém založený na bázi hradlových polí* [online]. Praha, 2006. 79 s. Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická. Dostupné z WWW: <https://dip.felk.cvut.cz/browse/pdfcache/kvasnj1_2006dipl.pdf>.

[6] DANĚK, M; HONZIK, P; KADLEC, J; MATOUSEK, R; POHL, Z. Reconfigurable System on a Programmable Chip Platform. *Atmel Applications Journal* [online]. 2005, 4, [cit. 2010-05-12]. Dostupný z WWW: <http://www.atmel.com/journal/documents/issue4/pg9_12_Reconfig.pdf>.

[7] ATMEL, STK 594 User Guide. 2002. 40s

[8] LAFAYETTE, Guy Lecurieux. *A System Level Integration Company* [online]. [cit. 2010-03-12]. Dostupné z WWW: <http://www-lasl.univ-littoral.fr/GRAISyHM-AAA-99/GRAISyHM-AAA-99/html/vol2/atmel.pdf>.

[9] KASI, K.; KAROUBALIS, T.; DANĚK, M.; POHL, Z. *Figaro – an automatic tool flow for desings with dynamic reconfiguration*. In IEEE *International Konference on Toled Progtammable Logic and Applications*. 2005. s 591 – 593.

[10] ATMEL, Figaro – nápověda.k programu 2002.

[11] WILLIAMS, Stephen. *Icarus Verilog* [online]. 2009 [cit. 2010-04-22]. Dostupné z WWW: <http://www.icarus.com/eda/verilog/>.

[12] *The Icarus Verilog Wiki* [online]. 2006, 3.8. 2009 [cit. 2010-04-22]. Dostupné z WWW: <http://iverilog.wikia.com/wiki/Main_Page>

[13] MINIXHOFER, R. Integrating Technology Simulation into the Semiconductor Manufacturing Environment [online]. 2002 [cit.2010-05-16]. Electronic Design Interchange Format (EDIF). Dostupné z WWW: <http://www.iue.tuwien.ac.at/phd/minixhofer/node53.html>.

[14] Wikipedie : The free encyclopedia [online]. 2008, 7.4 2010 [cit. 2010-05-16]. EDIF. Dostupné z WWW: <http://en.wikipedia.org/wiki/EDIF>.

[15] VAIDYANATHAN, Ramachandran; TRAHAN, Jerry. Dynamic reconfiguration : Architectures and Algorithms. New York : Kluwer Academic/Plenum Publisher, 2003. 512 s. ISBN 0-306-48189-8.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

BLIF	Berkley logic interchange format
CPLD	Complex Programmable Logic Device
DCF	Dynamic Constraints File
EDIF	Electronic Design Interchange Format
EIA	Electronic Industries Alliance
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FPSLIC	Field Programmable System Level Integrated Circuits
IDE	Integrated development environment
IEC	International Electrotechnical Commission
IP	Intellectual Property
ISP	In-System Programming
SRAM	Static Random Access Memory
TWSI	Two-Wire Serial Interface
UART	Universal Asynchronous receiver/transmitter
VHDL	Very high speed circuits Hardware Description Language
XNF	Xilinx Netlist Format

