Technická Univerzita v Liberci

Fakulta mechatroniky, informatiky a mezioborových studií

Ústav informačních technologií a elektroniky

Aplikačně závislé testování FPGA obvodu

Application dependent FPGA testing

Autoreferát disertační práce

2011 **Ing. Martin Rozkovec**

Aplikačně závislé testování FPGA obvodu

Application dependent FPGA testing

Autoreferát disertační práce

Autor: Ing. Martin Rozkovec

Studijní program: P2612 Elektrotechnika a informatika

Studijní obor: 2612V045 – Technická kybernetika

Školící pracoviště: Ústav informačních technologií a elektroniky

Fakulta mechatroniky, informatiky a mezioborových studií

Technická Univerzita v Liberci

Studentská 2/1402, 461 17, Liberec

Školitel: Prof. Ing. Ondřej Novák, CSc.

Rozsah práce:

Počet stran: 21

Počet obrázků: 19

Počet tabulek: 9

Abstrakt

Tato práce se zabývá testováním obvodů FPGA. V textu jsou představeny základní pojmy testování obvodů ASIC a vybrané techniky testování obvodů FPGA. Jádrem textu je představení nové metody, která je určena k testování FPGA obvodu.

Představená metoda pracuje s modulárně rozděleným obvodem, který je implementován s pomocí částečné dynamické rekonfigurace. Změna okolí testovaného modulu – dynamické vytváření testerů a analyzátoru testu umožňuje otestovat celý obvod. Ověřovány jsou analyzátory na bázi MISR a skenovacího řetězce a testery na bázi LFSR, generátoru deterministického testu a dekompresoru zkomprimované sekvence.

Na různých experimentech jsou ověřeny dílčí části metody – schopnost dělení obvodu, transformace popisu obvodu pro ATPG, kvalita testovacích vektorů a schopnost detekce poruch SEU.

Abstract

This work is focused on the FPGA testing. The basics of the ASIC testing are presented in the text as well as some of the FPGA test methods. The core of the report is focused on presentation of the novel application dependent FPGA test method.

Presented method splits the tested circuit into several modules that are implemented using partial reconfiguration design flow. Partial reconfiguration allows changing the role of modules surrounding the tested module to testers and analyzers. All the functional modules can be tested using this approach. Analyzers based on a MISR and scan chains are evaluated as well as testers based on a LFSR, a deterministic test generator and a compressed sequence decompressor.

Circuit splitting, FPGA netlist to ASIC netlist transformation, quality of test patterns and ability of SEU detection are evaluated in various experiments within the text.

1 Úvod

Integrovaný obvod je vyroben z monokrystalického křemíkového plátku, na kterém jsou vytvořeny tranzistory, které jsou vzájemně propojeny pomocí několika metalických vrstev vodičů oddělených tenkými vrstvami izolantů. Motivy propojovacích vrstev jsou velké řádově desítky nanometrů a vrstvy jsou silné stovky atomů. Vlivem nedokonalostí při výrobě, nepříznivých provozních podmínek nebo degradace materiálů může v obvodu dojít k celé řadě poruch. Ty mohou mít na systém, jehož je obvod součástí nepříznivé následky. Ať už se jedná o řízení pracího cyklu nebo o zajištění bezpečnosti leteckého provozu, vždy jsou na obvody kladeny nemalé nároky na spolehlivost.

Spolehlivost obvodů lze zajistit nejen pomocí produkčních testů, ale i průběžným testováním obvodů v místě provozu. Cílem práce je vyvinout metodu testování obvodů FPGA, která umožní testování obvodu bez dodatečného přístrojového vybavení a za použití běžně dostupných nástrojů.

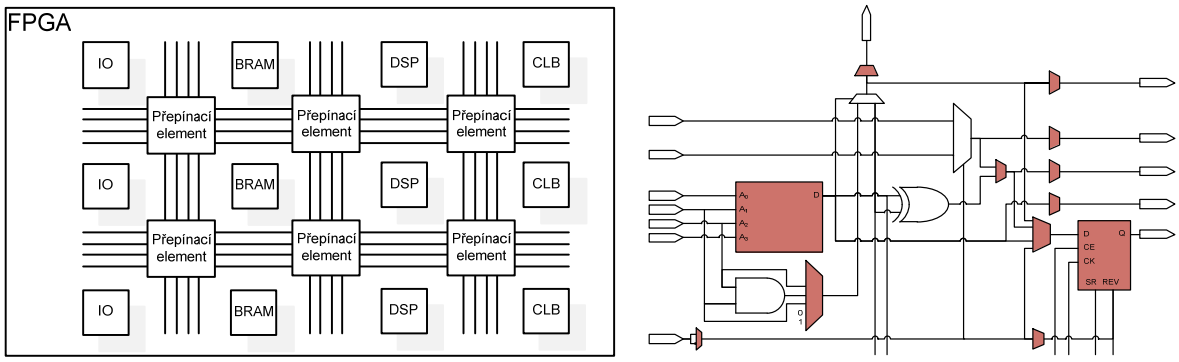
1.1 Poruchy číslicových obvodů

Trvalé poruchy vznikají většinou při výrobě obvodu nebo degradací materiálu integrovaného obvodu. Jelikož by pro účely testování bylo velmi náročné poruchy modelovat na fyzikální úrovni, používají se tzv. modely poruch [25]. Základními představiteli modelů trvalých poruch jsou trvalá 0/1, model zkratu, model zpoždění, model trvalé vodivosti tranzistoru nebo model přerušeného vodiče [30], [16].

Dočasné poruchy vznikají především vlivem provozních podmínek (například přehřátí obvodu, vystavení silnému elektromagnetickému poli, anebo radioaktivita prostředí) a po jejich odeznění mohou zaniknout. Při průchodu těžkého iontu, protonu nebo neutronu křemíkem dochází k přenosu energie částice do prostředí, který se projeví vznikem nosičů náboje (děr). Akumulovaný náboj vytvoří přechodný proudový zákmit, který bývá označován jako SET [20] (Single Event Transient). V případě, že částice narazí do paměťové buňky, může dojít k inverzi uložené hodnoty tvz. SEU (Singe Event Upset), jehož výskyty byly zaznamenány v kosmickém prostoru [18], ale i v letových hladinách běžných komerčních letů [20].

