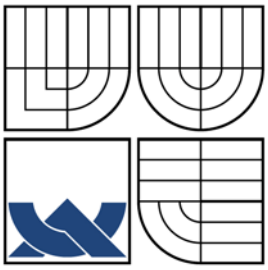
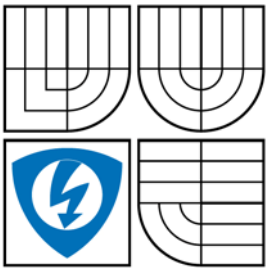


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

METODY ČÁSTEČNÉ REKONFIGURACE PROGRAMOVATELNÝCH STRUKTUR

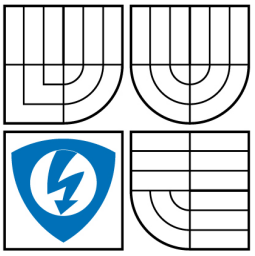
PARTIAL RECONFIGURATION METHODS BASED ON PROGRAMMABLE STRUCTURES

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. JAN KOLÁŘ
AUTHOR

VEDOUCÍ PRÁCE — ING. SOBĚSLAV VALACH
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Jan Kolář	***ID:***	22995
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Metody částečné rekonfigurace programovatelných struktur

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti částečné rekonfigurace programovatelných struktur obvodů FPGA firmy Xilinx. Zaměřte se na všechny možné přístupy, zhodnoťte jejich vlastnosti, výhody a nevýhody.

DOPORUČENÁ LITERATURA:

Firemní literatura Xilinx

Termín zadání:	9.2.2009	***Termín odevzdání:***	25.5.2009

Vedoucí práce: Ing. Soběslav Valach

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

Anotace závěrečné práce:

Tato diplomová práce se zabývá možnostmi částečné rekonfigurace programovatelných struktur.

Teoretická část obsahuje základy metod částečné rekonfigurace FPGA firmy Xilinx a je zpracována pro procesory Spartan 3, Virtex II, Virtex 4, Virtex 5. Zahrnuje popis konfiguračních rozhranní a jejich využití při rozdílové a modulární částečné rekonfiguraci.

Rozdílová částečná rekonfigurace je prakticky v druhé části testována na desce Spartan 3E Starter Kit a modulární částečná rekonfigurace na desce ML501. Konfigurační bitstreamy jsou přiloženy na CD.

Potřebný software poskytla firma Xilinx Inc. a konkrétně jde o programy ISE 9.2i a PlanAHEAD 9.2

Anotace závěrečné práce ENG:

This master's thesis dissertates of partial reconfiguration methods based on programmable structures.

In theoretical part it deals with difference and modular-based method of Xilinx's FPGAs Partial reconfiguration. Options of both reconfiguration techniques were written for Spartan 3, Virtex II, Virtex 4 and Virtex 5 processors.

Diference-based method was in practical part tested on Spartan 3E Starter Kit and modular-based on ML501 board. All configuration bitstreams are included on CD.

Xilinx Inc. provided all needed software tools such as ISE9.2i and PlanAHEAD.

Klíčová slova:

FPGA
částečná rekonfigurace
rozdílový
modulární
Xilinx
Spartan 3
Virtex 4
Virtex 5
Virtex II

Klíčová slova ENG: FPGA
partial reconfiguration
difference-based
modular-based
Xilinx
Spartan 3
Virtex 4
Virtex 5
Virtex II

Bibliografická citace mé práce:

KOLÁŘ, J. *Metody částečné rekonfigurace programovatelných struktur*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 73 s. Vedoucí diplomové práce Ing. Soběslav Valach.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Metody částečné rekonfigurace programovatelných struktur 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: **25. května 2009** ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Soběslavu Valachovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **25. května 2009** ……

podpis autora

OBSAH

1. PŘIBLÍŽENÍ PROBLEMATIKY 6
2. ÚVOD DO ARCHITEKTURY FPGA 8
2.1 CLB 8
2.2 DCM 9
2.3 IOB 9
2.4 Spoje 10
2.5 Paměť 10
3. MOŽNOSTI KONFIGURACE FPGA 11
3.1 Konfigurační rozhranní 11
3.1.1 Sériové konfigurační rozhranní 12
3.1.2 SelectMAP 13
3.1.3 SPI 15
3.1.4 BPI 17
3.1.5 JTAG 18
3.1.6 ICAP 18
3.2 Bezpečnost a omezení 20
3.2.1 Bezpečnostní bity 20
3.2.2 Kódování 20
3.2.3 Autentizace 21
4. TVORBA BITSTREAMU 22
4.1 Jednotlivé kroky implementace 23
4.1.1 NGDBuild 23
4.1.2 The User Constraints File (UCF) 24
4.1.3 The Logical Design Rule Check 24
4.1.4 MAP The Technology Mapper 24
4.1.5 PAR – Place and Route 25
4.1.6 BitGen 25
5. ZÁKLADY MODULÁRNÍHO NAVRHOVÁNÍ 27
5.1 Top-Level Design 28

5.2 Moduly 28
5.3 Přímá implementace modulů 28
5.4 Konečné přiřazení 29
6. ÚVOD DO REKONFIGURAČNÍCH TECHNIK 30
6.1 Základní rozdělení z hlediska funkčnosti zařízení 31
6.1.1 Dynamická částečná rekonfigurace 31
6.1.2 Statická částečná rekonfigurace 31
6.2 Základní rozdělení z hlediska rozsahu změn 31
6.2.1 Modulární částečná rekonfigurace 31
6.2.2 Rozdílová částečná rekonfigurace 31
7. ROZDÍLOVÁ ČÁSTEČNÁ REKONFIGURACE 32
7.1 Provádění malých změn v designu pomocí FPGA Editoru 33
7.1.1 Změna LUT logiky 33
7.1.2 Změna obsahu Block RAM 35
7.1.3 Změna I/O atributů 36
7.1.4 Další měnitelné elementy 37
7.2 Vytvoření rozdílového bitstreamu 37
8. MODULÁRNÍ ČÁSTEČNÁ REKONFIGURACE 39
8.1 Definice a omezení rekonfigurovatelných modulů 39
8.2 Požadavky na strukturu HDL 40
8.3 Design flow pro částečnou rekonfiguraci 41
8.4 Komunikace přes bus makro 42
8.5 Implementace modulárního designu 43
8.5.1 Initial budgeting 44
8.5.2 Active modul 44
8.5.3 Final assembly 45
8.6 Tvorba bitstreamu pro částečnou rekonfiguraci 47
9. MOŽNOSTI REKONFIGURACE JEDNOTLIVÝCH ČIPŮ 48
9.1 Spartan 3 48
9.2 Virtex II a Virtex II pro 50
9.3 Virtex IV 51

9.3.1 DRP 51
9.3.2 DPSM 51
9.4 Virtex V....52
9.4.1 Fallback 52
9.4.2 Watchdog....52
10. SOFTWARE PRO PRÁCI S FPGA XILINX....54
10.1ISE Foundation/WebPACK 54
10.2PlanAHEAD....54
10.3Early Access....55
11. PRAKTICKÁ ČÁST 56
11.1Diferenční částečná rekonfigurace....56
11.1.1 Testovací kód....56
11.1.2 Postup při rekonfiguraci 56
11.1.3 Subjektivní zhodnocení 59
11.2Modulární částená rekonfigurace....59
11.2.1 Testovací kód....59
11.2.2 Vytvoření modulárního systému schopného PR 60
11.2.3 Konfigurace zařízení 64
11.2.4 Ověření funkčnosti 65
11.2.5 Subjektivní zhodnocení 66
12. ZHODNOCENÍ PRAKTICKÉ ČÁSTI 67
13. ZÁVĚR 68
14. POUŽITÁ LITERATURA 70
15. POUŽITÉ ZKRATKY A REGISTRY 71
16. SEZNAM PŘÍLOH 73

SEZNAM OBRÁZKŮ

Obrázek 2.1-1 Základní architektura FPGA 8
Obrázek 2.1-1 Vnitřní struktura CLB 9
Obrázek 3.1-1 Primitiva sériového konfiguračního rozhranní 12
Obrázek 3.1-2 Průběh signálů pro sériová konfigurační rozhranní 12
Obrázek 3.1-3 Sériové konfigurační rozhranní 13
Obrázek 3.1-4 rozhranní SelectMAP 14
Obrázek 3.1-5 SelectMAP primitiva 14
Obrázek 3.1-6 Průběh signálů SelectMAP při nepřerušovaném nahrávání dat 15
Obrázek 3.1-7 standardní zapojení pro rozhranní SPI 16
Obrázek 3.1-8 Průběh signálů pro SPI 16
Obrázek 3.1-9 Průběh signálů pro BPI 17
Obrázek 3.1-10 rozhranní BPI 17
Obrázek 3.1-11 Typická architektura JTAG 18
Obrázek 3.1-12 ICAP primitiva 19
Obrázek 3.2-1 Xilinx design flow 22
Obrázek 4.1-1 NGDBuild design flow 23
Obrázek 4.1-2 MAP design flow 24
Obrázek 4.1-3 PAR design flow 25
Obrázek 4.1-4 BitGen design flow 26
Obrázek 4.1-1 Postup prací při modulárním navrhování 27
Obrázek 5.1-1 Top-level design 28
Obrázek 7.1-1 Zobrazení logického bloku 34
Obrázek 7.1-2 Změna logické funkce bloku 35
Obrázek 7.1-3 Změna obsahu blokové paměti RAM 36
Obrázek 7.1-4 Změna I/O standardů 37
Obrázek 8.1-1 Návrh FPGA s rekonfigurovatelnými moduly (PR Logic) 40
Obrázek 8.2-1 Komunikace modulů přes bus makro 41
Obrázek 8.4-1 Bus makro 42
Obrázek 8.4-2 Fyzická reprezentace bus makra 43

Obrázek 8.5-1 Správně navržený modul pro částečnou rekonfiguraci 45
Obrázek 8.5-2 Finální podoba modulárního systému v FPGA Editoru 46
Obrázek 9.1-1 příklad MultiBootu pro Spartan 3E 50
Obrázek 9.3-1 Dynamická rekonfigurace bloků pomocí DRP 51
Obrázek 11.1-1 Výřez desky s použitými přepínači 56
Obrázek 11.1-2 Cesta k FPGA Editoru 57
Obrázek 11.1-3 logická funkce pro vybranou LUT 58
Obrázek 11.1-4 Konfigurace desky pomocí bitstreamu 59
Obrázek 11.2-1 Celkový přehled příkladu MPR 60
Obrázek 11.2-2 Výběr statických bloků 61
Obrázek 11.2-3 Vytvoření PRR 61
Obrázek 11.2-4 Umístění komponent 62
Obrázek 11.2-5 Rozmístění bus maker 62
Obrázek 11.2-6 Nastavení rekonfigurovatelných modulů 63
Obrázek 11.2-7 Implementace modulů 63
Obrázek 11.2-8 Generování konfiguračního bitstreamu 64
Obrázek 11.2-9 Konfigurace zařízení pomocí bitstreamu 65
Obrázek 11.2-10 Výřez desky ML505 s použitými tlačítky a ledkami 65

SEZNAM TABULEK

Tabulka 3-1 Srovnání signálů ICAP a SelectMAP primitiv 19
Tabulka 3-2 Možnosti zabezpečení bitstreamu 21
Tabulka 9-1 Srovnání základních parametrů FPGA 48
Tabulka 9-2 MultiBoot pro rodinu Spartan 3 49

1. PŘIBLÍŽENÍ PROBLEMATIKY

Stále více vývojářů se v posledních letech snaží porozumět problematice programovatelných logických součástek. Stejně jako ve světě software existují closed-source a open-source řešení, můžeme stejné prvky nalézt i ve světě hardware. Proč mluvíme zrovna o open-source u programovatelných logických zařízení? Popis hardwaru je totiž ve formě zdrojového textu na poměrně vysokém stupni abstrakce. Proto je srozumitelný a může být dále šířen a upravován, stejně jako u softwaru.

Obecně je zažitá představa, že hardware je těžko modifikovatelný a že hotový čip nelze jednoduše zkopírovat či upravit. Přes toto omezení se ale přenesla programovatelná logika (FPGA či CPLD), na jejímž vývoji se začalo pracovat již v 60-tých letech minulého století. Programovatelný obvod je sám o sobě křemíková deska, kterou je pro uvedení do činnosti potřeba nakonfigurovat. Programování probíhá na nižší úrovni, než jsme zvyklí z implementace softwaru, a má i svá další specifika jako nutnost dodržení architektury nebo rozměrů. V současné době se využívá abstraktního programovacího jazyka VHDL případně Verilogu a je kladen důraz na precizní možnost simulace navrženého obvodu. Návrh tedy musí být vytvořen, odsimulován a posléze nahrán do pole, které je v tu dobu odstavené z činnosti. Od roku 1993 přibyla možnost částečné úpravy již vytvořených polí „za letu," což znamená, že zařízení může při rekonfiguraci některé své části dál pokračovat v práci.

Hlavní výhodou programovatelné logiky je tedy možnost její modifikace, kdy již nepoužitelný návrh můžeme vyměnit za nový, aniž by se musel kupovat nebo vyvíjet nový hardware. Současným problémem je ovšem dostupnost vývojových nástrojů, která je horší než ve světě programování softwaru.

K čemu je ale takovéto zařízení v praxi? Programovatelná logika je sice oproti mikrokontrolérům dražší, je však mnohem rychlejší. Proto lze hledat uplatnění především v aplikacích, ve kterých je třeba zpracovat velká množství dat v reálném

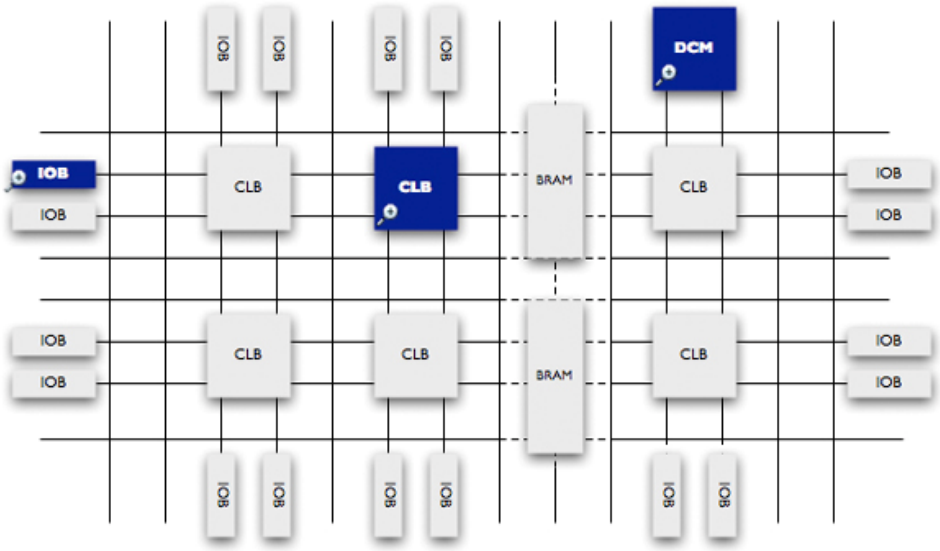
čase. Takovým systémem jsou například aplikace počítačového vidění spojené s kompresí a analýzou pořizovaných obrazových sekvencí. Další oblastí použití jsou telekomunikace, kde změna protokolu znamená pouze nahrání nové konfigurace do pole. Pro všechny aplikace reálného času je také dobře využitelná částečná rekonfigurace za plného provozu zbytku systému, aby nedošlo k přerušení toku a zpracování dat.

Ačkoliv práce s FPGA je v současné době doménou především akademických pracovišť, v budoucnu lze očekávat nasazení programovatelných systémů také přímo v osobních počítačích. Takové PC bude vybaveno speciálním akcelerátorem s jedním nebo více FPGA obvody. Běžící aplikace si do FPGA nahraje svůj konkrétní hardwarový návrh pro urychlení specifických operací, které by se jinak prováděly aplikačním programem a po ukončení uvolní prostor další aplikaci. Tento trend povede k dalšímu nárůstu výkonu počítačů.

Vzhledem k tomu, že zpracovávaná problematika je globálního charakteru, dochází v textu k použití nepřeložených anglických výrazů. To je způsobeno především neexistujícími ryze českými ekvivalenty a již zaběhlou terminologií využívající původních anglických názvů. Většinou jde o pojmenování fází postupu navrhování, případně o běžné termíny z IT praxe. Součástí práce je tedy i seznam použitých zkratek, které vyžadovaly bližší vysvětlení. V práci je většina textu převzata z konfiguračních manuálů a toolboxů firmy Xilinx Inc. , proto citace nejsou uváděny standardní formou, ale pouze seznamem literatury, ve kterém je uvedeno, z jaké literatury bylo v dané kapitole čerpáno.

2. ÚVOD DO ARCHITEKTURY FPGA

Field Programmable Gate Arrays (FPGAs) jsou polovodičová zařízení založená na matici konfigurovatelných logických bloků (CLB), které jsou vzájemně spojeny programovatelnými propojeními. Jako protiklad k Application Specific Integrated Circuits (ASICs), u kterých je zařízení stavěno speciálně pro požadovaný návrh, mají FPGA výhodu v možnosti naprogramování požadované aplikace do již navrženého univerzálního pole. Ačkoliv jsou ještě na trhu k dispozici jednorázově programovatelná FPGA, dominantní zastoupení mají zařízení založená na SRAM, která mohou být opětovně programována podle potřeby.

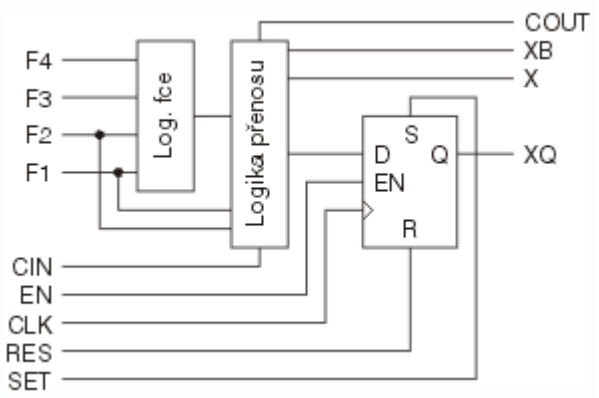


Obrázek 2.1-1 Základní architektura FPGA

2.1 CLB

Configurable Logic Block je základní stavební jednotka FPGA. Každé CLB obsahuje matici spínačů s čtyřmi nebo šesti vstupy, specifický elektrický obvod (MUX, atd.…) a klopný obvod. Matice spínačů je vysoce flexibilní a umožňuje konfiguraci jakékoliv kombinační logiky, posuvných registrů nebo paměti RAM. Navíc některé signály mohou být vedeny z jednoho CLB do druhého bez nutnosti použití globální propojovací matice, což výrazně redukuje dopravní zpoždění

signálů. To je v praxi nutná podmínka správné funkce například sčítaček a násobiček, kde je potřeba rychlý přenos signálu.



Obrázek 2.1-1 Vnitřní struktura CLB

2.2 DCM

S růstem velikosti FPGA roste požadavek na kvalitu hodin na čipu, aby se eliminovala zpoždění plynoucí z distribuce hodinového signálu zařízením. Velmi často obvody FPGA obsahují PLL (Phase Locked Loop) nebo DLL (Delay Locked Loop) pro obnovení charakteristik hodinového signálu, případně pro násobení nebo dělení jeho frekvence. Každý DCM modul spravuje několik tras globálních hodin. Monitorováním vzorku hodinového výstupu z DCM, který je díky PLL/DLL nezpožděný, dochází ke srovnání globálních hodin a hodin DCM. Následně je upravena fáze globálních hodin tak, aby byla shodná s fází DCM a tím pádem i stejná, tedy nezpožděná, pro celé FPGA.

2.3 IOB

Vstupně výstupní bloky poskytují obousměrné rozhranní mezi I/O pinem a interní logikou FPGA. Vstupní cesta přenáší data z I/O pinu do vnitřní logiky přes volitelné programovatelné zpoždění nebo přes klopné obvody, zatímco výstupní cesta zajišťuje přenos opačný. Tyto bloky obvykle obsahují registr, budič, multiplexer a ochranné obvody.

Dnešní FPGA zajišťují podporu pro mnoho I/O standardů pro zajištění ideálního propojení jednotlivých součástí rozsáhlých systémů. I/O jsou sdružovány do bank, kdy každá banka může nezávisle na jiné podporovat rozdílný I/O standard.

2.4 SPOJE

Zatímco CLB zajišťuje logické schopnosti, flexibilní spoje přenášejí signál mezi CLB a I/O. Návrh tras probíhá automaticky a tyto spoje jsou před uživatelem schovány, pokud není specifikováno jinak. Tím se výrazně redukuje komplexnost návrhu.

2.5 PAMĚŤ

Většina moderních FPGA obsahuje několik bloků rychlé synchronní statické paměti RAM, což umožňuje vytvářet design s pamětí integrovanou přímo na čipu. Xilinx na svých deskách poskytuje maximálně 10Mbit paměti v 36kbit velkých blocích, které podporují duální přístup.