SEZNAM PŘÍLOH

A ZDROJOVÉ KÓDY

A.1 Ukázka zdrojového kódu pro AVR jádro

```
;*****************************************************************
;*
;* Nazev:               AT94K FPSLIC test led
;* Verze:
;* Cilovy obvod:              AT94K10
;*
;*
;*****************************************************************

;*****************************************************************
;*                              Vkladane soubory
;*****************************************************************

.nolist
.include "AT94KDEF.INC"
.list

;*****************************************************************
;*                            Globalni registry
;*****************************************************************

.def    Delay   =r17
.def    Delay2  =r18
.def    Delay3  =r20
.def    Delay4  =r21
.def    rTemp   =r16                ;docasny pomocny registr
.def    Temp    =r19                ;docasny pomocny registr

;*****************************************************************
;*                            Vektory preruseni
;*****************************************************************

        jmp     RESET                   ;Reset    Handle:    Program
Execution Starts Here
        jmp     FPGA_INT0               ;FPGA Interrupt0 Handle
        jmp     EXT_INT0                ;External Interrupt0 Handle
        jmp     FPGA_INT1               ;FPGA Interrupt1 Handle
        jmp     EXT_INT1                ;External Interrupt1 Handle
        jmp     FPGA_INT2               ;FPGA Interrupt2 Handle
        jmp     EXT_INT2                ;External Interrupt2 Handle
        jmp     FPGA_INT3               ;FPGA Interrupt3 Handle
        jmp     EXT_INT3                ;External Interrupt3 Handle
        jmp     TIM2_COMP              ;Timer/Counter2 Compare Match
Interrupt Handle
        jmp      TIM2_OVF                  ;Timer/Counter2 Overflow
Interrupt Handle
        jmp      TIM1_CAPT                 ;Timer/Counter1 Capture
Event Interrupt Handle
        jmp      TIM1_COMPA                ;Timer/Counter1 Compare
Match A Interrupt Handle
```

```
        jmp      TIM1_COMPB                ;Timer/Counter1 Compare
Match B Interrupt Handle
        jmp      TIM1_OVF                  ;Timer/Counter1 Overflow
Interrupt Handle
        jmp      TIM0_COMP                 ;Timer/Counter0 Compare
Match Interrupt Handle
        jmp      TIM0_OVF                  ;Timer/Counter0 Overflow
Interrupt Handle
        jmp     FPGA_INT4                ;FPGA Interrupt4 Handle
        jmp     FPGA_INT5                ;FPGA Interrupt5 Handle
        jmp     FPGA_INT6                ;FPGA Interrupt6 Handle
        jmp     FPGA_INT7                ;FPGA Interrupt7 Handle
        jmp      UART0_RXC                   ;UART0 Receive Complete
Interrupt Handle
        jmp     UART0_DRE                ;UART0 Data Register Empty
Interrupt Handle
        jmp      UART0_TXC                  ;UART0 Transmit Complete
Interrupt Handle
        jmp     FPGA_INT8                ;FPGA Interrupt8 Handle
        jmp     FPGA_INT9                ;FPGA Interrupt9 Handle
        jmp     FPGA_INT10               ;FPGA Interrupt10 Handle
        jmp     FPGA_INT11               ;FPGA Interrupt11 Handle
        jmp      UART1_RXC                   ;UART1 Receive Complete
Interrupt Handle
        jmp     UART1_DRE                ;UART1 Data Register Empty
Interrupt Handle
        jmp      UART1_TXC                  ;UART1 Transmit Complete
Interrupt Handle
        jmp     FPGA_INT12               ;FPGA Interrupt12 Handle
        jmp     FPGA_INT13               ;FPGA Interrupt13 Handle
        jmp     FPGA_INT14               ;FPGA Interrupt14 Handle
        jmp     FPGA_INT15               ;FPGA Interrupt15 Handle
        jmp       TWS_INT                     ;TWS Serial Interrupt
Handle

FPGA_INT0:
        reti
EXT_INT0:


  loop:

        out     PORTE,Temp      ;vystupni data na PORTE
        inc     Temp

  DLY:
        dec     Delay
        brne    DLY
        dec     Delay2
        brne    DLY
        rjmp    loop            ;opakovaci smycka


        reti
FPGA_INT1:

        reti
EXT_INT1:
  loop1:
```

```
        out     PORTE,Temp      ;vystupni data na PORTE
        dec     Temp

  DLY1:
        dec     Delay3
        brne    DLY1
        dec     Delay4
        brne    DLY1
        rjmp    loop1            ;opakovaci smycka
        reti
FPGA_INT2:
        reti
EXT_INT2:
        reti
FPGA_INT3:
        reti
EXT_INT3:
        reti
TIM2_COMP:
        reti
TIM2_OVF:
        reti
TIM1_CAPT:
        reti
TIM1_COMPA:
        reti
TIM1_COMPB:
        reti
TIM1_OVF:
        reti
TIM0_COMP:
        reti
TIM0_OVF:
        reti
FPGA_INT4:
        reti
FPGA_INT5:
        reti
FPGA_INT6:
        reti
FPGA_INT7:
        reti
UART0_RXC:
        reti
UART0_DRE:
        reti
UART0_TXC:
        reti
FPGA_INT8:
        reti
FPGA_INT9:
        reti
FPGA_INT10:
        reti
FPGA_INT11:
        reti
UART1_RXC:
        reti
```

```
UART1_DRE:
        reti
UART1_TXC:
        reti
FPGA_INT12:
        reti
FPGA_INT13:
        reti
FPGA_INT14:
        reti
FPGA_INT15:
        reti
TWS_INT:
        reti

RESET:
        cli                             ;zakazani globalniho preruseni

        ldi     rTemp, high(0x0FFF)    ;inicializace Stack Pointer
        out     SPH, rTemp
        ldi     rTemp, low(0x0FFF)
        out     SPL, rTemp

        ldi         rTemp,(1<<INT0)+(1<<INTF0)+(1<<INT1)+(1<<INTF1)
        ;povoleni externiho preruseni 0 a 1
        out     EIMF, rTemp

        ser     Temp
        out     DDRE,Temp             ;PORTE nastaven jako vystupni
        clr     Temp                        ;vymazani promene Temp


        sei                           ;povoleni globalniho preruseni
MAIN:
        rjmp    MAIN     ;nekonecna smycky, ktera ceka na preruseni
```