1.2 Obvody FPGA

FPGA obvod tvoří do matice uspořádaná soustava různých programovatelných bloků vzájemně spojených přepínatelnou propojovací maticí, viz obrázek 1.

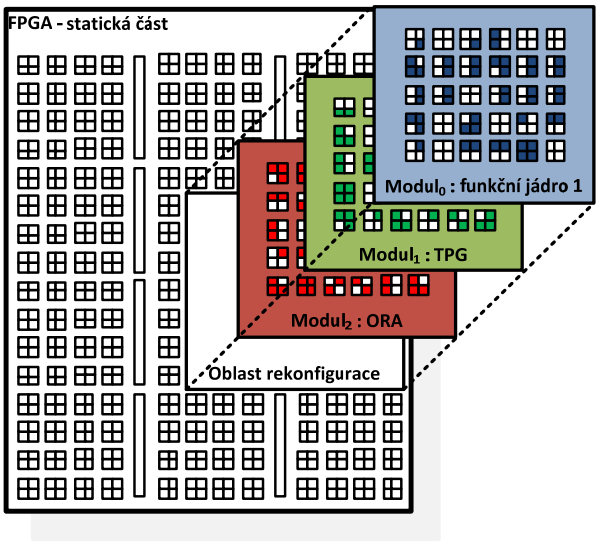


Obrázek 1: Struktura FPGA a části SLICE L

Největší procento programovatelných bloků je zastoupeno makrobloky CLB (Complex Logic Block). Ty se skládají ze dvou dvojic podbloků nazývaných SLICE-L (obrázek 1) a SLICE-M. Každý SLICE je tvořen dvojicí 16 bitových vyhledávacích tabulek LUT (Look Up

Table), výstupními multiplexory, dedikovanou aritmetickou logikou a konfigurovatelnými výstupními klopnými obvody typu D. V moderních obvodech FPGA tvoří další podstatné zdroje složitější obvodové prvky jako konfigurovatelné bloky paměti RAM/FIFO (BRAM), hardwarové MAC (Multiply-ACcumulate) bloky, rychlá sériová rozhraní a v některých obvodech dokonce procesory

Funkce obvodu implementovaného v FPGA je dána nastavením programovatelných bloků a jejich vzájemným propojením. V případě rodin Virtex-4 je nastavení obvodu ovládáno obsahem volatilní konfigurační paměti typu SRAM. V případě kompletní změny konfigurační paměti hovoříme o úplné rekonfiguraci. V případě, že nepřepíšeme celou konfigurační paměť, ale pouze její část, změníme pouze část funkce, která je příslušnou pamětí ovládána. Takovou konfiguraci nazýváme částečnou konfigurací. Je-li měněna část konfigurační paměti za běhu zbytku obvodu, mluvíme o částečné dynamické rekonfiguraci. Mění-li se základní elementy obvodu (vyhledávací tabulky, vzájemné propojení apod.) hovoříme o jemnozrnné rekonfiguraci. O hrubozrnnou rekonfiguraci (viz obrázek 2) se jedná v případě změny větších celků, například oblastí CLB i s propojením.



Obrázek 2: Princip hrubozrnné rekonfigurace

1.3 Poruchy FPGA založených na SRAM

Ačkoliv se FPGA řadí mezi obvody programovatelné, jedná se o obvody typu ASIC a jako takové je mohou postihovat všechny defekty ASIC obvodů. Z dlouhodobých měření výrobců [37], [22] lze usuzovat, že jsou FPGA obvody pro běžné aplikace dlouhodobě spolehlivé. V poslední dekádě se začaly FPGA používat v oblasti letectví, kosmonautiky, medicínské techniky, apod., kde jsou nároky na spolehlivost podstatně vyšší. FPGA zde získávají uplatnění hlavně kvůli nízké ceně vývoje, dobrému poměru výpočetního výkonu ke spotřebě a rekonfigurovatelnosti umožňující určitou míru samoopravitelnosti, která je v dlouhodobých „misích“ výhodná. Rekonfigurovatelnost FPGA založených na statické paměti je Achillovou patou jejich spolehlivosti. V leteckých a vesmírných aplikacích dochází v mnohem větší míře k poruchám typu SEU [20]. Výskyt a projevy těchto poruch jsou u FPGA obvodů založených na SRAM markantnější, nežli u ASIC obvodů:

- Největší část obvodu FPGA (plochu obvodu) zabírá konfigurační paměť. Pravděpodobnost vzniku SEU je tak vyšší než u obvodů, ve kterých převládá kombinační logika.
- SEU v konfigurační paměti není možné okamžitě detekovat. U ASIC obvodů lze paměť opatřit kontrolními a opravnými kódy, u hradlových polí se toto většinou nevyužívá.

- Porucha typu SEU se v FPGA může projevit jako trvalá porucha, protože SRAM paměť řídí konfiguraci obvodu, na rozdíl od ASIC, kde bývá použita především k uchování dat.
- Za určitých okolností může dojít vlivem SEU k trvalé poruše obvodu – např. zkratováním výstupů LUT. U ASIC hrozí „pouze“ ztráta dat.

1.4 Techniky testování FPGA

Metody testování lze rozdělit do několika kategorií [35]. Testy, které pro testování obvodu vyžadují zastavení činnosti obvodu, bývají označovány jako off-line testy. Testy označované jako on-line testy testují obvod za běhu tak, aby nebyla jeho funkce narušena. Off-line testy často využívají úplnou rekonfiguraci obvodu, zatímco online testy využívají částečnou rekonfiguraci. Testy nezávislé na struktuře implementovaného obvodu (aplikačně nezávislé [8], [9], [28]) testují všechny prvky FPGA bez ohledu na to, zdali jsou využity implementovaným obvodem, zatímco testy závislé na struktuře implementovaném obvodu (aplikačně závislé[12], [24], [32]) testují pouze ty zdroje FPGA, které implementovaný obvod využívá.