3. MOŽNOSTI KONFIGURACE FPGA

K definování chování FPGA se používá jazyk HDL nebo schématické navrhování. Běžně užívaná HDL jsou VHDL a Verilog. Následuje generování technologického návrhu systému pomocí automatizovaného nástroje, který umí převést HDL do elektronického návrhu. Tento návrh se pak musí hodit k použité architektuře FPGA. Toho se docílí pomocí procesu nazývaného place-and-route, který bývá většinou dodávaný firmou dodávající příslušné FPGA. Jakmile je hotová kontrola platnosti všech cest (simulace, analýza časování a další ratifikační metody), je generován binární soubor pro konfiguraci FPGA.

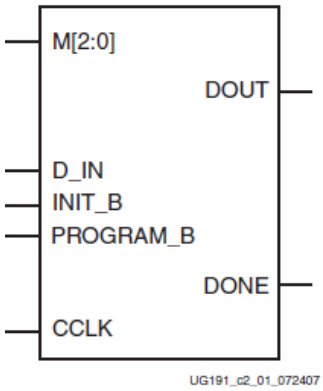
Při pokusu o zjednodušení komplexnosti navrhování v HDL, který se dá přirovnat k assemblerům, se zavedly abstraktní formy programování. V současné době se HDL podobá nejvíce standardům C a C++, které jsou rozšířeny o knihovny umožňující paralelní programování. Zároveň existují i knihovny komplexních funkcí a obvodů, které pomáhají urychlit navrhování. Tyto předdefinované obvody se obvykle nazývají IP cores a jsou k zakoupení u výrobců FPGA, případně firem, které se zabývají návrhem těchto obvodů. Kromě placených služeb lze také využít open source řešení, která jsou distribuována zadarmo, většinou pod licencí GPL nebo BSD.

3.1 KONFIGURAČNÍ ROZHRANNÍ

Všechny zařízení podporují některé z konfiguračních standardů, proto je nutné mít alespoň základní přehled o jejich funkčnosti a zapojení. Ilustrační obrázky jsou převzaty z brožury k zařízení Virtex V, ale metodicky platí i pro starší typy zařízení. Účelem této práce není dopodrobna rozebrat signály všech konfiguračních rozhranní, proto budou uvedeny pouze základy s příklady zapojení a primitivami. Nebude ale opomenuto srovnání odlišností v rozhranních pro jednotlivé rodiny FPGA firmy Xilinx.

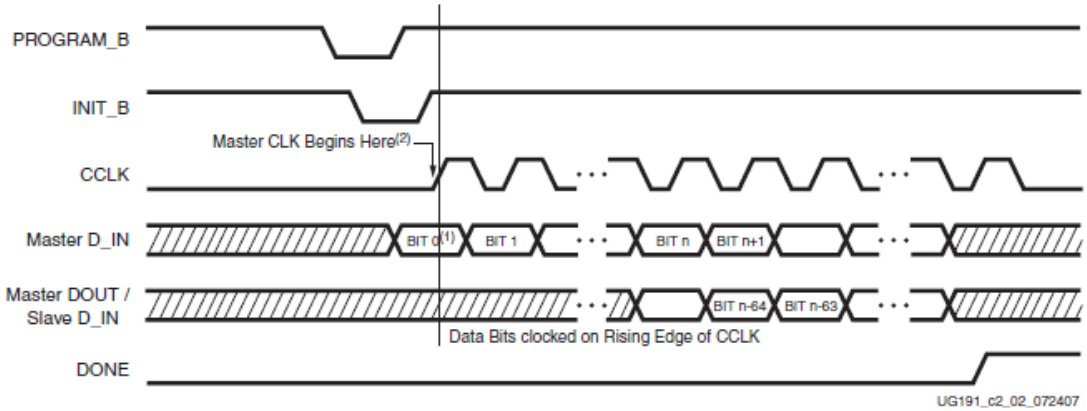
3.1.1 Sériové konfigurační rozhranní

V tomto módu je FPGA konfigurováno nahráváním jednoho konfiguračního bitu za cyklus CCLK. V Master módu je CCLK výstupem, v Slave vstupem. V praxi se používají čtyři metody sériové konfigurace (Master, Slave, Daisy-chain, Ganged).



Obrázek 3.1-1 Primitiva sériového konfiguračního rozhranní

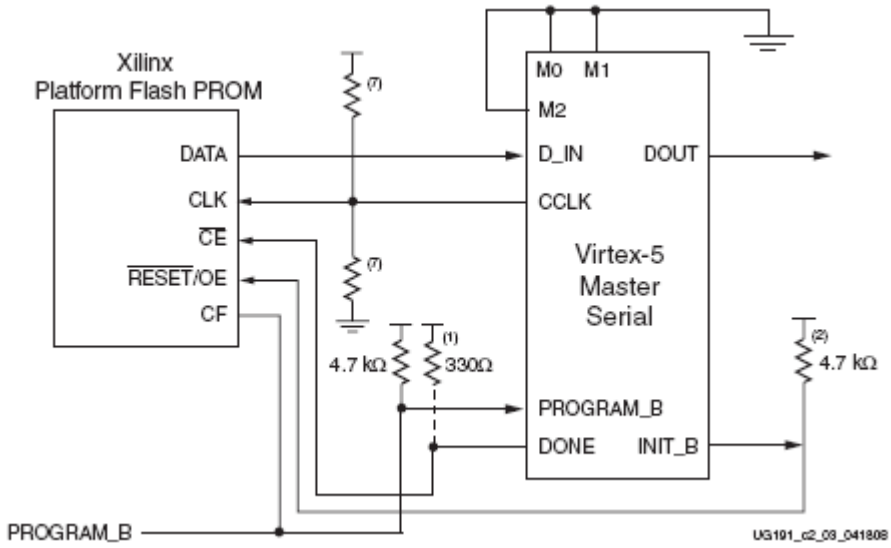
Master Serial mód je navržen tak, aby FPGA mohlo být nakonfigurováno z konfigurační PROM. Slave Serial mód se typicky využívá při více zařízeních spojených do uzavřeného řetězce nebo pokud ke konfiguraci dochází z vnějšího mikroprocesoru nebo CPLD. Serial Daisy-chain je metoda, kdy jsou zařízení seřazena do sériového řetězce a jsou postupně konfigurována jedním bitstreamem. Oproti tomu Ganged mód tyto zařízení seřazuje paralelně a proto je může konfigurovat pomocí jednoho bitstreamu všechny najednou.



Obrázek 3.1-2 Průběh signálů pro sériová konfigurační rozhranní

Jak je vidět na obrázku výše, datové bity jsou synchronizovány podle náběžné hrany CCLK signálu. Rozhodnutí, zda se konfigurace provádí v master nebo slave módu, přísluší pinu M[2:0]. Po zvolení módu je uvolněn INIT_B a může dojít ke konfiguraci. PROGRAM_B slouží jako asynchronní reset čipu. DONE vrací 1, pokud došlo ke správnému nakonfigurování zařízení.

Sériová konfigurační rozhranní jsou v podstatě to nejjednodušší řešení pro konfigurování FPGA a mezi jednotlivými rodinami čipů nejsou žádné výrazné rozdíly pro jejich využití. Konfigurace probíhá z externí paměti Xilinx Platform FLASH PROM, které se vyrábějí ve dvou verzích, lišících se v rychlosti (32/128MB), šířce sběrnice (8/16b) a velikosti paměti (320/800Mbs).

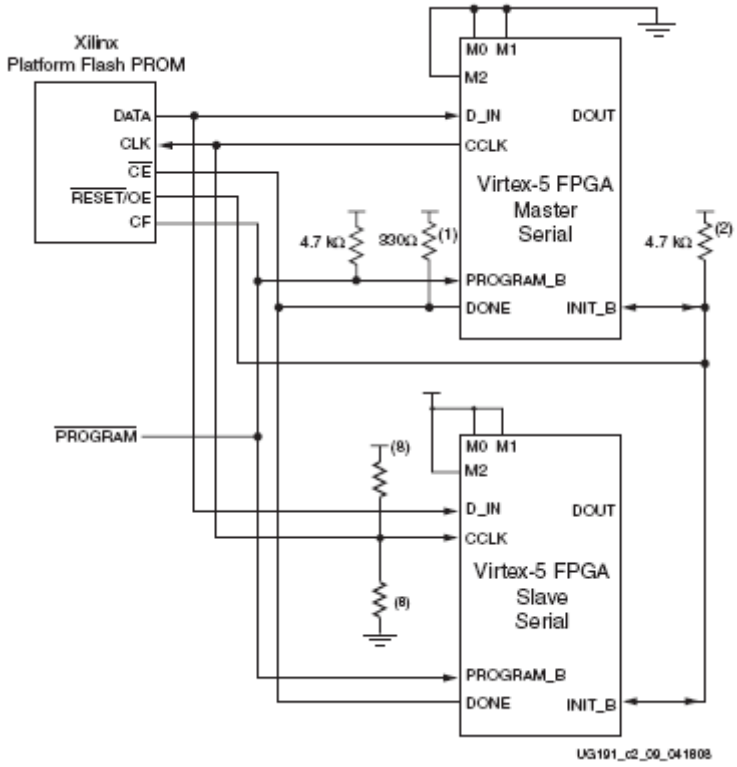


Obrázek 3.1-3 Sériové konfigurační rozhranní

3.1.2 SelectMAP

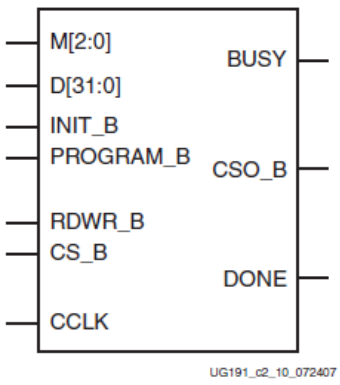
Rozhranní SelectMAP (nebo také paralelní konfigurační rozhranní) zajišťuje obousměrnou komunikaci s FPGA v 8-bit, 16-bit nebo 32-bit šířce. Výhodou je automatická detekce šířky sběrnice. SelectMAP opět umožňuje stejné čtyři metody konfigurace jako sériové konfigurační rozhranní, tedy Master, Slave, Daisy-chain a

Ganged. Ve své podstatě se jednotlivé metody neliší od sériového rozhranní, pouze zapojení jednotlivých bloků jsou jiná.

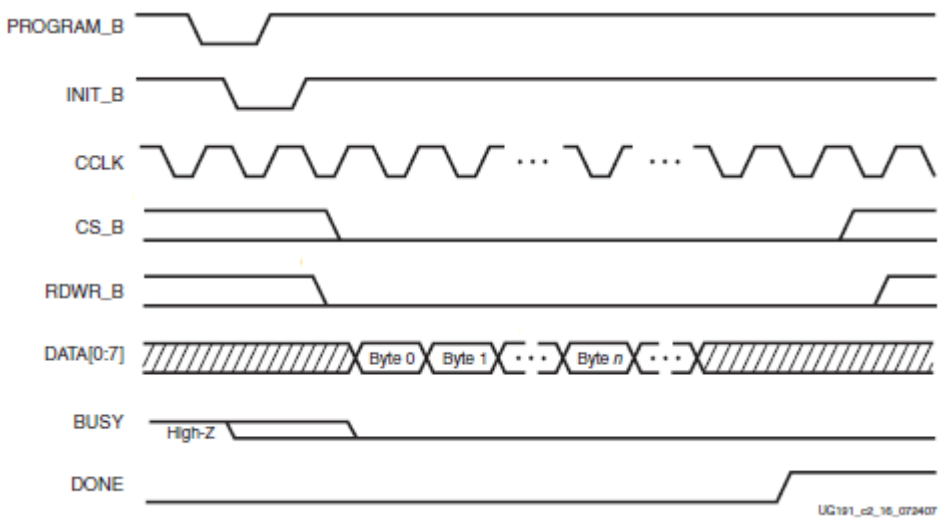


Obrázek 3.1-4 rozhranní SelectMAP

SelectMAP podporuje také buď přerušované nebo nepřerušované nahrávání dat. Přerušované nahrávání dat se používá v případě, že během konfigurace je potřeba dohrát dodatečná data. Zapuzování se provádí většinou zastavením CCLK signálu a jeho opětovným spuštěním po dohrání dat.



Obrázek 3.1-5 SelectMAP primitiva



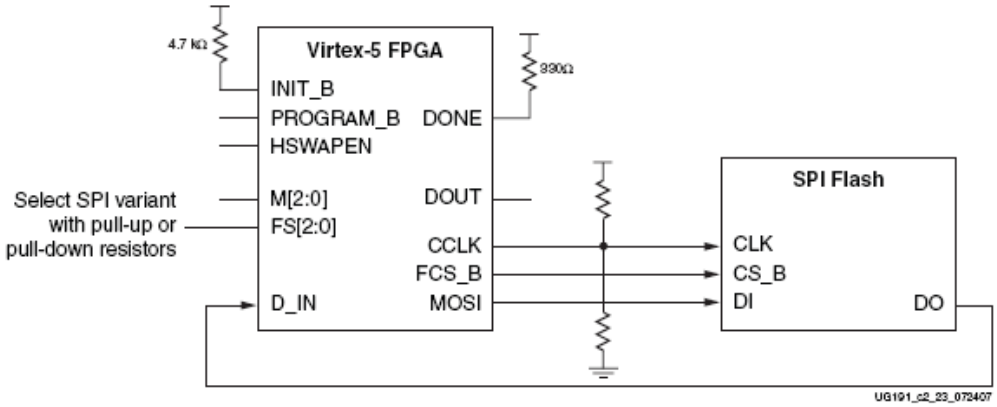
Obrázek 3.1-6 Průběh signálů SelectMAP při nepřerušovaném nahrávání dat

Pokud je konfigurováno pouze jedno zařízení, není potřeba využívat SelectMAP rozhranní pro konfiguraci více zařízení a pin CS_B=1. Další změnou oproti sériovému konfiguračnímu rozhranní je bit RDWR_B, který řídí zpětnou vazbu ze zařízení, tudíž rozhoduje, jestli datové piny budou vstupy (RDWR_B=0) nebo výstupy (RDWR_B=1). U přerušovaného módu se mění průběh CS_B signálu. Pokud je potřeba pozastavit konfiguraci zařízení, je CS_B=0 a konfigurace pokračuje po jeho opětovném nastavení na 1.

3.1.3 SPI

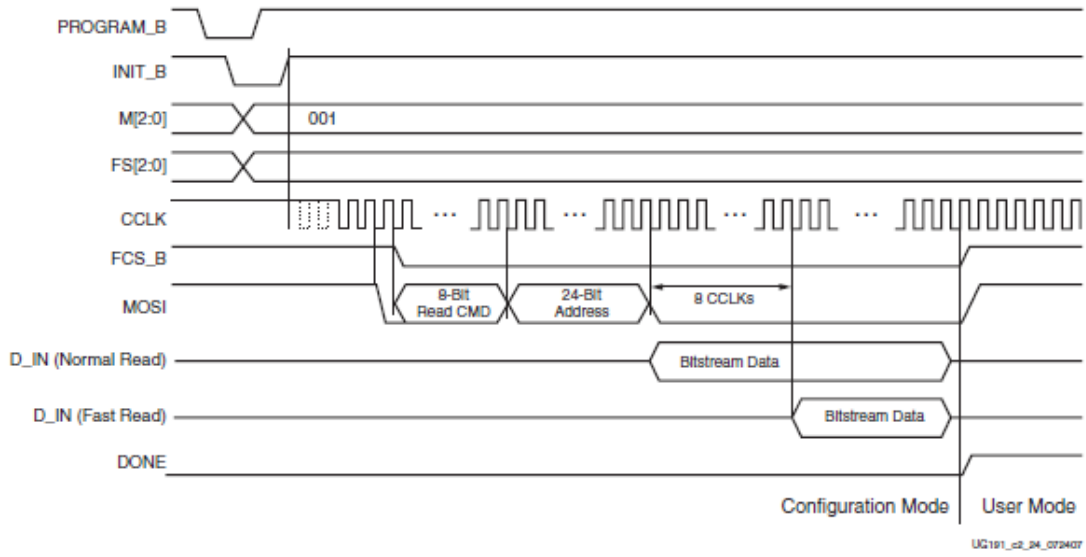
V SPI Flash módu konfiguruje zařízení samo sebe z dodávaného standardu SPI Flash PROM. Ačkoliv je standardně SPI 4-drátovým rozhranním, lze v praxi nalézt mnoho různých pamětí SPI Flash, které využívají své vlastní komunikační protokoly. Pro ilustraci je přiloženo schéma nejobvyklejšího konfiguračního rozhranní pro Virtex V.

Stejně, jako u předchozích módů, i SPI podporuje možnost sériového řazení více konfigurovaných zařízení metodou Daisy-chain. Principielně ovšem nepodporuje paralelní konfiguraci více zařízení z jednoho zdroje.



Obrázek 3.1-7 standardní zapojení pro rozhranní SPI

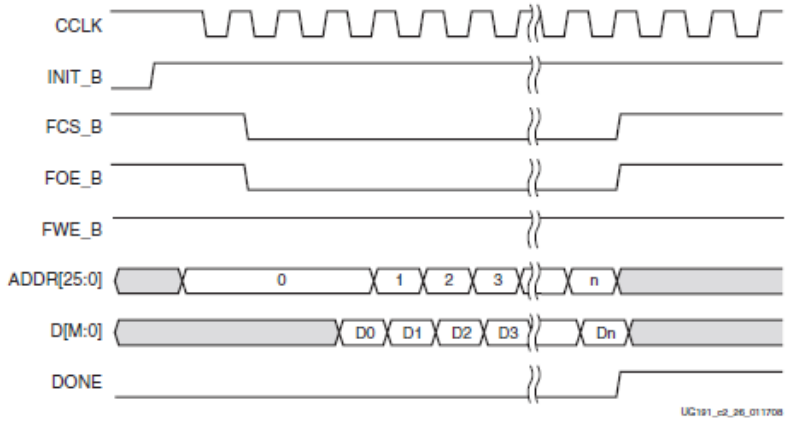
Oproti předchozím konfiguracím přibyly nové piny. FS[2:0] určuje, zda jsou při konfiguraci přítomny pull-up/pull_down rezistory, které umožňují uživateli řídit samokonfiguraci zařízení. Při konfiguraci pomocí SPI piny FCS_B a MOSI reagují na sestupnou hranu CCLK.



Obrázek 3.1-8 Průběh signálů pro SPI

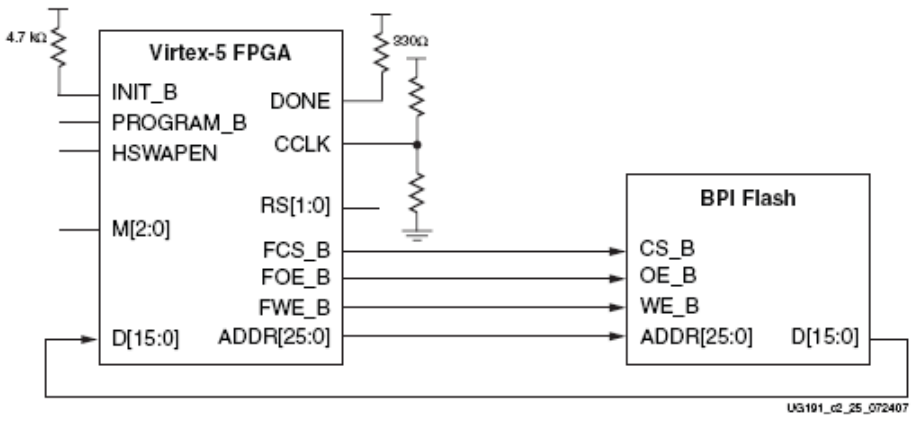
3.1.4 BPI

V módu BPI-Up (M[2:0]=010) nebo BPI-Down (M[2:0] = 011) konfiguruje zařízení opět samo sebe z paralelní paměti NOR Flash PROM. Je podporována buď 8bit nebo 16bitová sběrnice, přičemž její šířka je detekována automaticky. Vzorkování dat je prováděno při nástupné hraně CCLK.