A.2 Ukázka EDIF souboru generovaný nástrojem Icarus Verilog

```
(edif andgate
 (edifVersion 2 0 0)
 (edifLevel 0)
 (keywordMap (keywordLevel 0))
 (status
  (written
   (timestamp 2009 11 28 11 34 46 )))
 (external at94k
  (edifLevel 0)
  (technology (numberDefinition ))
  (cell obuf (cellType GENERIC)
   (view NETLIST  (viewType NETLIST)
    (interface
     (port A (direction INPUT))
     (port PAD (direction OUTPUT)
      (property PAD (string "T"))))))
  (cell ibuf (cellType GENERIC)
   (view NETLIST  (viewType NETLIST)
    (interface
     (port PAD (direction INPUT)
      (property PAD (string "T")))
     (port Q (direction OUTPUT)))))
  (cell zero (cellType GENERIC)
   (view NETLIST  (viewType NETLIST)
    (interface
     (port Q (direction OUTPUT)))
    (property AREA (string "1.000000")))))
 (library OPERATORS
  (edifLevel 0)
  (technology (numberDefinition ))
)
 (library work
  (edifLevel 0)
  (technology (numberDefinition ))
(cell andgate (cellType GENERIC)
   (view INTERFACE  (viewType NETLIST)
    (interface
  (port a (direction INPUT))
  (port b (direction INPUT))
  (port q (direction OUTPUT))
)
    (contents
     (instance  i0  (viewRef  NETLIST   (cellRef  ibuf  (libraryRef
at94k ))))
     (instance  i1  (viewRef  NETLIST   (cellRef  ibuf  (libraryRef
at94k ))))
     (instance  i2  (viewRef  NETLIST   (cellRef  obuf  (libraryRef
at94k ))))
    (instance  GND0  (viewRef  NETLIST   (cellRef  zero  (libraryRef
at94k ))))
  (net n0
      (joined
```

```
       (portRef a)
       (portRef PAD (instanceRef i0 ))))
  (net n1
      (joined
       (portRef b)
       (portRef PAD (instanceRef i1 ))))
  (net n2
      (joined
       (portRef q)
       (portRef PAD (instanceRef i2 ))))
    (net GND0
      (joined
       (portRef Q (instanceRef GND0 ))
))))))
  (design andgate  (cellRef andgate (libraryRef work ))))
```