Aplikačně nezávislé metody vyžadují detailní znalosti struktury obvodu, která nebývá u nových generací FPGA k dispozici. Strukturní závislost omezuje přenositelnost metod mezi různými architekturami hradlových polí. Značná časová a paměťová náročnost je způsobená nutností vícenásobných rekonfigurací. Se zvyšující se heterogenitou obvodů nároky stoupají a test se prodlužuje a zabírá více místa v paměti. Testování všech zdrojů obvodu, bez ohledu na jejich reálné využití je předimenzované. Některé typy poruch (např. SEU, zpoždění) není možné testovat.

Aplikačně závislé metody založené na vyčítání konfigurační paměti mají nízkou závislost na struktuře FPGA, jsou snadno implementovatelné a lehce přenositelné na jiná FPGA. Netestují strukturní poruchy, ani všechny poruchy SEU. Metody založené na BIST vyžadují přístup k testovanému obvodu, který na hradlových polích významně zvyšuje spotřebu zdrojů obvodu a zhoršuje časovací charakteristiky. Strukturní závislost BIST je úměrná modelu FPGA obvodu, časová a paměťová náročnost je závislá na způsobu implementace BIST.

2 Metoda aplikačně závislého testování FPGA obvodů

Představená metoda spadá do kategorie aplikačně závislých testů založených na BIST. Režii BIST metoda redukuje pomocí částečné dynamické rekonfigurace. Částečnou rekonfiguraci metoda využívá k dynamickému vytváření vnořeného testovacího vybavení, k zajištění přenosu testovacích vzorků a ke zvýšení kontrolovatelnosti a pozorovatelnosti testovaného obvodu. Strukturní závislost na hradlovém poli, které je metodám založeným BIST vlastní je možné z velké míry odbourat použitím vhodného modelu obvodu; i při jen částečné znalosti struktury lze vytvořit model obvodu dostatečně kvalitní pro potřeby testování. Založení metody na BIST by však mělo přinést nesporné výhody v podobě postupů, nástrojů a technik, které jsou dlouhodobě aplikovány a prověřeny na obvodech ASIC. Pro specifické prostředí obvodů FPGA je však bude třeba přizpůsobit.

Podstatnou výhodou metody je i fakt, že většina obdobných prací se zaměřuje buď na zvýšení zabezpečení obvodu (např. [29]), anebo na aplikačně nezávislé testování specifického zdroje hradlového pole (např. [28]). Nevyužitý prostor, který tak vzniká, může metoda inovativně zaplnit.

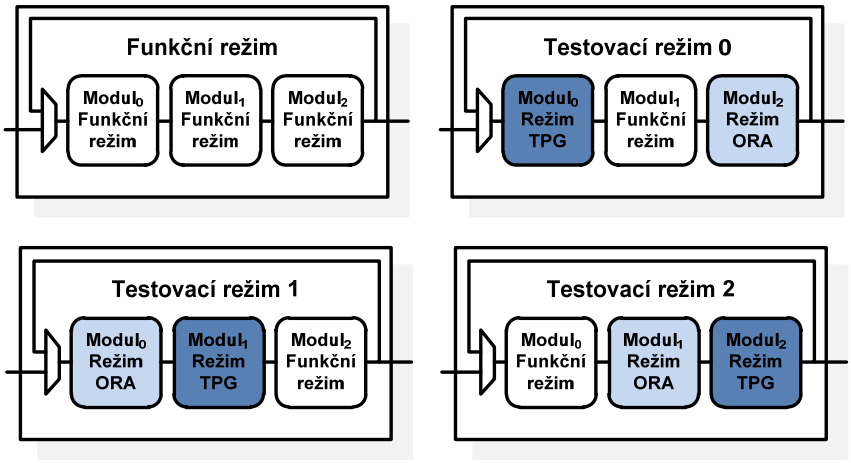
2.1 Přehled metody

Zjednodušený princip testování pomocí metody je následující: Návrh obvodu implementovaný v FPGA musí být rozdělen na nejméně tři části. Každá část je upravena a

implementována jako rekonfigurovatelný modul. Moduly jsou doplněny o obálku, která umožňuje minimalistické řízení testu a zároveň sjednocuje přístup ke každému jádru. Obálka zajistí, že je ke každému modulu možné jednoduše vytvořit další obvody – generátor testovacích vektorů (TPG) a analyzátor odezev (ORA).

Testování probíhá následovně: „pravý“ soused je překonfigurován na ORA, „levý“ soused na TPG (viz obrázek 3) a je zahájen autonomní test. Po ukončení testu jsou okolní moduly navráceny do funkčního režimu. Cyklickým posouváním generátoru a analyzátoru po rozděleném obvodu je možné celý obvod otestovat. Na hradlovém poli může být takovým způsobem implementováno více obvodů, ve kterých může testování probíhat paralelně.

Přístup k testovanému obvodu je řešen pomocí dynamické rekonfigurace, které zachovává paralelní vstupy a výstupy testovaného modulu. Mezi primárními vstupy a výstupy obou krajních sousedů je vytvořena zpětná vazba tak, aby se mohly vzájemně testovat. Každá z obálek je doplněna o sériové řízení testu, aby bylo možné testování zahájit a po ukončení testu vyčíst příznak nebo informaci o úspěšnosti testu. Testovací signály jsou spolu s řadičem testu cyklicky propojeny, podobně, jako tomu bývá například u rozhraní JTAG [17].



Obrázek 3: Přesouvání modulů v obvodu

Protože ke generování testovacích vektorů chceme využít nástroje běžné pro obvody ASIC, bude nutné popis obvodů s FPGA prvky přepsat do strukturního netlistu na bázi ASIC prvků. Deterministický test může být oproti pseudonáhodnému testu výhodnější co do počtu testovacích vektorů a tím pádem bude obvod testován kratší dobu.

Paměťovou režii, která využitím deterministických vektorů vzroste, je možné kompenzovat využitím komprese testovacích vektorů, se kterou máme dlouhodobé zkušenosti [19]. Testovací vektory lze uložit do některého z paměťových prvků FPGA ve formě inicializace konfiguračních dat. Využitím částečné dynamické rekonfigurace se tak sníží nároky testeru na paměť, ale i na zdroje FPGA, poněvadž se v FPGA bude trvale nacházet jen minimum logiky určené pro testování a pro režii rekonfigurace.