Obrázek 3.1-9 Průběh signálů pro BPI

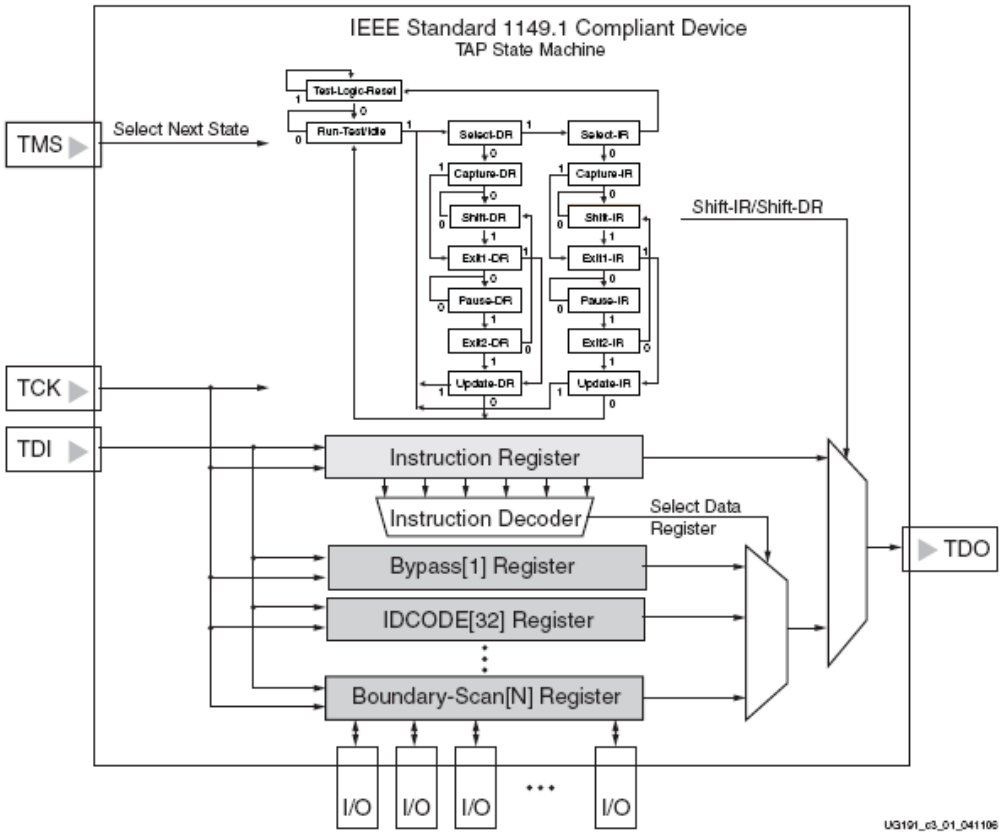
Při BPI-Up je počáteční adresa rovna 0 a inkrementuje se po 1, dokud není uplatněn pin DONE. Pokud dojde k přetečení adres, je generováno chybové hlášení a zařízení je zpětně rekonfigurováno. Identicky funguje BPI-Down mód, kdy se adresy dekrementují po jedné od 26h3 FFFFFF. Pokud dojde k podtečení paměti, opět je generováno chybové hlášení a proběhne zpětná rekonfigurace.



Obrázek 3.1-10 rozhranní BPI

3.1.5 JTAG

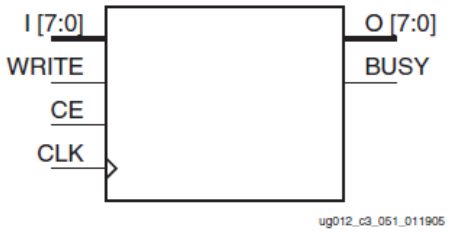
JTAG je obecně vžitý název pro standard IEEE1149.1 a 1532, kde 1532 je využito pro systémovou konfiguraci zařízení (ISC). JTAG je spojením Test Access Portu a architektury Boundary-Scan. Jako doplněk k standardním registrům obsahuje Boundary-Scan volitelné registry pro jednoduché testování a verifikaci a proto mohou mimo jiné posílat data na I/O piny, aby byla otestována komunikace mezi zařízeními.



Obrázek 3.1-11 Typická architektura JTAG

3.1.6 ICAP

Konfigurační rozhranní ICAP (Internal Configuration Access Port) pracuje na stejném principu jako SelectMAP s výjimkou toho, že ICAP má oddělené read a write datové sběrnice tam, kde SelectMAP používá obousměrné. Aby bylo možné ICAP používat, je potřeba v rámci návrhu definovat ICAP primitivu.



Obrázek 3.1-12 ICAP primitiva

Jak je vidět, tato primitiva koresponduje se SelectMAP rozhraním a její porty se chovají stejně jako SelectMAP. V následující tabulce jsou uvedeny signály ICAP primitivy v porovnání se SelectMAP pro čip Spartan 3A.

ICAP signál	SelectMAP signál	Směr	Popis
CLK	CCLK	Vstup	Hodiny ICAP
CE	CS_B nebo CSI_B	Vstup	Clock enable (aktivní na úroveň LOW)
WRITE	RDWR_B	Vstup	Read/Write přepínač (W=0, R=1)
BUSY	DOUT	Výstup	Indikátor zaměstnanosti (při read aktivní, při write pasivní)
I []	D []	Vstup	8-bytová vstupní datová sběrnice
O []	D []	Výstup	8-bytová výstupní datová sběrnice

Tabulka 3-1 Srovnání signálů ICAP a SelectMAP primitiv

Použití ICAPu se řídí několika omezeními:

ICAP a SelectMAP se nemohou nikdy po konfiguraci používat společně. Pokud je nastavený PERSIST bit na 1, ICAP se zakáže po náběžné hraně DONE pinu. Pokud je konfigurace napsána pro ICAP, je možné ji provést po SelectMAPu za předpokladu, že PERSIST bit je 0.

Pokud je používán zároveň s ICAPem i JTAG, nesmí během konfigurace pomocí ICAPu být používány některé instrukce JTAGu (*JTAG CFG_IN, CFG_OUT, JSTART,* a *JSHUTDOWN*). Po skončení práce JTAGu musí být desynchronizována veškerá konfigurační logika předtím, než se začne používat ICAP a to samé platí i opačně.

Jednoznačně největším přínosem ICAP rozhranní je ale možnost samokonfigurace. Pokud logika FPGA vyhodnotí, že je potřeba rekonfigurovat určitou část pole nebo přímo vyměnit modul, přes ICAP je schopná spustit proces částečné rekonfigurace bez vnějšího zásahu do chodu systému. Největší přínos této technologie je v zefektivnění chodu FPGA, kdy je nahrán pouze takový design, který je potřeba v daný moment. Samozřejmě je toto vykoupeno vyšší náročností na návrh top-level designu a jeho logiky.

3.2 BEZPEČNOST A OMEZENÍ

Základním problémem při konfiguraci zařízení je zajištění bezpečnosti a oprávnění přístupu k bitstreamů a architektuře pole. Na rozdíl od klasických procesorů neexistují pro FPGA reverzní asemblery. Tam kde procesor má svoji jasnou instrukční sadu, bitstream má miliony navzájem souvisejících bitů, což dělá jeho převod do pochopitelné formy extrémně složitým. Ačkoliv je takovéto dekódování bitstreamů nepoužitelné, stále existuje možnost kopírování bez nutného porozumění, co bitstream vlastně dělá. Pro tento případ je navrženo několik možností zabezpečení komunikace.

3.2.1 Bezpečnostní bity

Návrh CPLD je nahrán do paměti na chipu stejně jako u mikrokontrolerů. Obvykle je k dispozici tzv.: security bit, který znemožní čtení této paměti. Jde o jednoduchou ochranu proti kopírování, protože obsah paměti přestává být viditelný.

3.2.2 Kódování

FPGA nabízí oproti CPLD možnost kódování veškeré komunikace. V podstatě jde o to, že bitstream je nečitelný (jakoby poškozený) do doby, než je nahrán do FPGA obsahujícího správný klíč pro dekódování. V praxi se využívá encryptovací obvod, což je vestavěná funkce FPGA, která zabírá určitou část pole.

Pokud ji tedy uživatel nevyužívá, platí zbytečně část obvodu, kterou jinak neupotřebí. Encryptovací klíč může být do FPGA nahrán pouze přes JTAG. Jakmile je nahrán do zařízení, není již možné ho ze zařízení přečíst. Ke konfiguraci se poté dá použít pouze zakódovaný bitstream (uživatel specifikuje 256b klíč jako vstupní parametr pro BitGen).

Kódování je vysoce účinná metoda zabezpečení, proto je implementována zároveň se záložní baterií, aby nedošlo k odstavení enkryptovacího obvodu při vypnutí zařízení ze sítě. Možnou nevýhodou je to, že není možné po nakonfigurování FPGA pomocí zakódovaného bitstreamu nijak chránit aplikační data. Hlavním omezením pro použití je nutnost správy a distribuce klíčů.

3.2.3 Autentizace

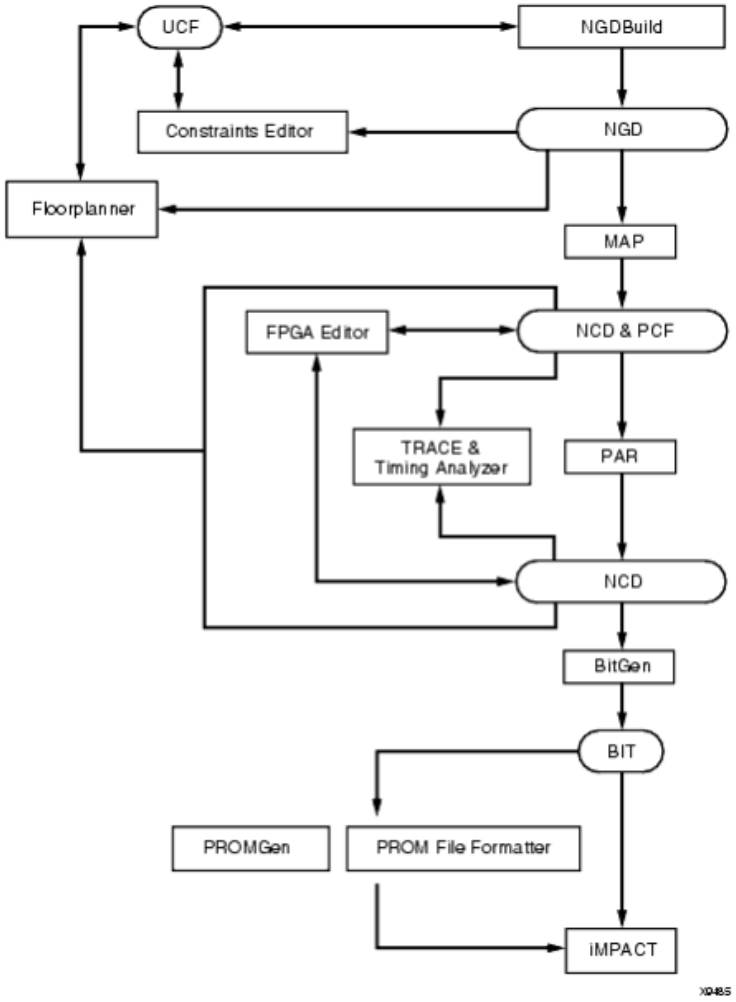
Autentizace je další široce využívaná metoda zabezpečení práce s FPGA. V podstatě jde o ověřování uživatele na začátku komunikace s FPGA. Aby byla autentizace maximálně efektivní, předpokládají se tyto dvě vlastnosti autentizačního kódu: jedinečnost a nemožnost duplikování, klonování nebo kopírování kódu. Tato metoda jako jediná chrání FPGA i po konfiguraci a je schopná zabezpečit i data aplikace.

	Bezpečnostní bity	**Kódování**	**Autentizace**
Typ zařízení	CPLD	FPGA	FPGA
Je bitstream viditelný po zabezpečení?	Ne	Pouze v zakódované podobě	Ano, ale použitelný pouze po autentizaci
Co se stane při nahrání neautorizovaného bitstreamu do FPGA?	N/A	Nedojde ke konfiguraci	Hlášení o chybě autentizace
Je možné zabezpečení výstupních dat?	Ne	Ne	Ano
Technická omezení	Potřeba hodně paměti na chipu	Správa kryptovacích klíčů	Vyžaduje logiku pro autentizaci

Tabulka 3-2 Možnosti zabezpečení bitstreamu

4. TVORBA BITSTREAMU

Na následujícím obrázku můžete vidět základní bloky postupu implementace a ověření funkčnosti návrhu. Na začátku je návrh převeden z konceptu do netlistu. Je mnoho způsobů, jak provést tento krok, jako například HDL návrhy, schématické návrhy, případně EDIF nebo XNF netlisty z dřívějšího jádra. Tyto vstupní metody dále vyžadují nástroj CAE, který vyprodukuje soubor formátu EDIF nebo XNF s obsahem sítě.



Obrázek 3.2-1 Xilinx design flow

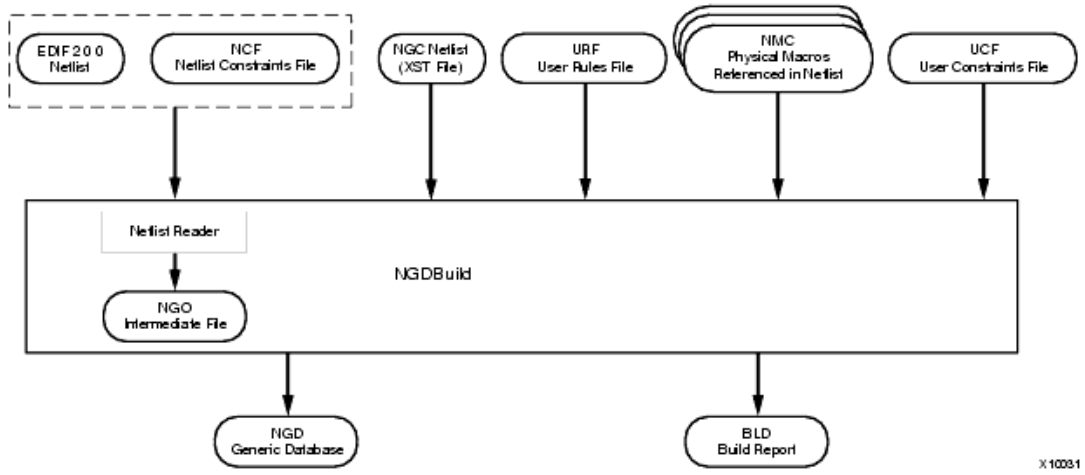
Implementace návrhu začíná konverzí netlistu do Native Generic Database (NGD) formátu a generováním konfiguračního bitstreamu pro FPGA. Tento krok zahrnuje optimalizaci a mapování, umístění a trasování a nakonec tvorbu bitstreamu. Tento postup jde udělat buď automaticky nebo manuálně pomocí nástrojů Xilinx Development System.

Verifikace návrhu obsahuje simulaci, analýzu časování a kontrolu obvodů. Simulace je ovšem provedena pomocí nástroje třetí strany, který je podporován firmou Xilinx. Vstupem do těchto nástrojů je specifický překlad NGD souboru do simulačního netlistu. Výhodou tohoto řešení je možnost simulace NGD kdykoliv během postupu návrhu. Analýza časování a verifikace obvodu je součástí Xilinx Development System.

4.1 JEDNOTLIVÉ KROKY IMPLEMENTACE

4.1.1 NGDBuild

Vykoná všechny nutné operace k přečtení netlistu ve formátu XNF nebo EDIF a vytvoření NGD popisující logický design redukovaný na primitivy. Zároveň zkontroluje návrh pomocí Logical DRC.



Obrázek 4.1-1 NGDBuild design flow

4.1.2 The User Constraints File (UCF)

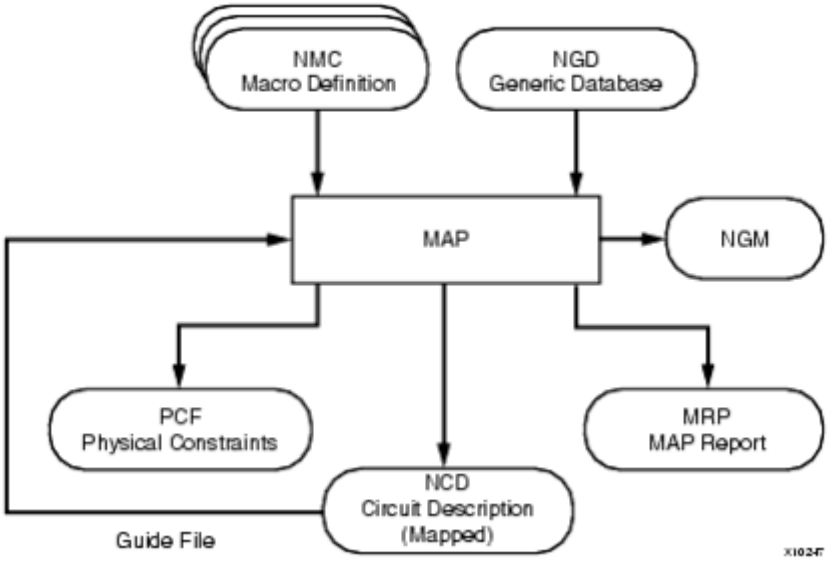
Jde o ASCII soubor s mapováním jednotlivých pinů a popisem implementace. Zahrnuje v sobě procesy Assign Package Pins a Create Area Constrains. Slouží jako jeden ze vstupů do NGDBuild.

4.1.3 The Logical Design Rule Check

Logical DRC je série testů spouštěných pro verifikaci návrhu popsaného v NGD (Native Generic Database). Spouští se automaticky na konci NGDBuild (než je zapsán .NGD) a NetGen (před vytvořením netlistu).

4.1.4 MAP The Technology Mapper

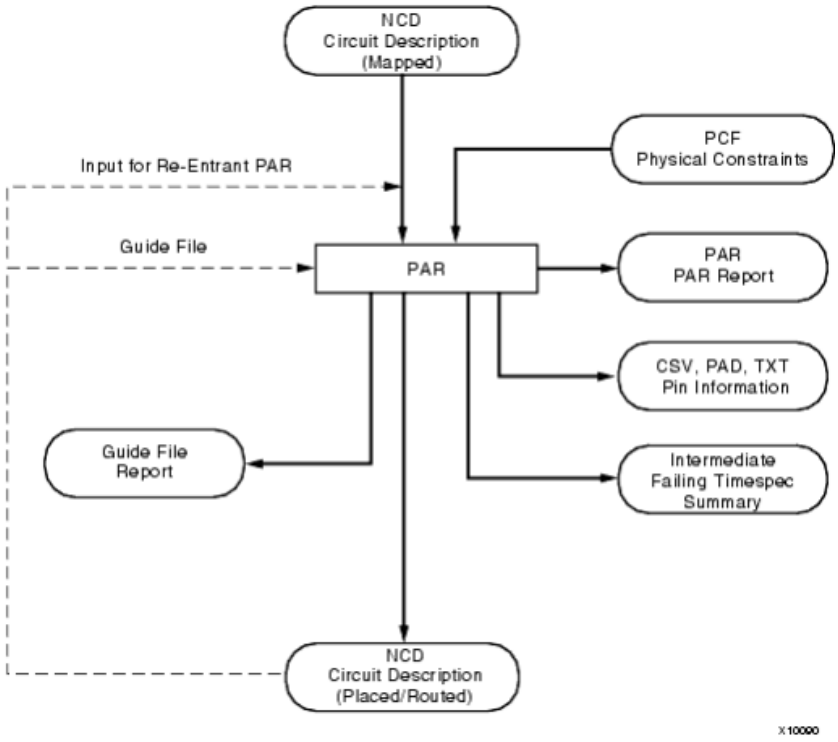
MAP mapuje logiku definovanou v NGD do jednotlivých částí FPGA (CLB, IOB nebo TBUF). Výstupem z MAP je NCD (Native Circuit Description), což je fyzická reprezentace návrhu namapovaná do jednotlivých komponent FPGA. I v této fázi probíhá Logical DRC.



Obrázek 4.1-2 MAP design flow

4.1.5 PAR – Place and Route

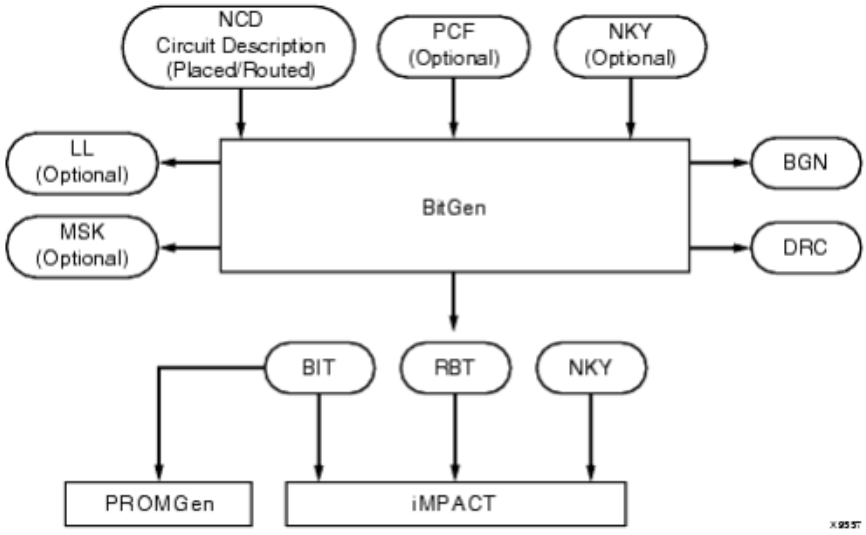
Po vytvoření NCD, PAR umístí a natrasuje jednotlivé bloky FPGA. Pod těmito pojmy je možné si představit rozmístění logických buněk, paměti nebo vstupů/výstupů do FPGA a vytvoření požadovaných vazeb mezi nimi. Jeho výstupem je opět NCD, který je použitelný generátorem bitstreamu BitGen.



Obrázek 4.1-3 PAR design flow

4.1.6 BitGen

Po natrasování je potřeba nakonfigurovat zařízení, aby vykonávalo požadovanou činnost. Toto je zajištěno právě BitGenem, který vytváří konfigurační bitstream pro implementaci návrhu. Jeho vstupem je tedy NCD z PAR a výstupem je binární soubor s příponou .bit. Jeho obsahem je mimo již uvedeného návrhu i kódování a tento soubor je možné nahrát buď přímo do FPGA (pomocí iMPACT) nebo do externí konfigurační PROM (pomocí PROMGen), kterou lze posléze konfigurovat více FPGA.



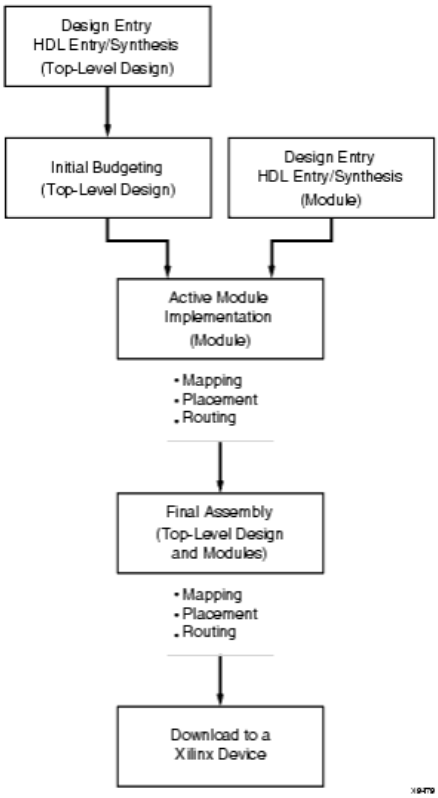
Obrázek 4.1-4 BitGen design flow

5. ZÁKLADY MODULÁRNÍHO NAVRHOVÁNÍ

Pro pochopení modulární částečné rekonfigurace, o které bude pojednávat jedna z hlavních kapitol, je nutná znalost modulárního navrhování systémů. Proto následující odstavce budou věnovány základům tohoto typu programování.

Modulární programování je víceúrovňový typ návrhu, kdy vedoucí týmu pracovníků vytvoří nejvyšší úroveň celého designu. Tento krok zahrnuje definování všech globálních proměnných, použitých I/O a rámcový návrh modulů, které jsou v tuto chvíli ve fázi tzv.: černé skříňky, kdy má modul přesně definovaný počet a typ vstupů a výstupů, je řečeno, co přesně má daný modul dělat, ale není hotový jeho vnitřní návrh.

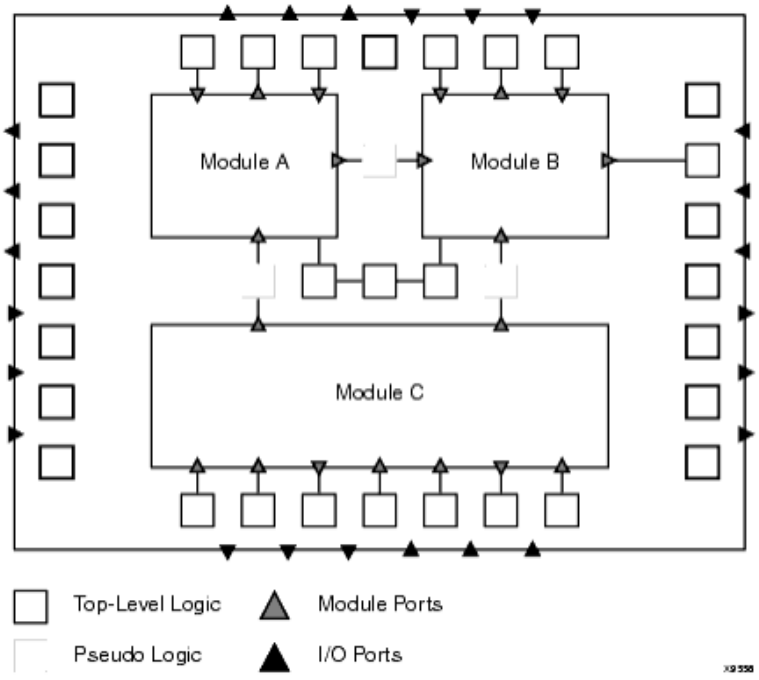
Na členech týmu je potom zadrátování jednotlivých modulů podle požadavku vedoucího teamu. Od této chvíle se návrh stává z globálního individuální záležitostí a pracovník si sám volí, jakým způsobem bude přidělený modul navrhovat.



Obrázek 4.1-1 Postup prací při modulárním navrhování

5.1 TOP-LEVEL DESIGN

Vedoucí pracovník musí vytvořit vstupní návrh nejvyšší úrovně projektu předtím, než může začít sestavování počátečního rámcového designu. To obsahuje návrh globální logiky, velikost a umístění každého modulu v rámci chipu, umístění vstupů a výstupů pro každý modul a inicializaci vnitřních hodin.



Obrázek 5.1-1 Top-level design

5.2 MODULY

Jednotlivé týmy programátorů vytváří vstupní návrh jednotlivých modulů. Drží se přitom předem definovaných vstupů a výstupů ze svých modulů, které byly již určeny dříve.

5.3 PŘÍMÁ IMPLEMENTACE MODULŮ

Ke každému modulu musí být vytvořeno jeho propojení s top-level designem. Pro každý modul se pouští *NGDBuild* v aktivním módu. *NGDBuild* načte top-level design, netlist modulu a top-level UCF jako vstup. Je generován NGD se

specifikovaným aktivním modulárním rozšířením. V tuto chvíli je možnost použití *Constraints Editoru* pro dodatečná omezení pro tento modul. Poté, co je modul připraven, posílá se vedoucímu týmu na implementaci do top-level designu. Takto nachystaný modul je označen jako PIM (physically implemented module) a následuje fáze finálního propojení modulů v rámci top-level designu.

5.4 KONEČNÉ PŘIŘAZENÍ

Pokud jsou moduly připraveny k použití, dochází k jejich implementaci do top-level designu a překladu celého projektu. Zajišťuje se propojení jednotlivých modulů v rámci projektu, odstranění přebytečných signálů a design je připraven k nahrání do zařízení.

Vlastní propojení probíhá pomocí *NGDBuild*. Ten čte top-level NGO a UCF a všechny PIM, aby bylo možné vše namapovat, umístit a natrasovat, přičemž se automaticky odstraní všechny nepoužívané výstupy a signály, což vede ke zvýšení výkonnosti celého systému.

6. ÚVOD DO REKONFIGURAČNÍCH TECHNIK

Částečná rekonfigurace je proces konfigurace části FPGA zatímco statická část zařízení stále běží. Hardware, stejně jako software, může být navržen modulárně vytvořením subkomponent a poté sdružován do celku. V mnoha případech je proto výhodné mít možnost odpojit jeden z modulů, zatímco zbytek systému bude stále plně funkční.

V běžné praxi rekonfigurace zařízení vyžaduje odstavení celého systému a následné nahrání celého designu. Částečná rekonfigurace proti tomu umožňuje ponechat v chodu kritické části modelu, zatímco je zvenku nahráván částečný design do požadovaného modulu. Částečná rekonfigurace může ale sloužit také k úspoře paměti, protože není nahráván kompletní design pole ale pouze jeho pozměněná část.

Běžným příkladem z praxe jsou komunikační zařízení. Pokud zařízení kontroluje více spojení, z nichž některé vyžadují kódování, je téměř nutností například při změně kódování provést zásah bez nutnosti odstávky celého systému.

Částečná rekonfigurace není podporována všemi FPGA. V současné verzi firmware podporuje Xilinx rekonfiguraci u Virtex II, Virtex II Pro, Virtex IV a Virtex V. Co se týká rodiny Spartan 3, modely Spartan 3A, Spartan 3AN, Spartan 3A DSP a Spartan 3E mají také implementovanou podporu částečné rekonfigurace. U všech těchto zařízení je vyžadován speciální software s podporou modulárního designu. Tyto moduly bývají typicky vytvářeny uvnitř definovaných rozhraní FPGA, která vyžadují speciální mapování do hardwaru. Z funkčního hlediska můžeme rozdělit částečnou rekonfiguraci na dvě základní skupiny.

6.1 ZÁKLADNÍ ROZDĚLENÍ Z HLEDISKA FUNKČNOSTI ZAŘÍZENÍ

6.1.1 Dynamická částečná rekonfigurace

Pod tímto pojmem se také skrývá aktivní částečná rekonfigurace. Předpokládá se, že během výměny části designu zbytek FPGA stále běží.

6.1.2 Statická částečná rekonfigurace

Zařízení není aktivní během rekonfiguračního procesu, i když je stále zapnuté. Zatímco jsou nahrávána data do FPGA, zbytek zařízení je v shutdown módu a je probuzen po skončení konfigurace.

6.2 ZÁKLADNÍ ROZDĚLENÍ Z HLEDISKA ROZSAHU ZMĚN

6.2.1 Modulární částečná rekonfigurace

Předpokládá se možnost změny celých modulů designu. Pro zajištění komunikace během výměny kódu v modulu je nutné připravit speciální makro, které zajistí pozdější komunikaci modulu se zbytkem FPGA. Při modulární částečné rekonfiguraci je nutné postupovat podle jasných směrnic, protože pro každý modul se vytváří separátní bitstream, který musí být schopný začlenění po skončení rekonfigurace.

6.2.2 Rozdílová částečná rekonfigurace

Používá se, pokud měníme pouze malé bloky programu. Je velice užitečná při změnách rovnic v LUT nebo změnách v obsahu pamětí. Tímto typem rekonfigurace se budeme zabývat blíže v následující části.

7. ROZDÍLOVÁ ČÁSTEČNÁ REKONFIGURACE

Existují dva základní postupy pro přizpůsobení nebo pozměnění designu částečnou rekonfigurací. Design se může změnit buď postupem front-end (přímá změna HDL kódu nebo schématu) nebo jako back-end (odpovídá změnám pouze v generovaném NCD souboru). Hlavní rozdíl spočívá v tom, že při postupu front-end musí být design znovu syntetizován a implementován, aby se změny projevily v nově vytvořeném NCD. Oproti tomuto zdlouhavému postupu má back-end výhodu v tom, že je modifikováno již existující NCD, čímž odpadá fáze překladu HDL nebo schématu a trasování nového NCD. K úpravě existujícího NCD se přistupuje přes FPGA Editor. Příkaz BitGen potom může generovat pouze rozdílové bitstreamy, ve kterých jsou uloženy informace pouze o provedených změnách a pomocí kterých dochází k rekonfiguraci pouze pozměněných částí.

Pozměnit konfiguraci modulu z jedné implementace na druhou je tedy velmi rychlé, protože rozdílové bitstreamy jsou menší než bitstreamy měnící celé zařízení. Mohou být nahrány do zařízení rychle a hlavně snadno díky jejich velikosti.

Pokud je potřeba měnit větší logické bloky systému, využívá se modulární částečné rekonfigurace, která bude popsána později v této práci. Co všechno tedy vlastně lze měnit? Nejjednodušším příkladem mohou být přímé změny logiky v LUT a změny I/O standardů. Tento typ změn se provádí přímou editací NCD v Xilinx FPGA Editoru. Pokud je potřeba modifikovat BRAM, je k dispozici utilita Data2MEM, případně lze opět použít FPGA Editor.

Poté, co jsou uloženy změny, použije se BitGen pro vygenerování rozdílového bitstreamu, který porovnává původní a nově vygenerovaný bitstream a uloží si pouze rozdílné části a jejich umístění v poli. Záleží pouze na typu a množství změn, ale ve většině případů je tento bitstream menší, než bitstream původní. Vše, co je nutné k pozměňování NCD, je pochopení a znalost dělání logických změn v FPGA Editoru a příslušné příkazy pro práci s BitGenem.

7.1 PROVÁDĚNÍ MALÝCH ZMĚN V DESIGNU POMOCÍ FPGA EDITORU

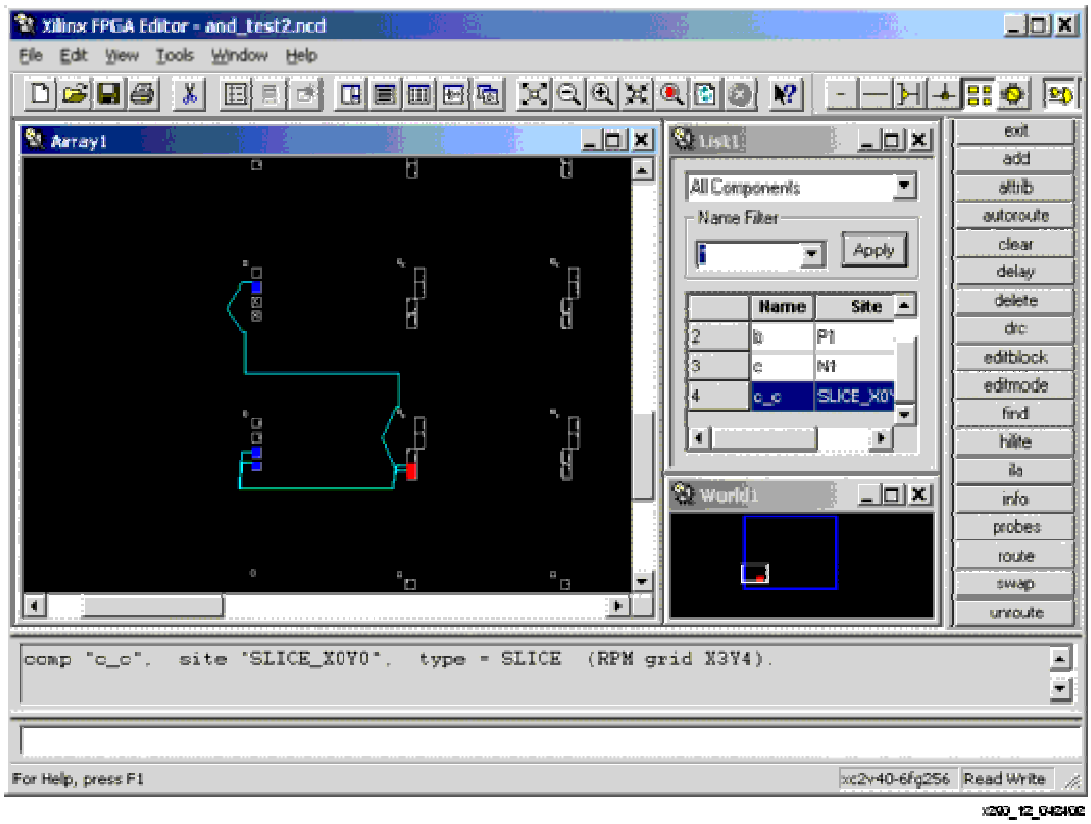
FPGA Editor umožňuje mnoho různých druhů změn, ale zaměříme se pouze na tři základní: změna I/O standardů, obsahu BRAM a programování LUT. I když je možné měnit například i trasy, tyto změny se nedoporučují dělat kvůli existující možnosti vnitřní kolize během rekonfigurace.

Jakmile je v editoru otevřen soubor s natrasovaným NCD, okamžitě uložte tento soubor pod jiným jménem před provedením jakýchkoliv změn, aby nedošlo ke ztrátě původních dat. V textu se objeví i ukázkové příklady použití, proto NCD byl přejmenovaný z *and_test.ncd* na *and_test2.ncd*. Nyní je nutné umožnit FPGA Editoru provádět změny ve schématu. Toho dosáhnete volbou File→Main Properties a změnou Edit Mode na Read Write.

7.1.1 Změna LUT logiky

Nejmenší možný element, který je možné měnit, je logika. Napřed je nutné najít a zobrazit příslušný logický blok například pomocí tlačítka „find“ z panelu na pravé straně okna editoru. Označení bloku je indikováno červenou barvou ve schématu a je možné přejít pomocí „editblock“ k jeho editaci.

Aby se zabránilo náhodné editaci, je ve výchozím nastavení zakázána editace logiky. Pokaždé, když je blok otevřen, je třeba použít tlačítko „Begin Editing,“ pro umožnění provádění změn. Díky tomuto postupu se změní šedá barva pozadí okna na černou a je možné editovat rovnice v LUT.



Obrázek 7.1-1 Zobrazení logického bloku

Po stisknutí tlačítka „F=“ se otevře okno se jménem bloku a dvěma rovnicemi. Nyní je možné měnit logiku bloku pomocí následujících operací

*** → Logický AND**

+ → Logický OR

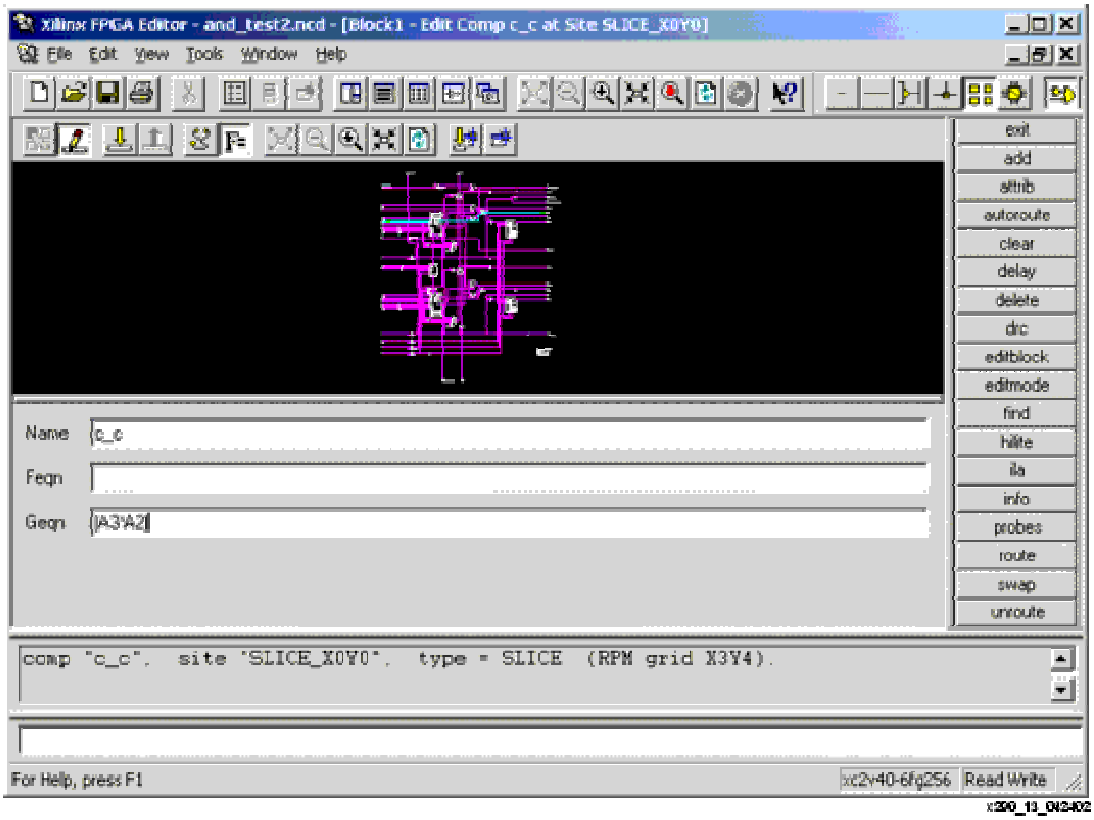
@ → Logický XOR

~ → Unární NOT

Platné proměnné v rovnicích jsou A1, A2, A3 a A4, které representují čtyři adresované vstupy do LUT. V rovnici je možné použít též závorky pro sdružování více příkazů, např.: (A4 * A1) @ ~A3). Použití jakýchkoliv jiných operátorů nebo

operandů vyústí v chybové hlášení. Pro příklad uvádím ukázku chybového hlášení, ve kterém je odkaz na nemapovatelnou strukturu mynet:

ERROR:FPGAEDITOR:24 - "(A3*~A2 + mynet) is not a valid value for the Geqn attribute."