Rekonfigurace okolních modulů bude fungovat podobně jako metoda bitstream scrubing, s tím rozdílem, že namísto porovnávání aktuálního obsahu konfigurační paměti s referenčním se bude při vracení okolních modulů do funkčního režimu konfigurační paměť obnovovat a tím bude docházet k odstranění přechodných poruch.

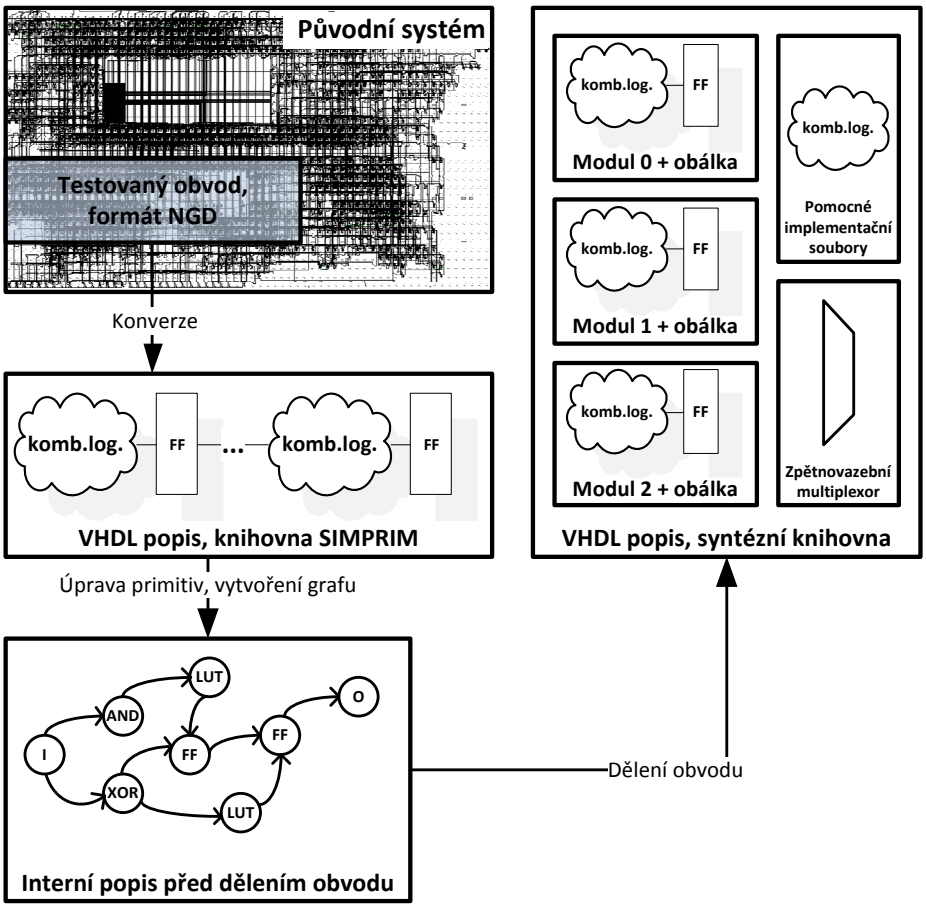
Tím, že v testovaném modulu (modul ve funkčním režimu) nezměníme konfigurační paměť, umožníme částečnou detekci SEU. Metoda odhalí pouze ty poruchy v konfigurační paměti, které vedou ke změně funkce implementovaného obvodu. Je zdokumentováno (např. v [13], [18]), že většina poruch konfigurační paměti na funkci obvodu vliv nemá.

Implementace metody sestává z několika dílčích kroků, které budou v následujícím textu představeny.

2.2 Dělení a přepis obvodu

Navržená metoda ke své funkčnosti vyžaduje rozdělení obvodu na pravidelné, podobně veliké části. Velké množství současných návrhů pro obvody FPGA využívá tzv. proudového zpracování (pipelining). Klopné obvody v pipeline vytváří přirozenou hranici, podél níž se vyplatí obvod dělit. Obvod bývá navržen tak, aby žádná z částí nebyla podstatně větší než ostatní. Tím by mělo být zajištěno i podobné kombinační zpoždění všech stupňů pipeline. Problém při dělení obvodu vyvstává v případě, když obvod, nebo jeho část, obsahuje zpětné vazby mezi registry. Zpětné vazby je možno virtuálně odstranit vložením skenovacího řetězce nebo sestavit sekvenční test.

Před samotným dělením je nutné implementovaný obvod převést z interního databázového formátu na (počítačem) čitelný strukturní netlist. Informace o struktuře obvodu lze získat z databázových souborů vytvářených v poslední fázi procesu implementace. Způsobů, jak přepis z interních databázových formátů získat ve zpracovatelné podobě je několik (viz [15], [36]). Představená metoda využívá program Netgen [39], který je původně určen k vytváření HDL popisů pro časovou simulaci. V obvodu, který přepisem vznikne, nejsou zastoupeny všechna primitiva, ale pouze ta, která jsou využita. Přepsaný soubor není dovoleno syntetizovat, pouze simulovat - za účelem syntézy je na souboru nutno provést několik úprav.



Obrázek 4: Přepis a dělení testovaného obvodu

Rozdělení obvodu provádí skript *split.* Skript načte ze zdrojového souboru všechna primitiva a jejich vzájemné propoje a v paměti vytvoří orientovaný graf obvod reprezentující. S původním obvodem provádí skript následující operace:

- Odstraňuje všechny ryze simulační konstrukty.
- Odstraňuje izolované prvky.
- Odstraňuje některé zdroje trvalých logických nul a jedniček.

- Odstraňuje šíření hodinového signálu a nahrazuje jej vlastním šířením.
- Nahrazuje simulační primitiva primitivy syntézními nebo vytváří funkčně shodné prvky.

Jako příklad nahrazení uveďme nahrazení nedělitelných posuvných registrů implementovaných pomocí vyhledávací tabulky na běžný posuvný registr tvořený z klopných obvodů.