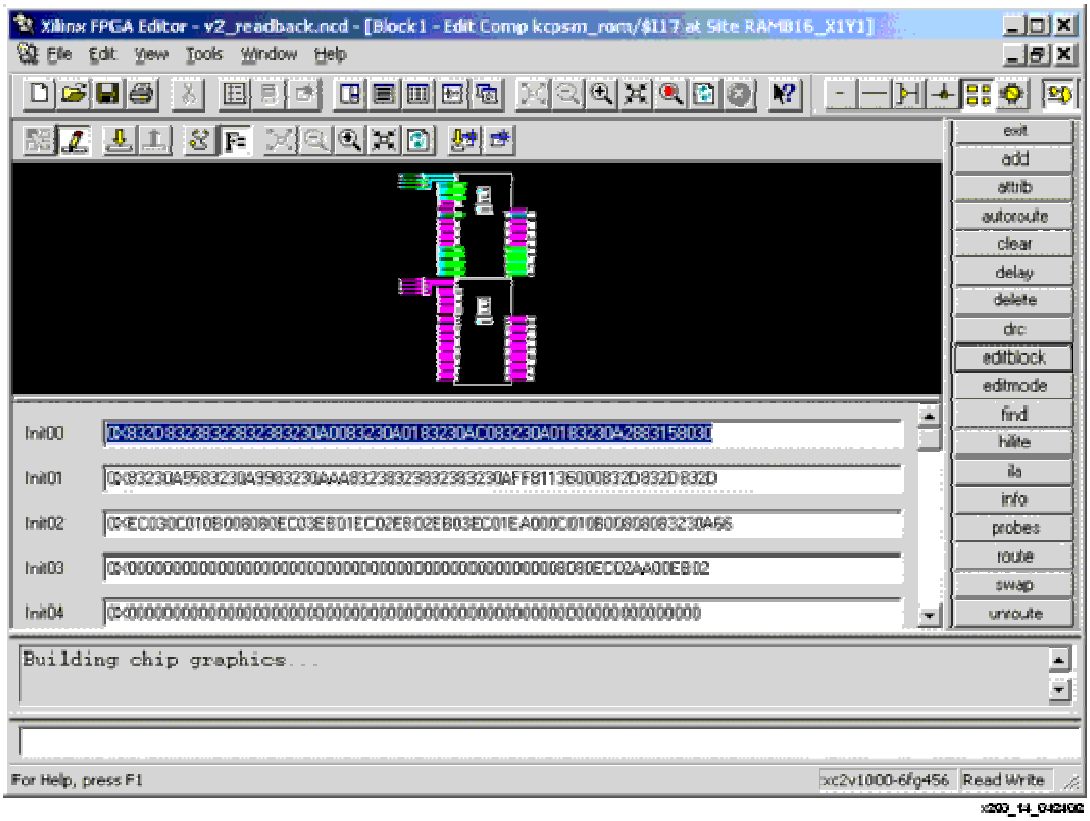
Obrázek 7.1-2 Změna logické funkce bloku

Jakmile jsou provedeny požadované změny, stiskněte Save Changes a Close Window pro uzavření Block Editor.

7.1.2 Změna obsahu Block RAM

Editor pro blokovou RAM je velmi podobný předchozímu editoru a platí pro něj stejný postup jako pro měnění logiky v LUT. Datový formát, ve kterém je zapsán obsah paměti odpovídá INIT vymezením v souboru UCF.

Opět po dokončení editace stiskněte tlačítka Save Changes a Close Window pro návrat do náhledu na celé pole.



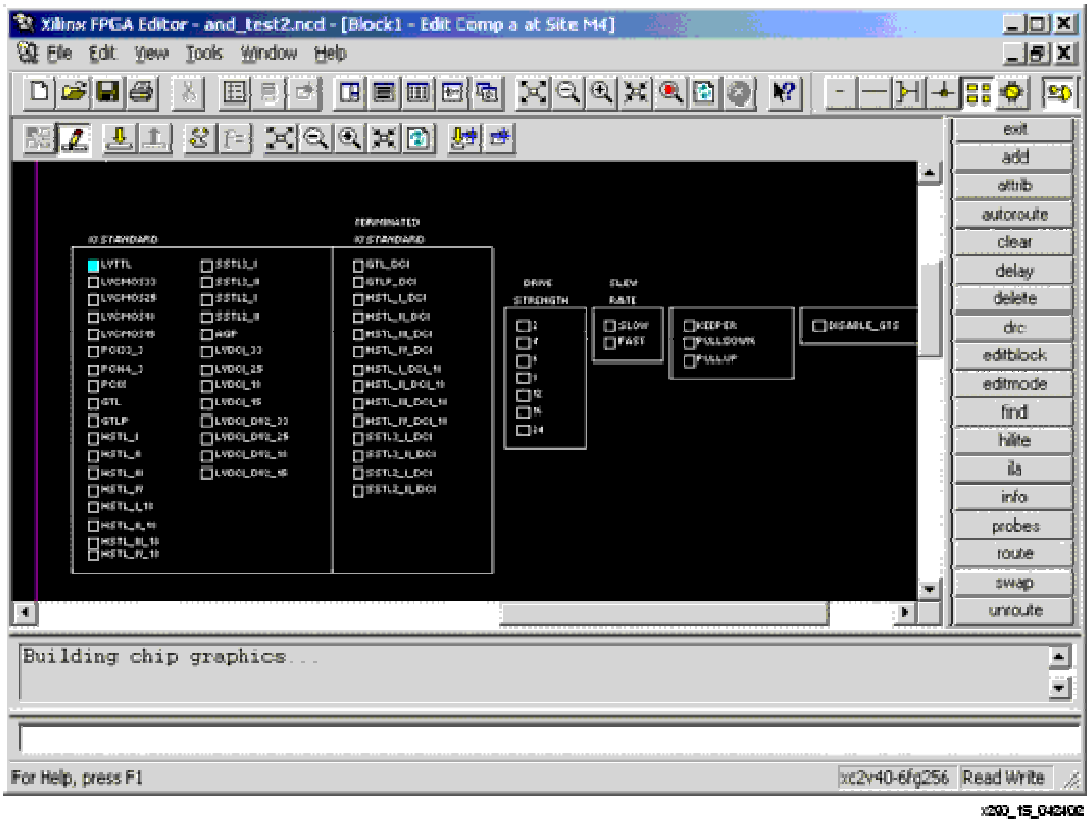
Obrázek 7.1-3 Změna obsahu blokové paměti RAM

7.1.3 Změna I/O atributů

Pro změnu I/O atributů otevřete „Block Editor“ opět stejným způsobem jako u BRAM. I/O standardy jsou umístěny v boxu v pravém horním rohu okna, jak je ukázáno na následujícím obrázku. Je třeba také zatrhnout rámeček vedlo požadovaného I/O standardu.

Po změnách je nutné porovnat VREF napětí (případně jejich absenci) s ostatními I/O zařízeními v bance, jinak změny nebudou správně fungovat.

Například není možné změnit LVTTL I/O uprostřed banky LVTTL na GTL, protože GTL vyžaduje referenční napájení, což LVTTL nepodporuje.



Obrázek 7.1-4 Změna I/O standardů

7.1.4 Další měnitelné elementy

Aktivní částečná rekonfigurace umožňuje kromě již přiblížených možností měnit například polaritu, inicializaci překlápění obvodů nebo resetovaní hodnoty, případně změny módů zápisu do RAM.

7.2 VYTVOŘENÍ ROZDÍLOVÉHO BITSTREAMU

Pro aktivní částečnou rekonfiguraci je potřeba nastavit -g ActiveReconfig:Yes, který vyjadřuje připravenost zařízení pro nahrání nového bitstreamu za plné funkčnosti zařízení. Dodatečně je vyžadováno -g Persist:Yes,

pokud je prováděna rekonfigurace přes SelectMAP mód. Tento příkaz umožní zachování mapování SelectMAP pinů po skončení konfigurace zařízení, aby bylo možné použít SelectMAP interface pro pozdější možnost rekonfigurace

Rozdílový bitstream může být vytvořen s jakýmkoliv dalším příkazem specifikujícím chování BitGenu, který neobsahuje šifrování. Stejně tak nemůže být zařízení částečně rekonfigurováno pomocí šifrovaného bitstreamu.

Pro naši potřebu je nejdůležitější atribut –r. Pokud je tento použit správně, vytvoří bitstream, který obsahuje pouze rozdíly mezi nově trasovaným NCD a starým bitstreamem. Jako příklad uvádím

bitgen -g ActiveReconfig:Yes -g Persist:Yes -r and_test.bit and_test2.ncd and_test2_partial.bit

Tímto příkazem byl vytvořen konfigurační soubor and_test2_partial.bit, který konfiguruje v zařízení pouze rámce, které jsou rozdílné mezi and_test a and_test2. JE NUTNÉ, aby před nahráním tohoto souboru bylo již zařízení naprogramováno pomocí původního bitstreamu.

8. MODULÁRNÍ ČÁSTEČNÁ REKONFIGURACE

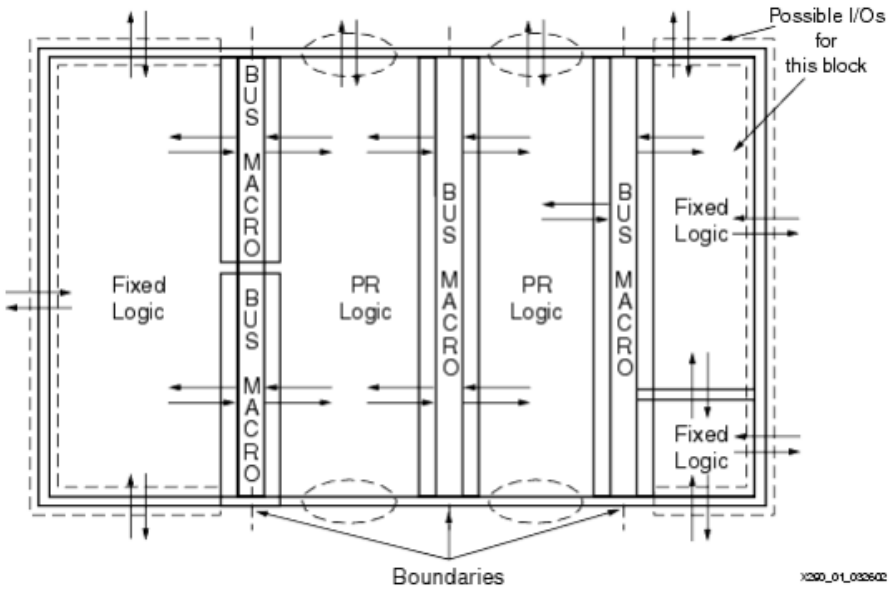
Pro pochopení tohoto typu částečné rekonfigurace je nutná znalost modulárního navrhování systémů. Oproti rozdílové rekonfiguraci, kdy se mění pouze jednoduché části designu (LUT, BRAM, atd.…), modulární rekonfigurace vyžaduje specifický vzhled modulu, aby jej bylo možné rekonfigurovat.

8.1 DEFINICE A OMEZENÍ REKONFIGUROVATELNÝCH MODULŮ

Pro správnou funkci modulární částečné rekonfigurace je nutné dodržet soubor pravidel pro návrh celého FPGA.

- Výška všech modulů musí být vždy stejná a neměnná
- Šířka modulu je w = 4k, kde k>0
- Veškerá logika modulu je považována za součást rekonfiguračního bitstreamu, což zahrnuje nejen TBUF, BRAM, ale hlavně veškeré trasování
- Časovací logika (BUFGMUX, CLKIOB) nesmí být součástí modulu, hodiny mají vlastní bitstream
- IOB spojené s modulem jsou součástí speciálního rekonfiguračního bitstreamu
- Pro zjednodušení rekonfiguračních procesů je potřeba minimalizovat počet rekonfigurovaných modulů. Ideální je, aby jela rekonfigurace pouze jednoho modulu najednou. Z toho vyplývá zároveň maximální počet rekonfigurovaných modulů na počet sliců / 4
- Pozice modulů se nesmí nikdy měnit
- Komunikace fixního a rekonfigurovaného modulu je zajištěna pomocí bus makra
- Celkový design systému musí být vyřešen tak, aby během rekonfigurace jednoho z modulů na něm nebyly závislé další

- Stav paměťových elementů uvnitř modulu je během rekonfigurace chráněný. Nelze ale použít pro inicializaci modulu globální set/reset, tzn.: je potřeba definovat uživatelský set/reset signál.

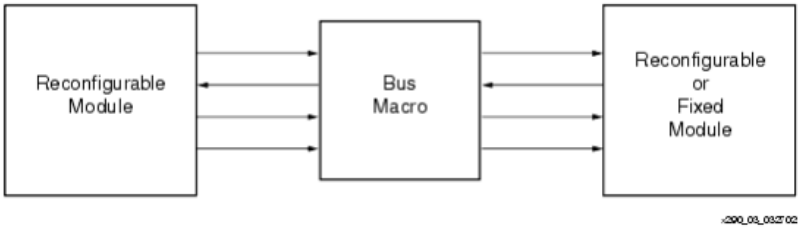


Obrázek 8.1-1 Návrh FPGA s rekonfigurovatelnými moduly (PR Logic)

8.2 POŽADAVKY NA STRUKTURU HDL

Kromě omezení a striktnosti návrhu modulů existuje i soubor pravidel a doporučení pro vlastní HDL kód.

- Měla by být dodržena struktura top-level design + moduly, tzn.: žádná další logika v top-level designu kromě modulů
- Pro komunikaci mezi fixními a konfigurovanými moduly se využívá výhradně bus makra. Každé makro používá 4bit komunikaci, takže pokud modul A komunikuje s modulem B přes 32bitů, je potřeba pro zajištění funkčního propojení 32/4 maker.



Obrázek 8.2-1 Komunikace modulů přes bus makro

- Signály, které se používají běžně pro komunikaci mezi moduly musí zůstat během rekonfigurace neaktivní a jejich funkci přebírají makra
- Moduly mohou sdílet pouze hodiny. Napájení, zem, atd. musí mít každý modul vlastní

8.3 DESIGN FLOW PRO ČÁSTEČNOU REKONFIGURACI

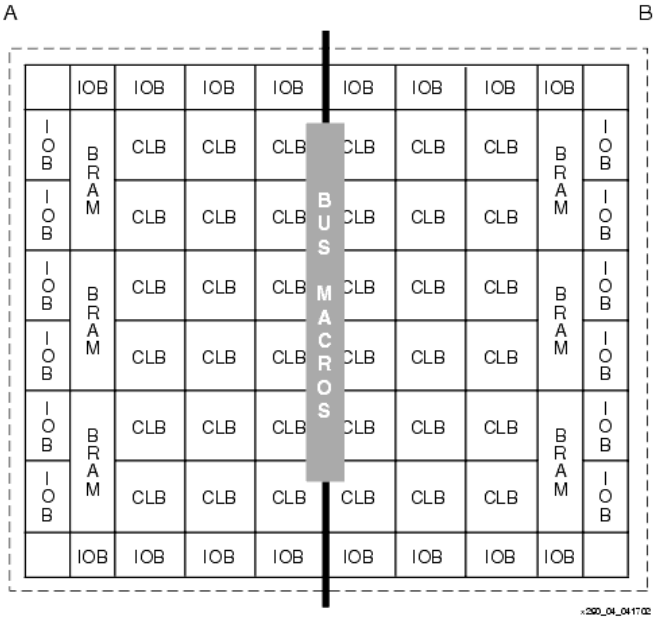
Design flow modulární rekonfigurace je v podstatě stejný jako u částečné rekonfigurace s rozdílem pouze ve formě rekonfiguračních bitstreamů. Následují kroky potřebné k úspěšné rekonfiguraci modulů.

- Vstupní návrh
  - HDL kód napsat podle pravidel pro konfigurovatelné moduly
- První sestavení
  - Nastavení omezení logiky, časování, atd.… pro top-level design i jednotlivé moduly
- Aktivní implementace
  - NGDBUILD, MAP, PAR
  - Dělá se pro každý modul a každou jednotlivou konfiguraci tohoto modulu
- Překladová fáze
  - Dělá se buď jen pro vstupní nastavení (nedoporučuje se) nebo pro všechny možné konfigurace fixních a rekonfigurovaných modulů (pro simulace a ověřování)

- Verifikace návrhu
  - Simulace a statická analýza
- Ruční kontrola rozhraní
  - V FPGA editoru ověřit, že neexistují neočekávaná křížení tras
  - I když se software snaží tomuto předejít, je potřeba pokaždé zkontrolovat výsledek ručně
- Generování bitstreamu pro kompletní design
  - Vstupní konfigurace při zapnutí
- Generování bitstreamu pro konfigurované moduly
- Nahrání bitstreamu se vstupní konfigurací
- Přeprogramování potřebných modulů
  - Použije se individuální nebo částečný bitstream pro tento modul

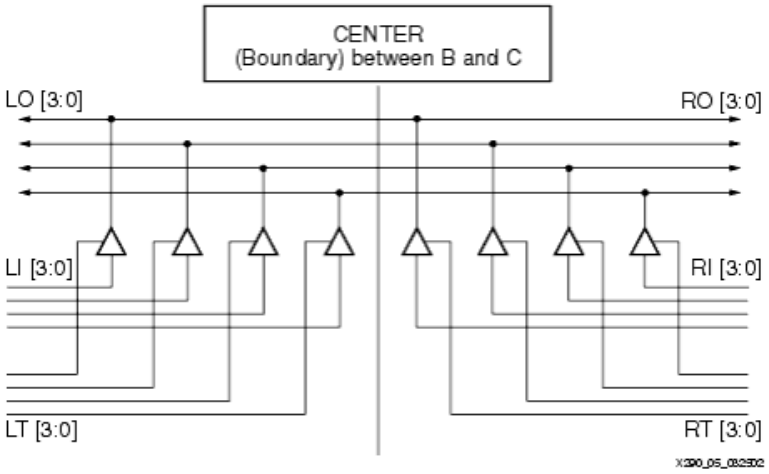
8.4 KOMUNIKACE PŘES BUS MAKRO

Jak již bylo uvedeno, nutnou podmínkou správné funkce bus makra je fixní trasování, které musí zůstat neměnné před i po rekonfiguraci.



Obrázek 8.4-1 Bus makro

Na obrázku výše jsou znázorněny dva rekonfigurovatelné moduly A a B. Pro jejich vzájemnou komunikaci je tedy vytvořeno bus makro. Jak bylo dříve řečeno, částečná rekonfigurace vyžaduje, aby signály, které budou zajišťovat komunikaci mezi moduly během rekonfigurace, byly pevně natrasovány. Jak je tedy vidět na obrázku, bus makro je tedy vlastně pevný trasovací most mezi jednotlivými moduly. Jde o předpřipravené makro, které natvrdo natrasuje komunikační kanály a nemění se kompilaci od kompilace. Pokud by se toto makro nepoužilo, signály se nepřiřadí správně a zničí se efektivní komunikace mezi moduly, což ale nemusí vést k chybovým hlášením.



Obrázek 8.4-2 Fyzická reprezentace bus makra

Signály v makru mohou být jedno nebo obousměrné. Doporučuje se, aby se po nastavení signálů již neměnil jejich směr pro celý návrh. Počet kanálů makra je omezený horizontálním rozměrem každého CLB.

8.5 IMPLEMENTACE MODULÁRNÍHO DESIGNU

Jak již bylo řečeno v kapitole o základech modulárního navrhování, implementační proces je rozdělen do tří hlavních kategorií.

- Initial budgeting (rozvržení a omezení pro celkový návrh)
- Active module (umístění a trasování jednotlivých modulů)
- Final assembly (propojení modulů do kompletního návrhu)

Podíváme se nyní na specifika modulární částečné rekonfigurace pro jednotlivé fáze. Většina z nich již byla uvedena dříve, proto by tato podkapitola měla sloužit jako souhrn poznatků.

8.5.1 Initial budgeting

- Rozmístění modulů
  - Šířka minimálně 4 slicy případně jejich násobek
  - Plné využití výšky celého zařízení
  - Určení rekonfigurovatelných modulů
- Rozmístění IOB
  - Umístění všech IOB musí být fixní
- Globální logika
  - Logika, která není součástí modulů, musí být správně přiřazena k top-level designu
  - Většinou tuto část vyřeší floorplanner sám
- Lokální vazby se manuálně přidají do každého bus makra (floorplanner nedělá automaticky)
- Vytvoření globálního časování

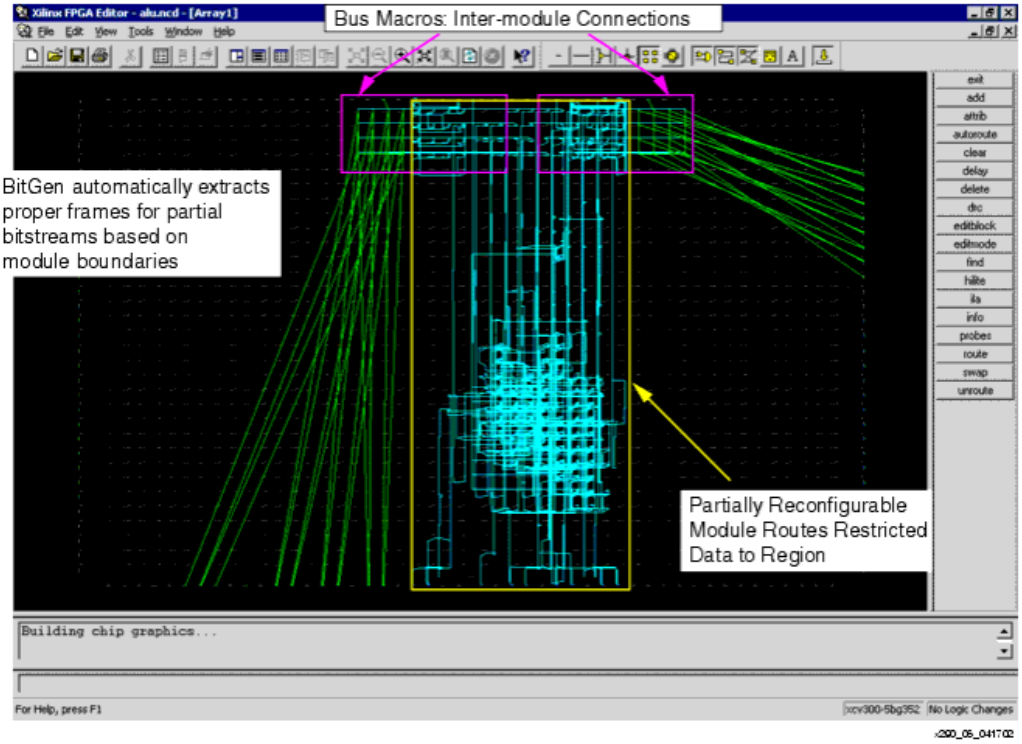
Výstupem této fáze je tedy UCF, které obsahuje vazby na rozmístění modulů a globální logiky. Každý modul se tedy poté implementuje pomocí tohoto UCF.

8.5.2 Active modul

Po Initial budgeting máme tedy připravený návrh na rozmístění jednotlivých modulů. Nyní je tedy potřeba jednotlivé moduly zasadit a natrasovat v rámci celkového návrhu. Moduly je možné implementovat nezávisle na sobě i paralelně. Navíc se generují bitstreamy pro rekonfigurovatelné moduly. Nyní se blíže podíváme na implementování jednoho modulu.

8.5.2.1 Implementace modulu

- Zkopírování UCF z Initial budgeting fáze do adresáře s modulem
- Upravit vazby v UCF specificky pro tento modul
- V Constrains Editoru můžeme vytvořit vnitřní časování modulu, pokud je potřeba
- Provést *ngdbuild, map, par, bitgen* a *pimcreate*. Toto vede k začlenění modulu do top-level designu a k vytvoření bitstreamu pro modul
- Pokud je nutná simulace modulu, provést *netgen*
- V FPGA Editoru zkontrolovat, jestli trasování modulu nevybočuje z rozměrů daného modulu, samozřejmě s výjimkou vazeb vedoucích do vedlejších modulů



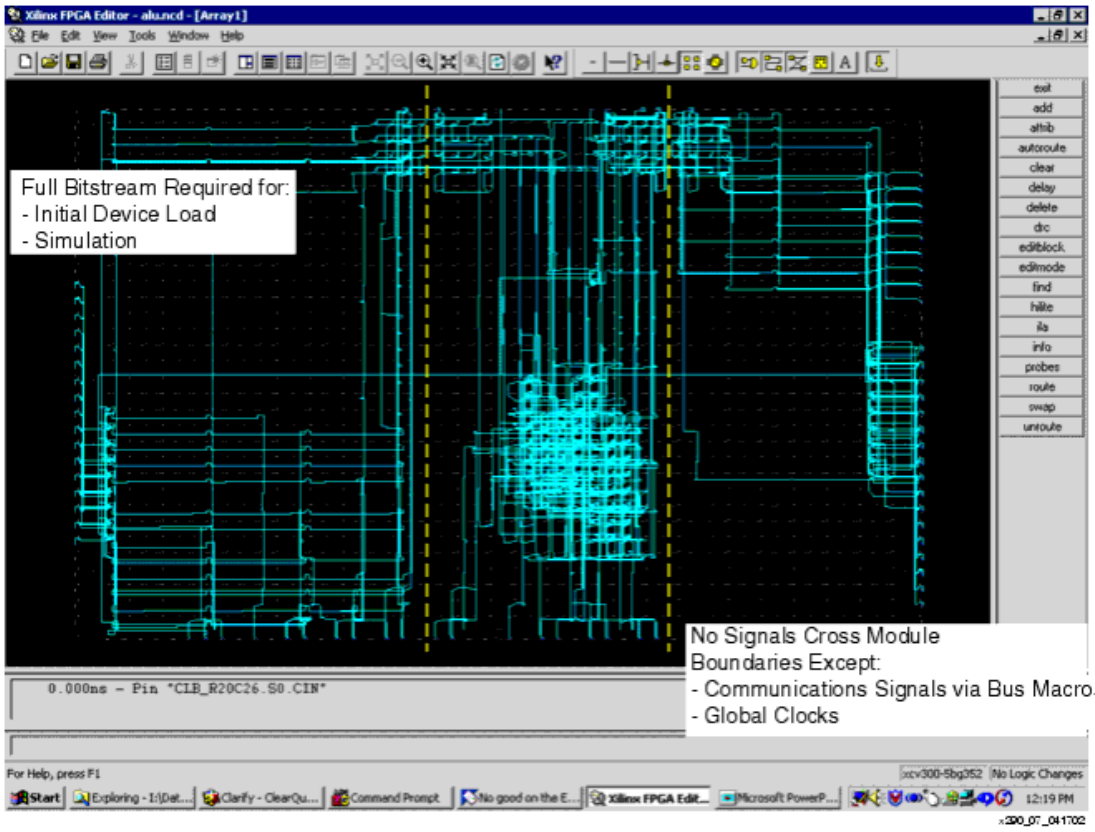
Obrázek 8.5-1 Správně navržený modul pro částečnou rekonfiguraci

8.5.3 Final assembly

V tuto chvíli máme připravené moduly pro implementaci do top-level designu a je potřeba je propojit mezi sebou a vytvořit vazby na globální logiku a časování.

Doporučuje se dělat tuto fázi pro každý modul zvlášť včetně simulace, aby se zamezilo křížení trasování modulů, případně špatnému nastavení vazeb mezi nimi. Pro správný chod částečné rekonfigurace je potřeba nahrát do FPGA počáteční bitstream.

Implementace finálního návrhu je úplně stejná jako pro jednotlivé moduly zvlášť(viz. předchozí kapitola), proto ji nebudu znovu uvádět.



Obrázek 8.5-2 Finální podoba modulárního systému v FPGA Editoru

Ve výše uvedeném obrázku lze vidět kompletní návrh FPGA s jedním modulem, který je zvýrazněn žlutým orámováním. V horní části jsou umístěna bus makra a kolem modulu je globální logika a IOB. Důležité je si povšimnout, že nedochází ke křížení tras globální logiky a modulu.

8.6 TVORBA BITSTREAMU PRO ČÁSTEČNOU REKONFIGURACI

Pro aktivní částečnou rekonfiguraci je nutný příkaz -g ActiveReconfig:Yes, který vyjadřuje připravenost zařízení pro nahrání nového bitstreamu za plné funkčnosti zařízení. Pokud toto není zajištěno, bitstream obsahuje své vlastní ukončení, aby nedošlo k poškození návrhu. Dodatečně je vyžadováno -g Persist:Yes, pokud je prováděna rekonfigurace přes SelectMAP mód. Tento příkaz umožní zachování mapování SelectMAP pinů po skončení konfigurace zařízení, aby bylo možné použít SelectMAP interface pro pozdější možnost rekonfigurace

Rozdílový bitstream může být vytvořen s jakýmkoliv dalším příkazem specifikujícím chování BitGenu, který neobsahuje šifrování. Stejně tak nemůže být zařízení částečně rekonfigurováno pomocí šifrovaného bitstreamu.

Pro každou konfiguraci modulu je potřeba vygenerovat vlastní bitstream, přičemž je nutné mít k dispozici i bitstream původní. Pokud je tedy například modul A rekonfigurovatelný, vytvoření jednoho z konfiguračních bitstreamů vypadá následujícím způsobem

bitgen -g ActiveReconfig:Yes -g Persist:Yes –d designA1.ncd designA1.bit

Tímto příkazem byl vytvořen konfigurační soubor designA1.bit, který konfiguruje v zařízení pouze rámec, definovaný oblastí AREA GROUP pro A1 uvnitř modulu A. JE NUTNÉ, aby před nahráním tohoto souboru bylo již zařízení naprogramováno pomocí původního bitstreamu.

9. MOŽNOSTI REKONFIGURACE JEDNOTLIVÝCH ČIPŮ

Základním předpokladem pro možnost částečné rekonfigurace zařízení je tedy, jak již bylo řečeno, mít možnost FPGA znovu rekonfigurovat. Tato část práce se zabývá bližším pohledem na jednotlivá technická řešení a specifika čipů. Pro základní přehled o nabízených čipech je dobré vědět, jak robustní vlastně jednotlivá zařízení jsou (tedy počet CLB a I/O, velikost paměti, atd.…). V následující tabulce jsou uvedeny základní vlastnosti jednotlivých čipů a dále se již budeme zabývat možnostmi rekonfigurace jednotlivých čipů.

Vlastnost	Virtex-5	Virtex-4	Virtex-2	Spartan-3A
Počet logických buněk	až 330000	až 200000	až 99000	až 53000
Počet I/O	až 1200	až 960	až 1164	až 519
Počet podporovaných I/O standardů	40	20	20	20
Podpora DCM	ano	ano	ano	ano
Podpora PLL	ano	ne	ne	ne
BRAM	18Mb	11Mb	7,9Mb	1,8Mb
Násobičky pro DSP	25x18 MAC	18x18 MAC	18x18 MAC	18x18 MAC
Gigabitové sériové rozhranní	ano	ano	ano	ne
Vestavěný PowerPC procesor	ano	ano	ano	ne

Tabulka 9-1 Srovnání základních parametrů FPGA

9.1 SPARTAN 3

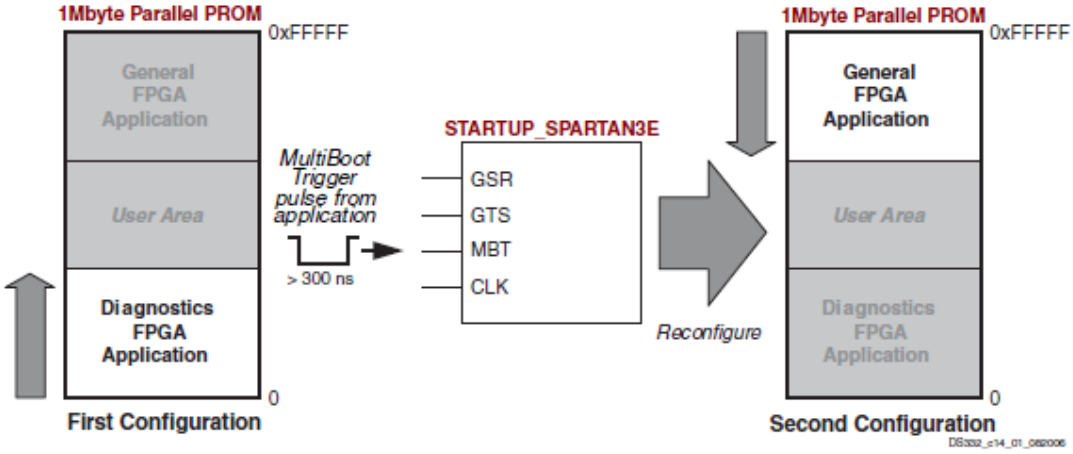
Nejstarší generace zařízení, podporujících rekonfiguraci, zahrnuje čipy Spartan 3A, Spartan 3AN, Spartan 3A DSP a Spartan 3E. Xilinx zde představuje novou schopnost svých zařízení nazývanou MultiBoot. Tento rys umožňuje FPGA aplikacím nahrání dvou a více bitstreamů bez ztráty kontroly nad zařízením. Mezi MultiBootem jednotlivých typů čipů rodiny Spartan 3 jsou patrné zásadní rozdíly. Zatímco základní typ Spartan 3 FPGA částečnou rekonfiguraci nepodporuje, s modelem Spartan 3E přišla první jednoduchá metoda částečné rekonfigurace. Pro

lepší přehled v jejich možnostech, jsou specifika jednotlivých typů Spartanů 3 uvedeny v tabulce.

	Spartan 3E	Spartan-3A/3AN/3A DSP
MultiBoot v BPI módu přes paralelní NOR flash paměť	ano	ano
MultiBoot v SPI módu přes sériovou SPI flash paměť	ne	ano
MultiBoot z interní flash paměti	ne	pouze Spartan 3A
MultiBoot pro více zařízení propojených do řetězu	ne	ano
Jak aplikace spouští MultiBoot	MultiBoot Trigger, STARTUP primitiva	sekvence příkazů pro ICAP, JTAG nebo SelectMAP
Maximální počet konfiguračních obrazů	2	limitováno pouze velikostí konfigurační paměti
umístění konfiguračního obrazu při inicializaci rekonfigurace	kontrolováno pinem M0 0 = nultá adresa PROM 1 = nejvyšší adresa PROM	vždy na nulté adrese
může aplikace FPGA určit počáteční adresu MultiBoot sekvence	ne, sekvence musí začínat vždy na nulté nebo nejvyšší adrese PROM	ano

Tabulka 9-2 MultiBoot pro rodinu Spartan 3

Jak je vidět, rekonfigurace Spartan 3 je závislá na funkci MultiBoot. Jak vlastně MultiBoot funguje, ukazuje následující jednoduchý příklad pro Spartan 3E. Poté, co se FPGA poprvé nakonfiguruje z jednoho konce PROM pomocí BPI módu, může spustit MultiBoot sekvenci z opačného konce paralelní paměti. V originálním nastavení je MultiBoot vypnut a aby mohl být použitý v aplikaci, je nutné definovat STARTUP primitivu. Pro spuštění sekvence slouží na vstupu MBT (MultiBoot Trigger) 300ns dlouhý Low pulz do primitivy. Po návratu MBT na hladinu High se FPGA automaticky překonfiguruje z opačného konce paralelní paměti.



Obrázek 9.1-1 příklad MultiBootu pro Spartan 3E

9.2 VIRTEX II A VIRTEX II PRO

Rekonfigurace Virtex II může být provedena pomocí Boundary-Scanu (JTAG) nebo SelectMAP. Pro spuštění rekonfigurační sekvence po SelectMAP je nutné povolit „*persist,*“ jak již bylo psáno u popisu jednotlivých rozhranní, proto se podíváme blíže na vlastní sekvenci pro aktivní rekonfiguraci.

Prvním krokem je synchronizace zařízení, stejně jako u klasické konfigurace při vypnutém zařízení. Nenásleduje ovšem zápis příkazu SHUTDOWN do CMD registru, ale pouze reset CRC. V tuto chvíli je potřeba předat FPGA informace o ID zařízení, velikosti rekonfigurované oblasti a vlastní informace o rekonfiguraci zakončené kontrolou (AutoCRC). Poté se opět resetuje CRC, nastaví se na počáteční hodnotu („0000DEFC“) a provede se desynchronizace zařízení. Vše je zakončeno dvěma NOOP slovy („0x20000000“) pro vyčištění bufferu.

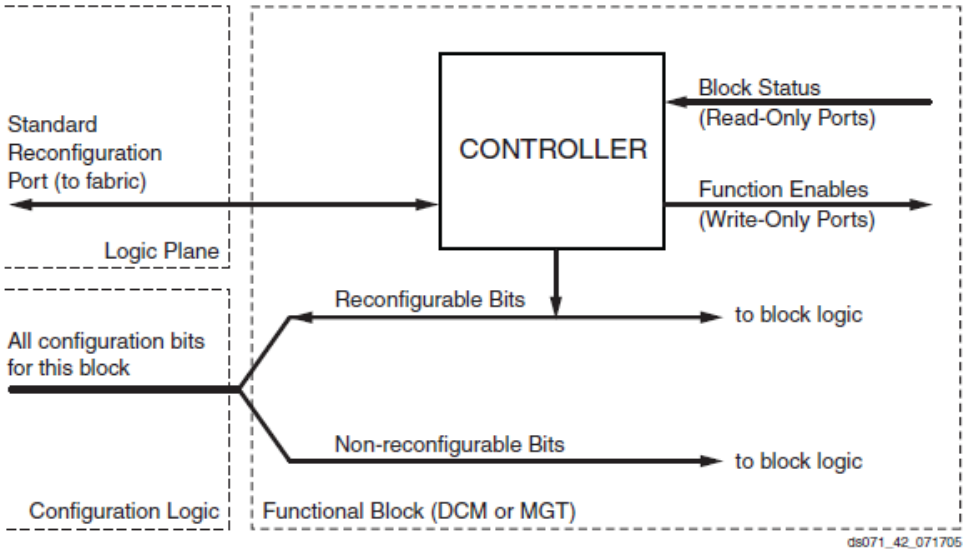
Programátor nemusí mít znalost, jak tyto příkazy provést, protože sekvenci ve správné formě za něj provede program iMPACT, který je součástí balíčku ISE (i jeho webové verze ISE Webpack). iMPACTU proto stačí dodat konfigurační bitstream.

9.3 VIRTEX IV

Virtex IV rozšiřuje možnosti Virtexu II o rekonfiguraci přes ICAP. Vzhledem k tomu, že signály pro ICAP jsou shodné se SelectMAP kromě oddělených datových sběrnic, rekonfigurace po ICAP se principielně neliší od SelectMAP.

9.3.1 DRP

V konfiguračním manuálu k Virtex IV lze najít rekonfigurační techniku nazvanou DRP (Dynamic Reconfiguration of Functional Blocks). Využívá se při ní dynamického rozhranní, které pracuje se všemi logickými bloky, které byly dříve označeny, že mohou být někdy rekonfigurovány. Takovýto blok obsahuje adresovatelnou paralelní read/write paměť, která je přístupná přímo z FPGA.



Obrázek 9.3-1 Dynamická rekonfigurace bloků pomocí DRP

9.3.2 DPSM

Další novinkou oproti Virtex II je rozšíření možností DCM o Direct Phase Shift Mode (DPSM). Uživateli je nyní umožněná přímá kontrola zpoždění fáze v DCM buď přes standardní Phase Shift rozhranní nebo přes nově prezentované DRP.

Za zmínku stojí i přítomnost dvou rozhranní ICAP. Jsou umístěna na horní a spodní straně Virtex IV a sdílí stejnou vnitřní logiku, což jim znemožňuje pracovat současně. V původním nastavení je jako primární ICAP nastaven ten na horní straně. Pokud je potřeba používat obě rozhranní, musí se vždy aktivovat jako první horní ICAP a poté teprve přepnout na spodní. Je dána i sekvence příkazů pro přepínání mezi jednotlivými ICAPy, která zahrnuje synchronizaci původního rozhranní, poté zápis jedničky do 30. bitu registrů MASK a CTL (zajišťují přepnutí mezi rozhranními: CTL=1 pro přepnutí na spodní ICAP, CTL=0 pro horní), příkaz DESYNCH do CMD registru a synchronizaci nového ICAPu.

9.4 VIRTEX V

Opět platí všechny uvedené možnosti jako pro Virtex IV, přičemž Virtex V k nim přidává zpětnou odezvu v podobě kontroly průběhu částečné rekonfigurace. Tyto utility jsou nazvány Fallback MultiBoot a Watchdog.

9.4.1 Fallback

Pokud během rekonfigurace nastane problém a je nutné použít Fallback rekonfiguraci, dochází k automatickému resetu celé konfigurační logiky a k opětovnému nastartování konfiguračního procesu vyčištěním konfigurační paměti. Tento restart může vyvolat chyba v IDCODE, CRC, případně služby Watchdog, která bude rozvedena podrobněji níže. Během Fallback rekonfigurace je změněna hodnota Revision Select pinu a pokud opětovná konfigurace selže podruhé, celý proces je zastaven.

9.4.2 Watchdog

Virtex V Watchdog se používá pro sledování průběhu rekonfigurace nebo logických operací FPGA. Používá přidělené interní hodiny o taktu 50MHz, které jsou předděleny 256, což dává periodu kolem 5120ns s maximálním trváním signálu

kolem 86s. Ke zpřístupnění této služby slouží zápis do registru TIMER, který může být implementován přímo v konfiguračním bitstreamu nebo předán přes kterýkoliv konfigurační port.

Jakmile je Watchdog uvolněn, začíná odpočítávat. Pokud narazí na nulu dřív, než FPGA po konfiguraci plně nastartuje, dochází k chybě a je nastartována Fallback rekonfigurace pomocí IPROG, během které je Watchdog vypnut. K jeho opětovnému spuštění je potřeba, aby proběhla v pořádku Fallback rekonfigurace.

10. SOFTWARE PRO PRÁCI S FPGA XILINX

Do této chvíle jsme se zabývali čistě teorií programovatelných hradlových polí a metodami jejich částečné rekonfigurace. Vzhledem ke komplexnosti současných návrhů byla vytvořena grafická rozhranní pro zjednodušení návrhu složitých modulárních systémů. Cílem této kapitoly je seznámit se se základními softwarovými prostředky, které byly použity během praktické části této práce.

Veškerý dostupný software je použitelný na operačních systémech Windows XP, Vista i na Linuxu. U Linuxu jsou primárně podporovány distribuce RedHat a SUSE, ale po úpravě instalačních balíčků je možná instalace na jakékoliv jádro. V současné době platí i podpora jak 32bit tak 64bitové architektury. Co se týká podpory jednotlivých čipů, nejnovější verze softwaru jsou určeny pro Spartan 3A a novější a Virtexy od verze II dále.

10.1 ISE FOUNDATION/WEBPACK

Veškeré nástroje nutné pro návrh se distribuují v balíčku programů ISE Foundation. Tato verze je placená, ale má i variantu, která je poskytována zdarma. Jde o ISE WebPACK, který oproti placené verzi neobsahuje optimalizační nástroj PlanAhead a neumožňuje modulární částečnou rekonfiguraci. ISE WebPACK je k dispozici ke stažení přímo na stránkach Xilinx Inc. a je dostatečným nástrojem k seznámení se základy diferenční částečné rekonfigurace.