Po modifikaci obvodových prvků se pomocí algoritmů prohledávání do hloubky a do šířky skript v grafu snaží nalézt všechny cesty mezi primárními výstupy a klopnými obvody, které oddělují jednotlivé stupně pipeline. Po nalezení všech cest a odstranění redundantních prvků, které se na nich nacházejí, jsou cesty a prvky, které na nich leží zapsány do nového souboru a odstraněny z původního obvodu. Nově vytvořený modul je doplněn o unifikované signály řízení testu (viz 2.6.1). Registry na hranici obvodu jsou interně přetypovány na primární výstupy. Algoritmus prohledávání se opakuje tak dlouho, dokud nenarazí na primární vstupy. Při každé iteraci je testována konzistence obvodu, především pak výskyt cyklů.

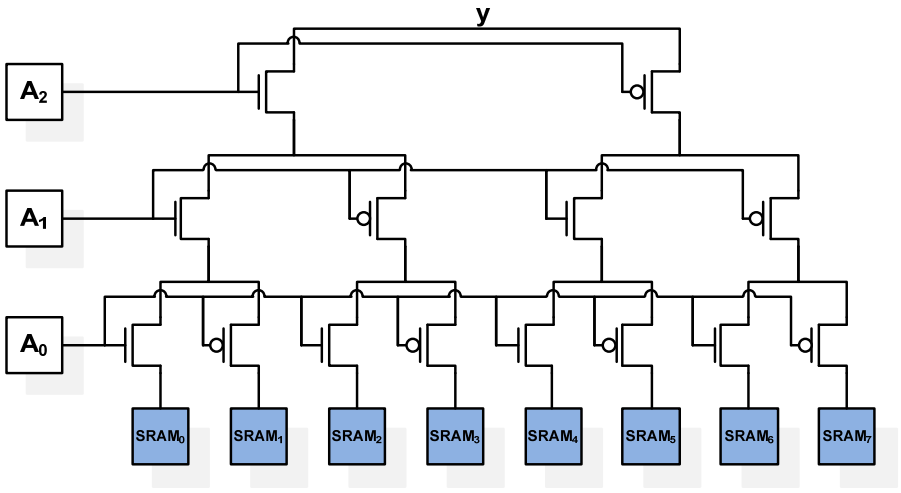
Poslední a první stupeň pipeline je nutno zpracovat odlišně, než zbytek obvodu. Výstupy posledního a vstupy prvního stupně jsou vzájemně porovnány. Je-li zjištěna neshoda, je obálka výstupního modulu doplněna tak, aby byly moduly vzájemně propojitelné. Prakticky to znamená, že skript doplní dodatečné primární výstupy posledního modulu, právě když je jejich počet nižší než počet primárních vstupů prvního modulu. Okrajové stupně obvodu jsou doplněny o multiplexor, který umožňuje cyklické testování všech modulů rozděleného obvodu. Modifikaci obálek a vytváření zpětnovazebního multiplexoru zajišťuje samostatný skript *make_feedback*, který navazuje na skript předchozí. Graficky je dělení a přepis obvodu znázorněno na obrázku 4.

2.3 Přepis pro ASIC ATPG

Po syntéze modulů a jejich implementaci je obvod konvertován do strukturního popisu způsobem uvedeným v předchozí kapitole. Strukturní popis je dále nutné přepsat do formátu čitelného některým z ATPG. V práci použité generátory používají formáty složené buď ze základních hradel (CIR[14], BENCH[21]) nebo z knihovních, strukturně popsaných prvků (Verilog), které jsou opět složeny ze základních hradel. U obou formátů je tak nutno všechna FPGA primitiva popsat pomocí hradel.

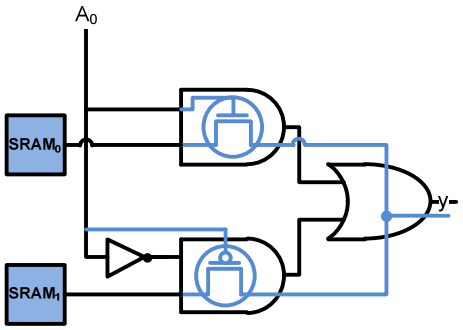
Jelikož není kvůli ochraně duševního vlastnictví možné zjistit přesný popis primitiv, jsou jednotlivá primitiva přepisována podle informací, které je možné nalézt v oficiálních dokumentech výrobce [38] nebo v nejrůznějších publikacích (např. [20], [28]). Náznaky informací o struktuře jsou k nalezení i ve zdrojových souborech knihoven UNISIM a SIMPRIM dodávaných společně s návrhovým prostředím. Jelikož jsou tyto knihovny určeny k simulaci, nemůže jim být při sestavování modelů přikládán velký význam.

FPGA primitiva lze rozdělit na několik skupin. V první skupině se nacházejí ta primitiva, jejichž přepis je zcela zřejmý a jejichž struktura naprosto odpovídá vytvořenému modelu. Prakticky se jedná pouze o hradla, která slouží k implementaci aritmetických funkcí (AND, XOR). Druhá skupina zahrnuje prvky, jejichž strukturu lze s pomocí výše uvedených zdrojů odhadnout, ale jejich model je pouze přibližný. Do této skupiny lze zařadit struktury budičů, multiplexorů (jak infrastrukturních, tak funkčních), klopných obvodů a vyhledávací tabulky. Třetí skupina obsahuje ty prvky, jejichž vnitřní zapojení je prakticky nezjistitelné nebo jsou tak komplexní, že je nereálné pro ně vygenerovat deterministický strukturní test. Prvky z třetí skupiny (DSP bloky, BRAM, přepínače propojovací matice apod.) přepisovány nejsou.



Obrázek 5: Strukturní model třívstupého LUT v generátorickém režimu.