10.2 PLANAHEAD

PlanAHEAD je optimalizační nástroj, jehož velkou výhodou je možnost rozdělit si složitý návrh mezi fázemi syntézy návrhu a place/route na menší bloky nebo moduly a optimalizovat návrh každého z nich zvlášť. Nabízí lepší přehled o pinech a jejich přidělování, přičemž umožňuje běh více implementací najednou. Tento postup výrazně zvyšuje výkonnost celého systému. Pro potřeby této diplomové

práce je ale hlavní, že v PlanAHEADu probíhá celý návrh modulárně rekonfigurovatelného systému, proto se v druhé části praktických testů neobejdeme bez jeho pomoci.

10.3 EARLY ACCESS

Early Access je oblast podpory zákazníkům, která není běžně viditelná ani dostupná. Potřebuje ji ovšem každý, kdo se seznamuje s částečnou rekonfigurací v její složitější, tedy modulární, podobě. Pro udělení přístupu nestačí automatická registrace na stránkách Xilinx Inc., ale je nutné o ni zažádat zvlášť. Udělením přístupu se otevírá uživateli nová podoblast, ve které nalezne kromě odkazů na potřebný software i řadu příkladů a demo videí s tématem modulární částečné rekonfigurace. V současné době podporuje Xilinx EA PR (Early Access Partial Reconfiguration) pro software verze 9.2 a starší. Součástí instalačních balíčků jsou ISE Foundation, PlanAHEAD, PR nadstavby pro ISE Foundation a potřebné záplaty a bus makra. Každá verze software má ještě svoje vlastní specifika, co se týká knihoven k čipům, a tyto lze v EA PR nalézt také.

Cesta k těmto datům není jednoduchá, proto je nutné použít vyhledávač na www.xilinx.com a přes klíčové slovo Early Access a případně přes měnící se fóra dohledat odkaz na registraci do EA PR. O udělení nebo zamítnutí žádosti o přístup je uživatel informován e-mailem, jehož součástí je odkaz do zabezpečené oblasti EA PR.

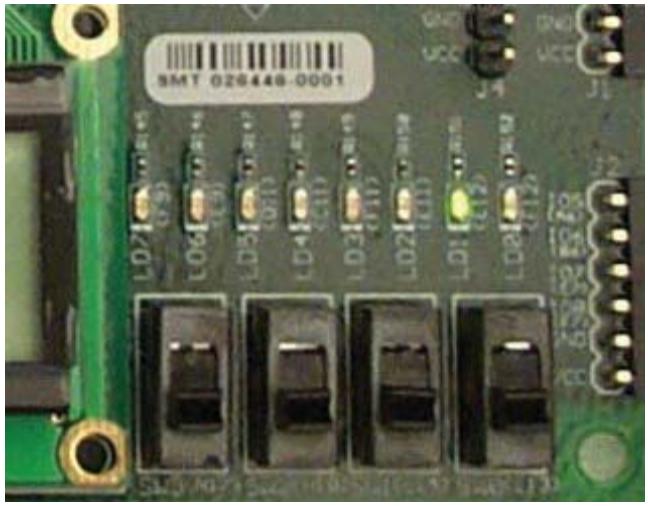
11. PRAKTICKÁ ČÁST

V této části bude ukázán postup částečné rekonfigurace v prostředí ISE 9.2i testovaný na desce Spartan 3E Starter Kit, na které byly zkoušeny základní postupy a principy diferenční i modulární metody částečné rekonfigurace.

11.1 DIFERENČNÍ ČÁSTEČNÁ REKONFIGURACE

11.1.1Testovací kód

Pro názornost příkladu jsem zvolil jednoduchý program, který ze dvou přímých vstupů ze switchů kombinuje výstup na led diodu. V původním kódu figuruje rozsvícení ledky LD7 při kombinaci N17 AND H18, který byl v FPGA Editoru překonfigurován na N17 OR H18.

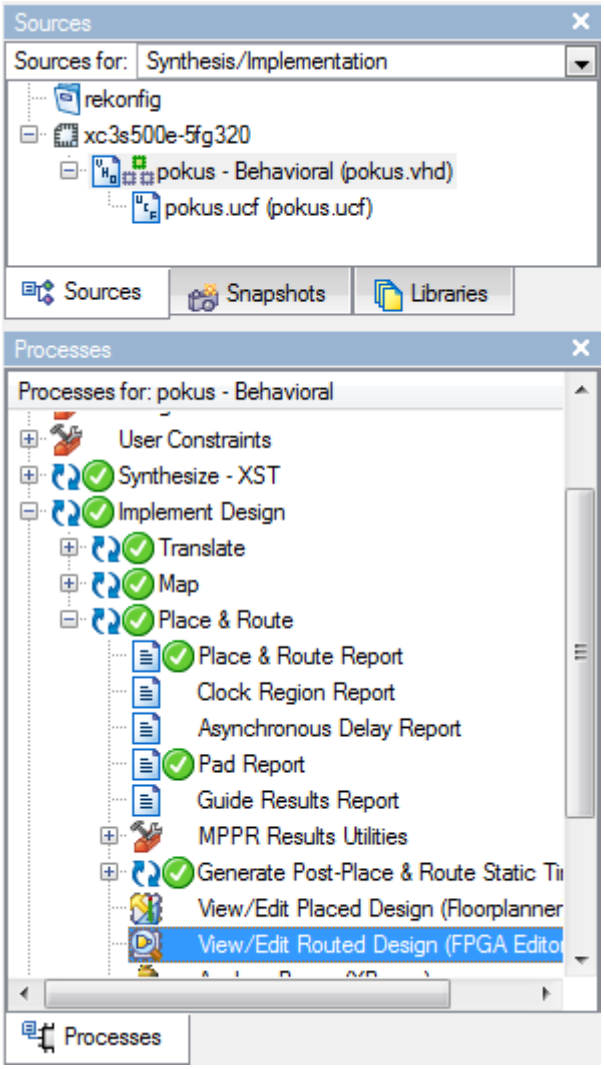


Obrázek 11.1-1 Výřez desky s použitými přepínači

11.1.2Postup při rekonfiguraci

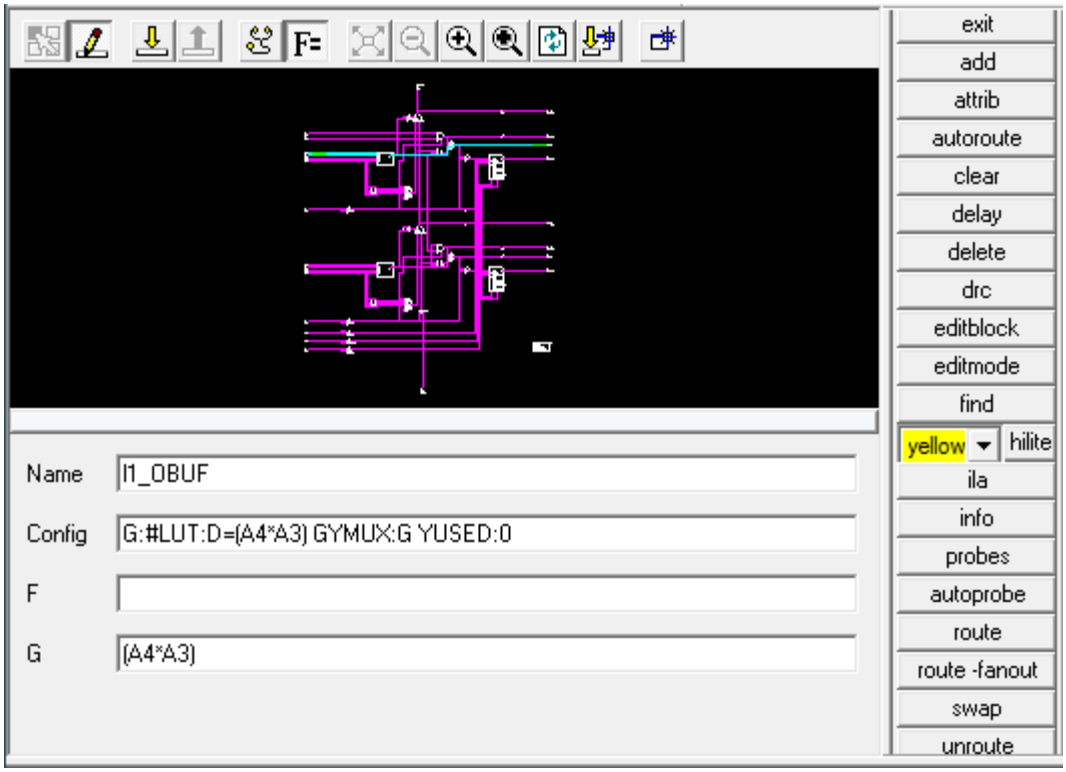
Při nahrání původního programu na desku byla konfigurace provedena standardně přes JTAG pomocí příkazu interface IMPACT. Pro vlastní rekonfiguraci už byl použit výše zmíněný FPGA Editor. Cesta k jeho otevření je ukázána na obrázku níže. Není nutné spouštět celý proces překladu znovu, protože nebyly

provedeny žádné změny, stačí proto prosté otevření záložky. Změny se dají zaznamenat jednoduše tak, že místo zelených ikonek jsou zobrazeny oranžové s otazníky. V tomto případě je nutné spustit celý proces překladu až k tomuto místu pomocí „rerun all.“



Obrázek 11.1-2 Cesta k FPGA Editoru

Pokud všechno proběhne korektně, došlo k otevření okna FPGA Editoru, ve kterém jsou ve výčtu neznámých vidět dva přímé vstupy, jeden výstup a jedna logika. Po nalezení logického bloku a jeho otevření v módu pro editaci můžeme měnit logickou funkci. V testovaném příkladu byla změněna z A4*A3 na A4+A3 (logický AND na OR).

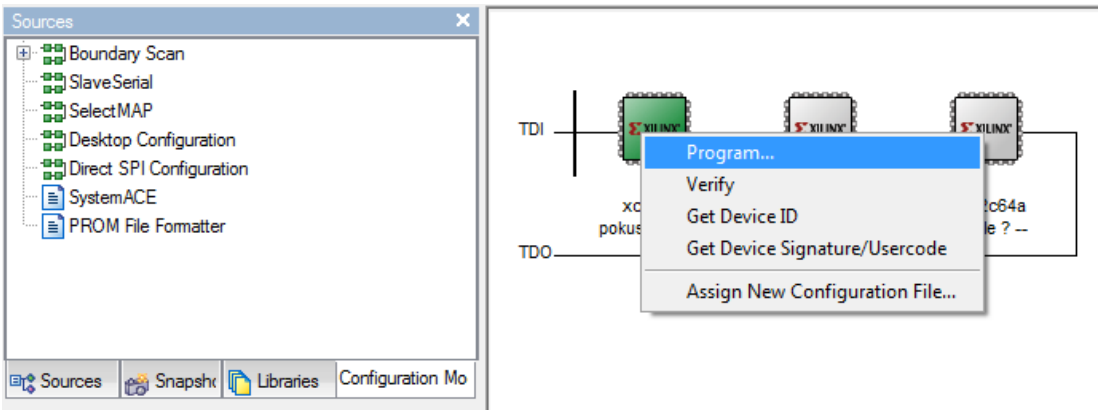


Obrázek 11.1-3 logická funkce pro vybranou LUT

Pozměněné NCD bylo uloženo do pokus2.ncd, čímž je připraveno vše nutné pro generování rozdílového bitstreamu. Ten byl vytvořen bez pomocí jakéhokoliv interface jednoduše z příkazové řádky:

bitgen -g ActiveReconfig:Yes -g Persist:Yes -r pokus.bit pokus2.ncd pokus_partial.bit

Nyní zbývá již jen spustit IMPACT a po vybrání správného pole nahrát rozdílový bitstream do desky. Při vybírání *.bit souboru pro nahrání je nutné tentokrát zvolit místo původního nový bitstream *pokus_partial.bit*.



Obrázek 11.1-4 Konfigurace desky pomocí bitstreamu

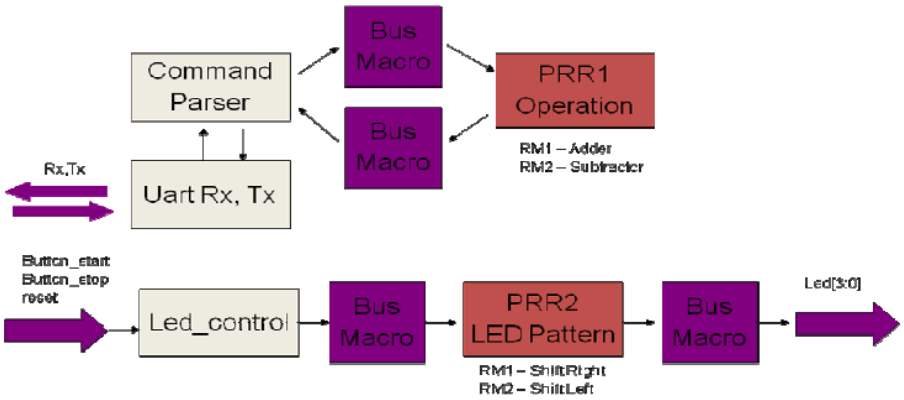
11.1.3Subjektivní zhodnocení

Orientace v FPGA Editoru je intuitivní a není složité najít všechny požadované funkce. Jedinou výhradu bych měl k vlastnímu programu ISE 9.2i, který se chová méně svižně, než jeho starší verze. Generování rozdílového bitstreamu prováděné z příkazového řádku je rychlá procedura, která zabírá čas úměrně provedeným změnám.

11.2 MODULÁRNÍ ČÁSTENÁ REKONFIGURACE

11.2.1Testovací kód

Vycházíme z předpokladu, že máme vytvořený kompletní modulární design, tedy globální logiku, statickou část a všechny potřebné verze modulů. Celý proces se tedy dále odehrává v prostředí PlanAHEAD a není již nutné používat ISE. Jako testovací projekt posloužil výukový zdroj přímo od Xilinx Inc., ke kterému se dá dostat po registraci a schválení přístupu Xilinxem k materiálům v sekci Early Access.



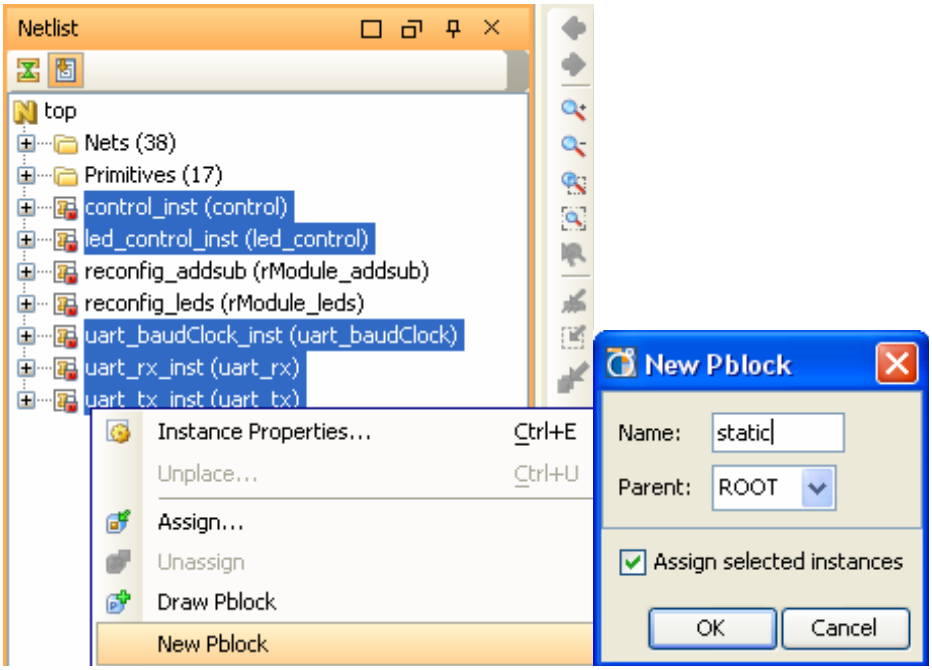
Obrázek 11.2-1 Celkový přehled příkladu MPR

Tento příklad obsahuje dvě rekonfigurovatelné oblasti, k nimž jsou ke každé k dispozici dva různé moduly. PRR1 obsahuje buď sčítačku nebo odčítačku a PRR2 dva různé výstupy na LED diody. Jak je vidět na obrázku výše, statické moduly (označeny šedě) spojují s PRR bus makra, která bude potřeba správně umístit a nakonfigurovat.

11.2.2Vytvoření modulárního systému schopného PR

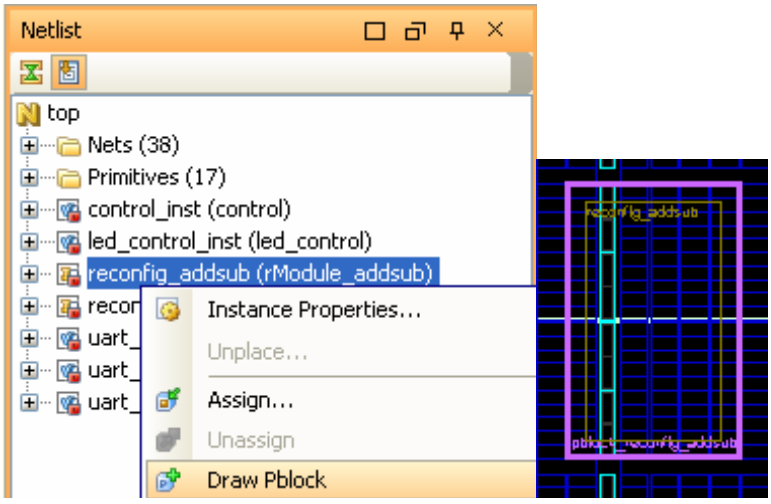
Po založení nového projektu přichází fáze importu jednotlivých netlistů. Akceptované formáty jsou EDIF a NGC. Následuje volba čipu a import UCF s namapovanými piny a navrženými omezeními jednotlivých modulů. V záložce „File“ se nastaví projekt jako otevřený pro částečnou rekonfiguraci pomocí „Set PR Project.“ Tím je projekt připraven a může se začít s konfigurací bloků.

Nejprve vyřešíme statické bloky. Označíme bloky, které v návrhu nebudou k dispozici pro rekonfiguraci a přes New PBlock vytvoříme skupinu statických modulů. Pojmenujeme ho a přidělíme do ROOTu.



Obrázek 11.2-2 Výběr statických bloků

Po statických modulech je na řadě vytvořit PRR. Jak již bylo řečeno, neumísťuje se modul přímo na místo na čipu, ale do předvytvořené oblasti pro rekonfigurovatelný modul (PRR). Vzhledem k tomu, že vidíme, jak je modul velký, vytvoříme obdélník tak, aby se do něj modul rozměrově vešel. Postup opakujeme pro všechny PRR.

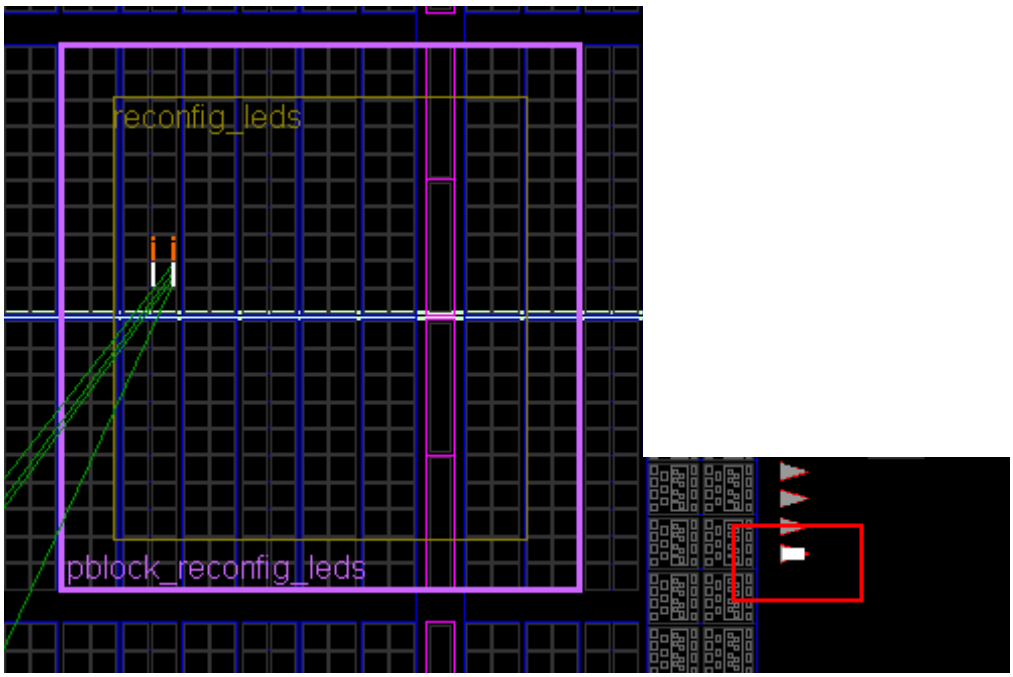


Obrázek 11.2-3 Vytvoření PRR

Aby bylo možné umístit všechny komponenty na čip, je nutné tuto možnost povolit „Create Site Constraint Mode“ níže ukázaným tlačítkem.

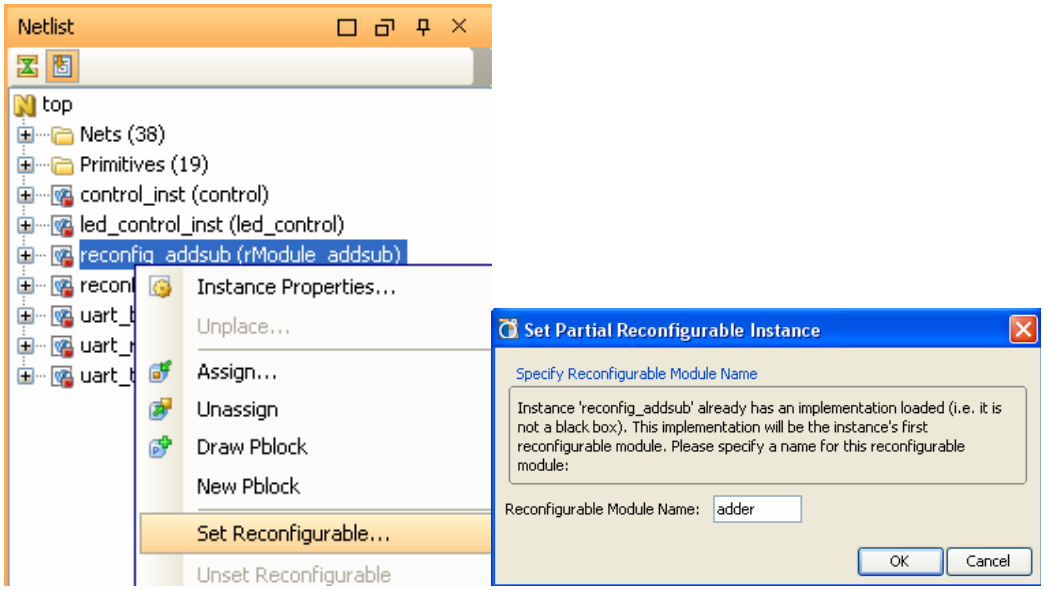
Obrázek 11.2-4 Umístění komponent

Nastává fáze rozmístění bus maker. V Netlist panelu se otevře adresář příslušící PRR bloku a pomocí drag’n’drop se makro umístí dovnitř obdélníku PRR.



Obrázek 11.2-5 Rozmístění bus maker

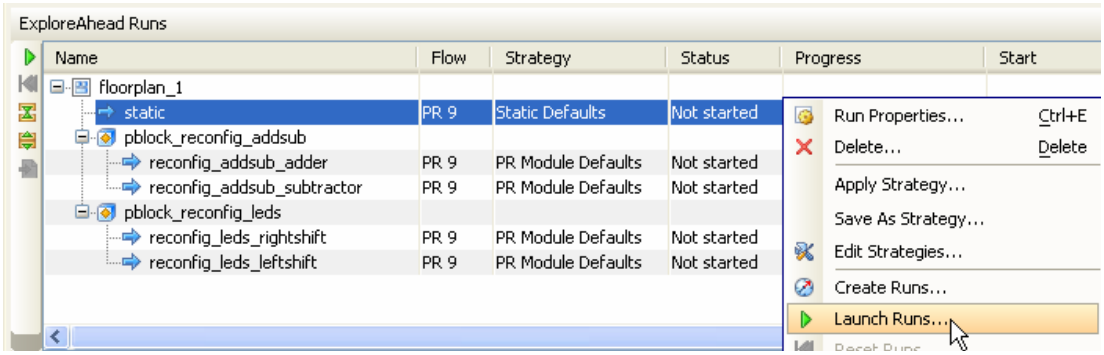
V tuto chvíli jsou vytvořeny PRR, přiřazeny jim odpovídající bus makra a je potřeba naplnit PRR jim odpovídajícími rekonfigurovatelnými (dynamickými) moduly. Postupně ke každé PRR přiřadíme její modul a pokud nejsou ještě importované všechny netlisty, přidají se stejně jako při zakládání projektu. Kontrolu správného přiřazení je možné sledovat v dolní části obrazovky pod záložkou „ExploreAhead Runs.“



Obrázek 11.2-6 Nastavení rekonfigurovatelných modulů

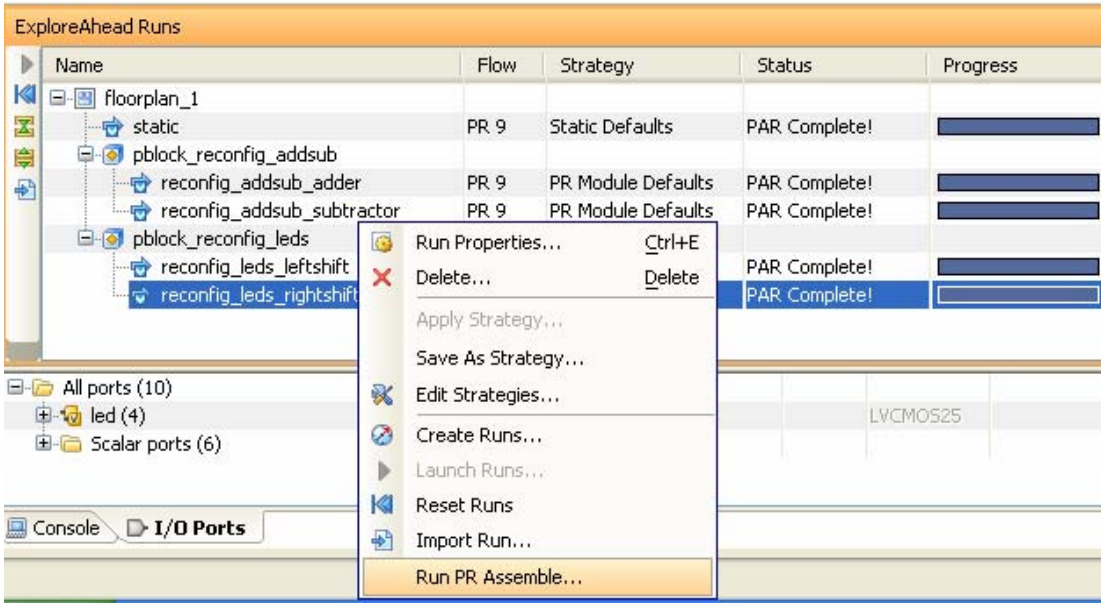
Před vlastním spuštěním konfigurace a implementací PRR do top-level designu je nutné provést tzv.: Design Rule Check (DRC), tedy kontrolu, zda je design správně navržen. Kontrolují se zde hodiny, I/O, paměti a přiřazení modulů. Tato kontrola se spouští v záložce „Tools“ volbou „Run DRC.“

Po DRC je floorplan připraven na implementaci statické části návrhu popsanou v teoretické části. Tato implementace se spouští v již zmíněném panelu „ExploreAhead Runs“ a je stejná jak pro statické, tak dynamické moduly.



Obrázek 11.2-7 Implementace modulů

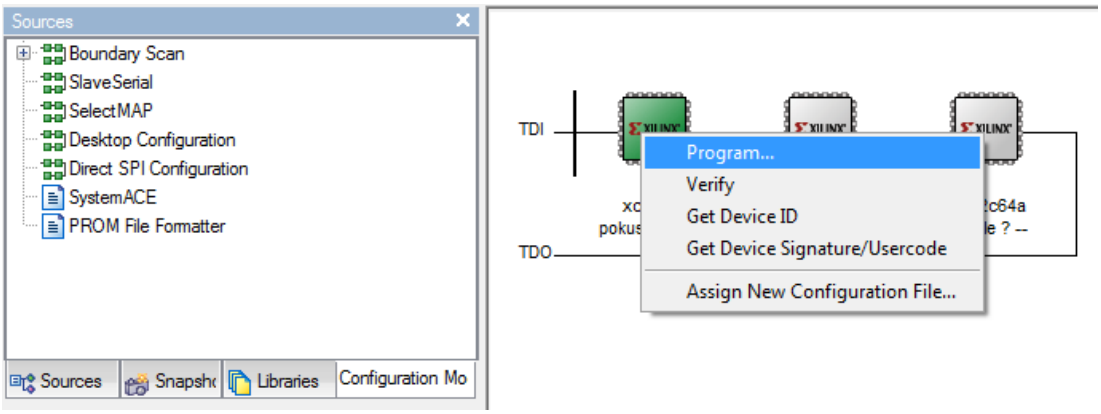
Po hotové implementaci je návrh připravený pro vygenerování konfiguračních bitstreamů. Ke generování slouží v PlanAHEADu opět záložka „ExploreAhead Runs.“ Vyskočí okno s výběrem modulu pro každou PRR a po potvrzení je vygenerován bitstream. Po jednoduché změně modulu se vygeneruje rozdílový bitstream, který slouží k pozdější možné částečné rekonfiguraci.



Obrázek 11.2-8 Generování konfiguračního bitstreamu

11.2.3Konfigurace zařízení

Stejně jako u rozdílové PR již zbývá jen spustit IMPACT a po vybrání správného pole nahrát konfigurační nebo rozdílový bitstream do desky. Jako první je nutné nahrát bitstream obsahující statickou část návrhu a poté nahrávat částečné bitstreamy obsahující jednotlivé moduly.

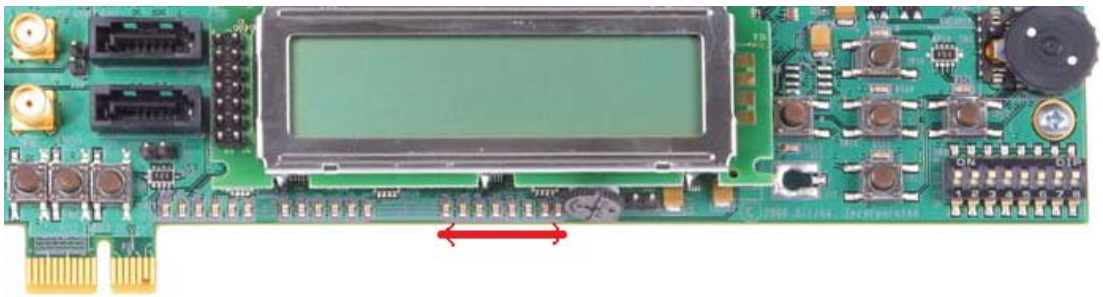


Obrázek 11.2-9 Konfigurace zařízení pomocí bitstreamu

11.2.4Ověření funkčnosti

Design poskytnutý Xilinx Inc. plní dvě nezávislé funkce. První PRR obsahuje informace o blikajících ledkách na pravé straně pod displejem a druhá zajišťuje po sériové lince podle nahraného modulu buď sčítačku nebo odčítačku.

Pro ovládání led diod slouží piny AK7 a AJ7 umístěné v pravém dolním rohu desky, kdy AK7 označený na desce jako SW_E startuje sekvenci blikání a AJ7 označený jako SW_W ji zastavuje. K resetu sekvence je určeno tlačítko CPU reset na pinu E9 označený jako SW7, který je umístěn nejvíce vpravo v trojici tlačítek vlevo od displeje. Podle nahraného modulu je sekvence blikání ledek pod displejem buď zleva nebo zprava, přičemž sekvence zahrnuje čtyři led diody na levé straně ledek označených na následujícím obrázku.



Obrázek 11.2-10 Výřez desky ML505 s použitými tlačítky a ledkami

Jak již bylo uvedeno, druhou PRR obsadily sčítačka a odčítačka. Pro komunikaci s těmito moduly je zapotřebí program TeraTerm přiložený na CD. Testování je možné i pomocí HyperTerminalu integrovaným přímo ve WindowsXP, ale vzhledem k tomu, že HyperTerminal není již součástí Windows Vista, bylo potřeba použít program třetí strany.

Pro testování jsou k dispozici dva registry „*a*“ a „*b,*“ do kterých je možné vkládat číselné hodnoty. Operátor „=“ poté vyvolá operaci příslušnou nahranému modulu, tedy sečtení nebo odečtení $a \pm b$. K dispozici je ještě operátor „*?,*“ který vrací symbol operace příslušící nahranému modulu.

11.2.5Subjektivní zhodnocení

Oproti diferenční metodě je modulární rekonfigurace podstatně náročnější proces, který má velkou řadu specifik, která se v teoretických popisech nedozvíte. Nejdůležitějším poznatkem z testování je absolutní funkčnost na Windows Vista, což jsem i přes proklamovanou kompatibilitu nepředpokládal. Testování proběhlo na Windows Vista Business Edition i na Windows XP a na obou nevykázalo žádné chyby.

Pro vlastní práci je nutné vlastnit veškerý software ve verzi 9.2 nebo starší, protože Xilinx neuvolnil do této doby Early Access pro 10.1. Přestože se při vlastní rekonfiguraci nevyužívá prostředí ISE ale PlanAHEAD a IMPACT, je požadována nainstalovaná plná verze ISE Foundation (WebPACK nestačí) doplněná o patch 9.2.4 a všechny toolboxy, které Xilinx poskytuje v rámci Early Access. Jedná se především o bus makra, ale velice užitečné jsou i příklady a demo videa, jak při návrhu postupovat. V této práci byl vyzkoušen příklad *pa_lab* dostupný v EA PR, který je přiložen k elektronické části práce.

12. ZHODNOCENÍ PRAKTICKÉ ČÁSTI

Během své práce jsem musel řešit problémy spojené pouze s operačním systémem, případně verzemi softwaru. Vlastní rekonfigurace probíhají bez problémů, pokud jsou dodrženy postupy uvedené v toolboxech Xilinx Inc. Co se týká rozdílové metody rekonfigurace, ta je dostupná všem bez omezení. Pro zpřístupnění podpory pro modulární metodu je nutná dodatečná registrace a žádost o povolení přístupu k Early Access Partial Reconfiguration (EA PR). Xilinx Inc. si vyhrazuje právo na zamítnutí této žádosti. Předpokládám, že je tomu tak z důvodu zamezení zahlcení technické podpory Xilinx dotazy směrovanými právě na částečnou rekonfiguraci, protože tato technologie je zatím přístupná pouze akademickým pracovištím a není masově nasazena v průmyslu.

Rozdílovou částečnou rekonfiguraci se podaří uvést do provozu na jakékoliv verzi ISE, ať již jde o volně šiřitelnou verzi WebPACK nebo placenou Foundation. Problémy vznikají při modulární částečné rekonfiguraci. Zde je nutné použít verze ISE Foundation a PlanAHEAD 9.2 nebo starší, protože Xilinx nedodává podporu pro modulární částečnou rekonfiguraci pro nejnovější software 10.1. Je otázkou, jestli pro 10.1 bude podpora vůbec vytvořena, protože tato verze software je již dlouho k dispozici a během roku by měla být uvolněna verze 11. Již současná verze EA PR ovšem plně podporuje Windows Vista jak v 32-bit (otestována) verzi, tak 64-bit (neotestována) a nabízí v současné době dostatek informací k pochopení částečné rekonfigurace.

13. ZÁVĚR

Cílem této práce bylo prozkoumat možnosti částečné rekonfigurace FPGA firmy Xilinx. Bylo zjištěno, že existují dva možné přístupy: rozdílová a modulární částečná rekonfigurace. Částečná rekonfigurace není složitá technika na pochopení základů. Celý postup lze najít na diskusním fóru firmy Xilinx, případně v tutoriálech k jejich toolboxům. Výčet zařízení, která tuto technologii podporují, zahrnuje Spartany rodiny 3 a Virtexy 2, 4 a 5. Všechny tyto procesory jsou softwarově podporovány v používaných programech ISE 9.2i a PlanAHEAD 9.2.

Největší výhodou této technologie je možnost změny části FPGA za chodu zařízení, protože jsou systémy, které se nedají jednoduše odstavit kvůli provedení drobných úprav. Částečná rekonfigurace navíc zabere při nahrávání do pole mnohem méně času než nahrání bitstreamu s kompletním kódem, což by se navíc provádělo obtížně za chodu systému. Co se týká velikosti rozdílového bitstreamu oproti původnímu nebo změněnému, je ve většině případů mnohem menší, protože obsahuje pouze informace o místu v poli, které je rekonfigurováno, a vlastní změnu logiky (standardu, obsahu paměti, modulu). Možnou nevýhodou je nutnost precizní znalosti použitých proměnných, což je ale vyváženo možností výběru měněné veličiny v LUT a ne jejím vyhledávání v HDL kódu nebo schématu.

Konfigurace zařízení (nejen částečná) se dá provést dvěma způsoby: z interního nebo externího zdroje dat. Interním zdrojem je míněna situace, kdy se FPGA rekonfiguruje samo od sebe na základě vnitřní logiky. Pro tento případ slouží konfigurační rozhranní ICAP. Konfigurace z externího zdroje vyžaduje zásah uživatele a provádí se pomocí ostatních rozhranní (sériová a paralelní rozhranní, SelectMAP, JTAG).

Závěrem bych se zamyslel nad možnostmi rozšíření této práce. Nabízí se možnost zapracování nově uvedených čipů Virtex 6 a Spartan 6, které bude možné v momentě, kdy budou k dispozici dostatečné podklady a hlavně podpora ze strany

Xilinx ve formě EA PR. Ta samá situace nastane v momentě uvedení software verze 11. Na základě tohoto předpokladu navrhuji studii uplatnění částečné rekonfigurace v průmyslu a zamyšlení nad možnostmi masového nasazení a využitelnosti postupů probraných v této práci.

14. POUŽITÁ LITERATURA

[1] Kapitoly 3, 8: Virtex-5 FPGA Configuration User Guide, dostupné z http://www.xilinx.com/support/documentation/user_guides/ug191.pdf

[2] Kapitoly 3, 8: Virtex-4 FPGA Configuration User Guide, dostupné z http://www.xilinx.com/support/documentation/user_guides/ug071.pdf

[3] Kapitoly 3, 8: Virtex-II Platform FPGA User Guide, dostupné z http://www.xilinx.com/support/documentation/user_guides/ug002.pdf

[4] Kapitoly 3, 8: Spartan-3 Generation Configuration User Guide, dostupné z http://www.xilinx.com/support/documentation/user_guides/ug332.pdf

[5] Kapitoly 5, 6, 7: Metody částečných rekonfigurací, tutoriál Xilinx dostupný z http://toolbox.xilinx.com/docsan/xilinx7/books/data/docs/dev/dev0036_8.html

[6] Kapitola 4: Modulární navrhování, tutoriál Xilinx dostupný z http://toolbox.xilinx.com/docsan/xilinx7/books/data/docs/dev/dev0025_7.html

[7] Kapitola 2: Základy FPGA, dostupné z http://www.xilinx.com/company/gettingstarted/index.htm

[8] Kapitoly 10, 11: Early Access Partial Reconfiguration, technická podpora Xilinx dostupná z http://www.xilinx.com/support/prealounge/protected/index.htm

15. POUŽITÉ ZKRATKY A REGISTRY

- BOOTSTS
  - Boot History Status Register
- BPI
  - Byte-wide Peripheral Interface
  - Konfigurační rozhranní
- BRAM
  - Block Radom Access Memory
  - Bloková paměť procesorů Xilinx
- BUFGMUX
  - Spojení signálů Clock a Clock Enable
  - Použitelné pouze ve Virtex II a II Pro
- CLKIOB
  - Hodinový signál z externího zdroje
- DCM
  - Digital Clock Manager
  - Vlastnost zařízení Xilinx umožňující kontrolu nad možnostmi časování signálů
- DLL
  - Delay Locked Loop
  - Frekvenční ústředna pro DCM
- DRM
  - Digital Rights Management
  - Ochrana dat proti neoprávněnému užití
- DRP
  - Dynamic Reconfiguration Port
- FPGA
  - Field Programmable Gate Array
  - Polovodičové zařízení obsahující programovatelné logické bloky

- HDL
  - Hardware description language
  - Programovací jazyk
- ICAP
  - Internal Configuration Access Port
  - Rozhraní pro konfiguraci FPGA
- IPROG
  - Internal PROG
  - Interní registr pro vyvolání restartu za tepla
- I/O
  - Input/Output
  - Označení pro vstupně-výstupní bloky
- JTAG
  - Joint Test Action Group
  - Jiné označení pro Boundary Scan, respektive standard IEEE 1149.1
  - Rozhraní pro konfiguraci FPGA
- LUT
  - Look-Up Table
  - Tabulka s přehledem logických funkcí
- PLL
  - Phase Locked Loop
  - Frekvenční ústředna pro DCM
- SelectMAP
  - Základní konfigurační mód zařízení Xilinx
- UCF
  - User Constrains File
  - Soubor obsahující namapování pinů
- WBSTAR
  - Warm Boot Start Address Register
  - Registr pro teplý restart

16. SEZNAM PŘÍLOH

[1] Difference_based: adresář, bitstreamy pro rozdílovou částečnou rekonfiguraci

[2] Modular_based: adresář, bitstreamy pro modulární částečnou rekonfiguraci

[3] TeraTerm: freeware aplikace, zprostředkování komunikace po sériové lince