Pravidla přepisu první skupiny jsou triviální. Instance prvku se nahradí příslušnou konstrukcí v syntaxi daného formátu. U prvků z druhé množiny je načtena konfigurace FPGA prvku, kterou je instance jeho modelu inicializována. Příkladem budiž přepis vyhledávací tabulky (obrázek 5). Jednotlivé transistory jsou v modelu nahrazeny hradly AND a invertory, spojení vodičů je realizováno jako hradlo OR (viz obrázek 6). LUT je tak modelován jako 16vstupý multiplexor se čtyřmi adresními vstupy. Hodnoty uložené v konfigurační paměti jsou nahrazeny konstantními signály log. 0 a log. 1. Dle podobného klíče jsou modelovány i další prvky z druhé skupiny a k nim příslušející část konfigurační paměti.



Obrázek 6: Model tranzistorového multiplexoru.

Transformované prvky jsou propojeny stejným způsobem jako originální obvod. Do struktury obvodu jsou dále přidány části obvodových struktur sloužících pro oddělení rekonfigurovatelných modulů. Pro transformaci obvodu za účelem generování testu byl vytvořen skript *trans*.

Výsledky přepisu vybraných benchmarkových obvodů řady ISCAS85 a ISCAS89 jsou zaneseny v tabulkách 1 a 2. V prvním řádku obou tabulek je počet hradel původního obvodu, druhý řádek udává počet hradel po implementaci a přepisu obvodu. Nárůst počtu hradel po přepisu ASIC→FPGA→ASIC je v průměru 30 násobný: při vynechání extrémně malých obvodů c17 a s27 jsou přepsané obvody v průměru 21 krát větší. Pro další krok (generování testu) je významným parametrem předposlední řádek tabulky, který udává počet zdrojů trvalé log. 0 a log. 1 v obvodu

Tabulka 1: charakteristiky přepsaných obvodů ze sady ISCAS85

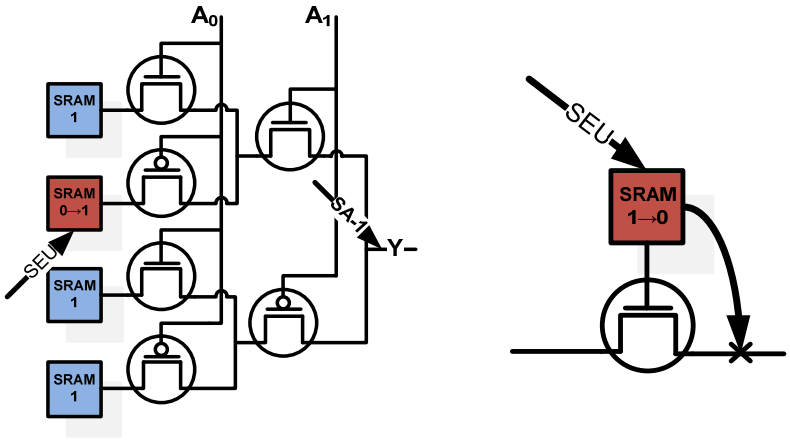
Obvod	c1355	c17	c1908	c3540	c432	c499	c880
ASIC hradel	546	6	880	1669	160	202	383
FPGA přepsaných hradel	7487	447	8020	20225	5870	7480	9589
Počet konstant v obvodu	3708	247	3589	7999	2557	3764	4555
FPGA SLICE	45	2	58	181	43	45	60

Tabulka 2: charakteristiky přepsaných obvodů ze sady ISCAS89

Obvod	s1494	s208	s27	s298	s344	s349	s386	s832	s1488
ASIC hradel	647	96	11	119	169	170	159	287	653
FPGA přepsaných hradel	14374	2586	792	3244	4418	4420	3867	7602	14606
Počet konstant v obvodu	5715	1304	424	1652	2310	2324	1736	3318	5723
FPGA SLICE	134	12	3	15	17	17	27	57	136

2.4 Generování testu

V práci použité ATPG (Atalanta [21], MILEF[14] a TetraMax[33]) podporují generování testů pro nejrůznější modely poruch, metoda využívá model trvalá 0/1. Ačkoliv není tento model pro moderní CMOS obvody nejvhodnější, jedná se o nejrozšířenější model poruch, který podchytí celou řadu poruch jiných modelů [23]. Použitím modelu trvalá 0/1 (ale i jiných) dochází ke zcitlivění cesty, což zvyšuje šance, že se porucha projeví.



Obrázek 7: Projev SEU jako trvalá 1(vlevo) a přerušení vodiče (vpravo)

Jednou z tezí práce je, že pomocí modelu trvalá 0/1 lze do jisté míry modelovat i poruchy SEU. SEU jsou v přepsaném obvodu modelovány jako poruchy trvalá 0/1 na vodičích vedoucích z buněk konfigurační paměti. Je-li v buňce uložena hodnota log. 1, pak je případná SEU modelována jako trvalá 0, v případě log. 0 jako trvalá 1. Ukázka korektního a nekorektního použití modelu trvalá 0/1 u SEU je zobrazena na obrázku 7 vlevo, respektive vpravo.

Přepis prvků se nepříznivě projevuje na velikosti obvodu, což komplikuje generování testu. Každý LUT se přepíše na 60 hradel, zatímco booleovská funkce, kterou realizuje, může být vyrobena „levněji“. V kombinaci se zanesením značného počtu konstant do struktury obvodu se prodlužuje doba generování testu. ATPG při budování citlivých cest často naráží na konstanty a rychle vyčerpává limit návratů, který je nutné navýšit i o několik řádů. Doba běhu ATPG je několikanásobně delší, než pro obvody ASIC se srovnatelným počtem hradel. Řešením dlouhých běhů ATPG může být redukce redundantních prvků v přepsaném obvodu, případně využití ATPG založeném na principu jiném, než je zcitlivění cesty, např. na SAT [34], [11].

Realistické ověření testovacích vektorů vyžaduje přesný model struktury, který má k dispozici pouze výrobce. Použít simulátor poruch a ověřovat testy na modelu, ze kterého byly generovány, nepřináší informaci o skutečné kvalitě testu. Na druhou stranu taková simulace může poskytnout srovnání jednotlivých testovacích sad. Porovnání vektorů vygenerovaných pro přepsaný obvod s vektory, které byly vygenerovány pro původní obvody je zaznamenáno v tabulce 3.

Označení sloupců v tabulce 3 má následující význam. „FPGA“ jsou testy vygenerované pro obvody přepsané dle metodiky uvedené v předchozím textu. Označením „ASIC“ jsou rozuměny testy vygenerované pro původní strukturu. Zkratka ND v pátém sloupci znamená nedetekovatelné poruchy. Poslední dva sloupce udávají dobu, po kterou ATPG generovalo test včetně spuštění prostředí a zápisu výsledků. Hodnoty posledního sloupce jsou uvedeny

v sekundách. Rovnice 1,2,3 definují pokrytí testu, pokrytí poruch a efektivitu ATPG tak, jak jsou uvedeny v [31].

Tabulka 3: Srovnání testovacích vektorů

Obvod	Počet vektorů		Počet poruch v obvodu		Max. pokrytí poruch [%]	Pokrytí testu [%]		Efektivita ATPG [%]		Generování [h:m:s]	
	FPGA	ASIC	Celkem	ND		FPGA	ASIC	FPGA	ASIC	FPGA	ASIC
c1355	211	123	41712	23607	56,60	90,08	69,03	95,69	86,56	03:46	0,898
c1908	275	148	44544	22836	51,27	89,75	83,19	95,01	91,81	02:50	0,931
c3540	447	162	111730	50373	45,08	85,27	78,38	91,91	88,13	01:13	1,183
c432	206	47	32288	16981	52,59	82,52	68,65	91,71	85,14	12:10	0,938
c499	171	101	41602	24330	58,48	88,36	68,13	95,17	86,77	01:58	1,090
c880	145	37	53234	30116	56,57	86,91	75,18	94,31	89,22	00:03	0,809
s1488	283	120	81028	39056	48,2	86,43	80,91	92,97	90,11	00:03	1,047
s1494	294	122	79844	38301	47,97	85,12	79,53	92,26	89,35	00:03	0,843
s832	243	113	42328	23208	54,83	84,56	77,55	93,03	89,86	00:02	0,821

$$Pokrytí\ testu = \frac{detekované\ poruchy}{detekovatelné\ poruchy} \quad (1)$$

$$Pokrytí\ poruch = \frac{detekované\ poruchy}{všechny\ poruchy} \quad (2)$$

$$Efektivita\ ATPG = \frac{řešitelné\ poruchy}{všechny\ poruchy} \quad (3)$$

Všechny testy byly generovány v ATPG TetraMax s použitím modelu trvalá 0/1 na počítači s Intel Xeon L5640 @ 2.27-2.8GHz s 32GB RAM. ATPG bylo nastaveno na generování testu se 100 % pokrytím, vysokou prioritou slučování vektorů a s limitem návratů 2000000.

2.5 Komprese testovacích dat

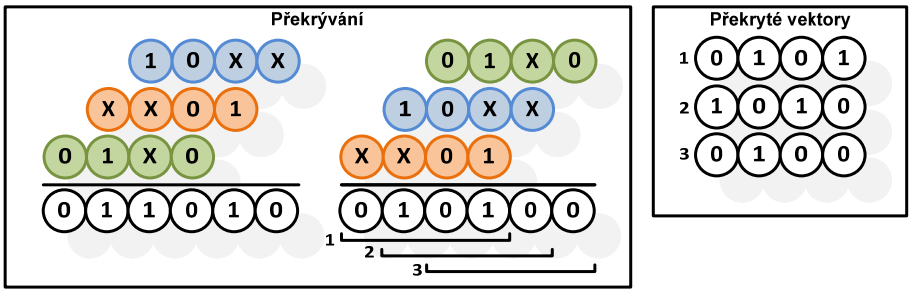
Cílem dělení obvodu je rozdělit ho na takové části, které umožní implementaci TPG i ORA bez nutnosti okolním blokům přidělovat větší prostor nebo vytvářet mechanismy přenosu testovacích vzorků ze systémové paměti. Velikost okolních modulů je vždy dostatečná pro TPG založená na LFSR, ale je omezující pro deterministický generátor založený na čítači a paměti ROM. Kapacita ROM v TPG nemusí být dostatečná k uložení všech vektorů a v některých případech je ji nutné navýšit. Abychom přístup do paměti nebo počet rekonfigurací omezili, je možné testovací vektory komprimovat.

Pro snížení objemu testovacích dat byl použit kompresní algoritmus COMPAS , který testovací data komprimuje pomocí překládání vektorů. Princip překládání vektorů je založen na empirickém zjištění, že většina testovacích vektorů bývá z velké většiny vyplněna nedefinovanými hodnotami. Při budování citlivé cesty není vždy nutné ošetřit všechny primární vstupy obvodu.

COMPAS ke své činnosti vyžaduje seznam poruch v obvodu a k nim příslušejících testovacích vektorů, které jsou, pokud možno, zaplněny nespecifikovanými hodnotami. Nejprve se snaží jednotlivé vektory slučovat. Pokud ve slučování neuspěje, posouvá vzájemně vektory (viz obrázek 8) tak, aby mohl překrýt alespoň jejich části. Při každém kroku překrytí je původní vektor posunut v pomyslném skenovacím řetězci o jednu pozici. Nově vzniklý vektor (s nespecifikovanou hodnotou na svém konci) je možné sloučit s dalším vektorem ze seznamu. Když algoritmus s hledáním vhodného vektoru uspěje, překryté vektory a detekované poruchy odstraňuje ze seznamu.

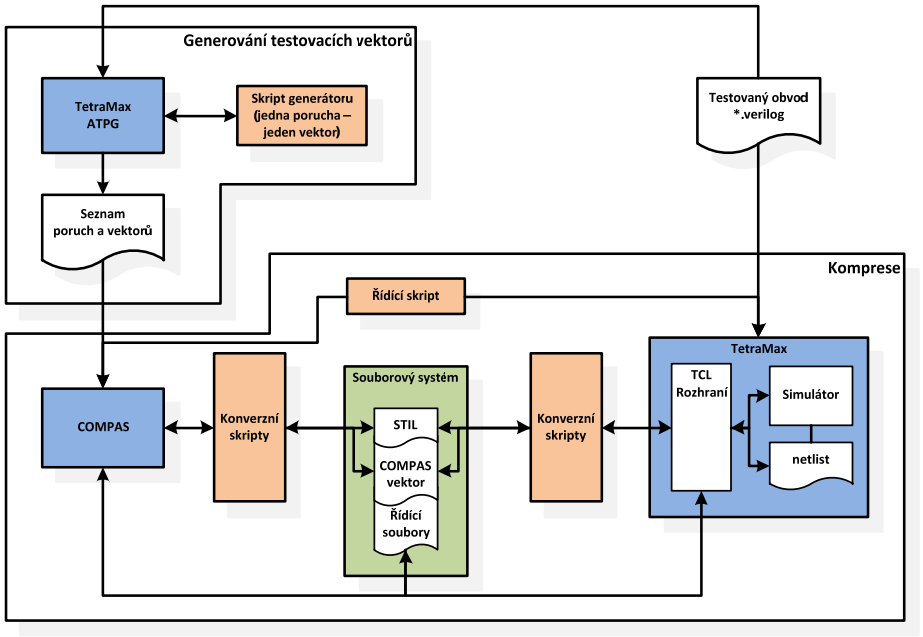
Počet vektorů, které se překrýváním vytvoří, je oproti jiným metodám [33] vysoký, ale vektory vygenerované COMPAS mají zajímavou vlastnost – každý vektor se od předchozího liší pouze v jednom bitu. *N* testovacích vektorů o délce *m* je ve vhodném dekompresním obvodu

možné rekonstruovat z binární sekvence o délce $m+N$ tak, že každý vektor bude vytvořen z předchozího vektoru posunutého o jeden bit a na konci doplněného nový bit ze zkomprimované sekvence. K dekompresi je zapotřebí posuvného registru o délce m, případně nevyužitého skenovacího řetězce se zpětnou vazbou o stejné délce.



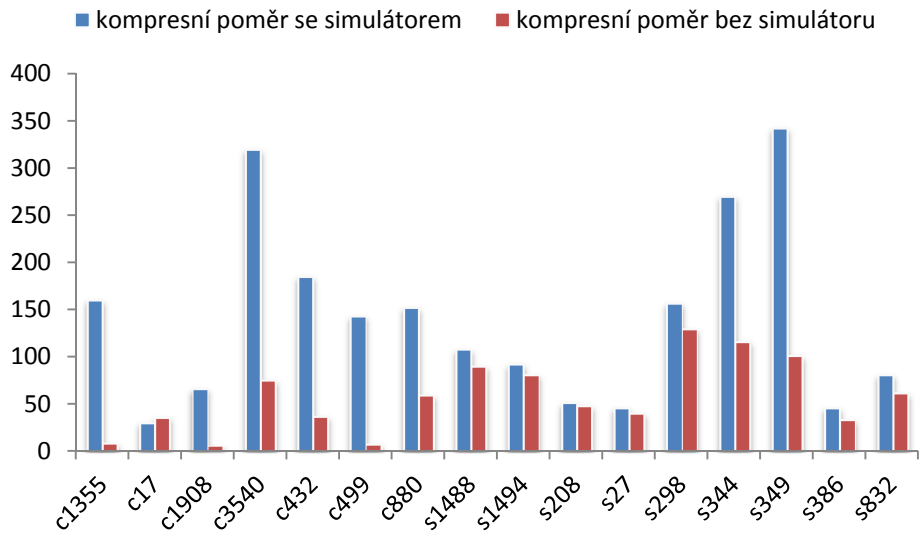
Obrázek 8: Ilustrace možných překrytí a výsledných vektorů algoritmu COMPAS

COMPAS může pracovat ve dvou režimech – se simulátorem poruch a bez něj. Použití simulátoru významně ovlivňuje délku zkomprimované sekvence. Interně obsahuje COMPAS simulátory FSIM a HOPE, které slouží pro práci se soubory formátu BENCH, potažmo pro spolupráci s ATPG Atalanta. Jelikož Atalanta řadu obvodů nebyla schopna zpracovat, bylo nutno vytvořit nové rozhraní mezi COMPAS a interním simulátorem TetraMax.



Obrázek 9: Blokové schéma rozhraní COMPAS – TetraMax

Spolupráce COMPAS s ATPG je zobrazena na obrázku 9. Pro testovaný obvod je nejprve vygenerován seznam všech poruch. Seznam je automaticky zpracován a dle něj je vytvořen skript generující vektory pro interpret TCL jazyka, který je integrován v ATPG. Skript je řízen výsledky běhu ATPG, které je v počátku nastaveno na generování vektorů s nespecifikovanými hodnotami s nízkým limitem návratů. Nedaří-li se generátoru vektor sestavit, limit návratů skript navyšuje tak dlouho, dokud není překročen uživatelem zadaný limit. Poté se opouští od vektorů s nespecifikovanými bity a skript nastaví ATPG pro generování standardních vektorů. Samotná komprese je realizována v druhé fázi. COMPAS postupně předkládá konverzním skriptům vektory (kandidáty na překrytí), které jsou konvertovány, v TetraMax simulovány a COMPASU je (opět přes konverzní skripty) navrácen seznam detekovaných poruch. Na obrázku 10 je graficky znázorněno porovnání kompresních poměrů s bez použití interního simulátoru TetraMax.



Obrázek 10: Porovnání kompresních poměrů při a bez použití simulace

2.6 Generátory a analyzátory

S pomocí testovacích dat získaných v předchozí kapitole a rozdělených a upravených obvodů je možné vytvořit pro každý modul unikátní generátor testovacích vektorů a analyzátor odezev. Představená metoda explicitně nespecifikuje, jakým způsobem mají být TPG a ORA implementovány. Dynamicky rekonfigurovatelné programovatelné obvody díky své flexibilitě poskytují oproti obvodům s pevnou strukturou při takto volném zadání nespornou výhodu. Různé varianty generátorů je možné ukládat do systémové paměti a dynamicky je dle potřeb testu měnit. V průběhu životního cyklu výrobku je dokonce možné navrhnout jejich nové varianty a bez větších potíží je do systému včlenit. Při návrhu je nutno splňovat základní parametry, jako jsou velikost, časování a pinová kompatibilita s původním obvodem.

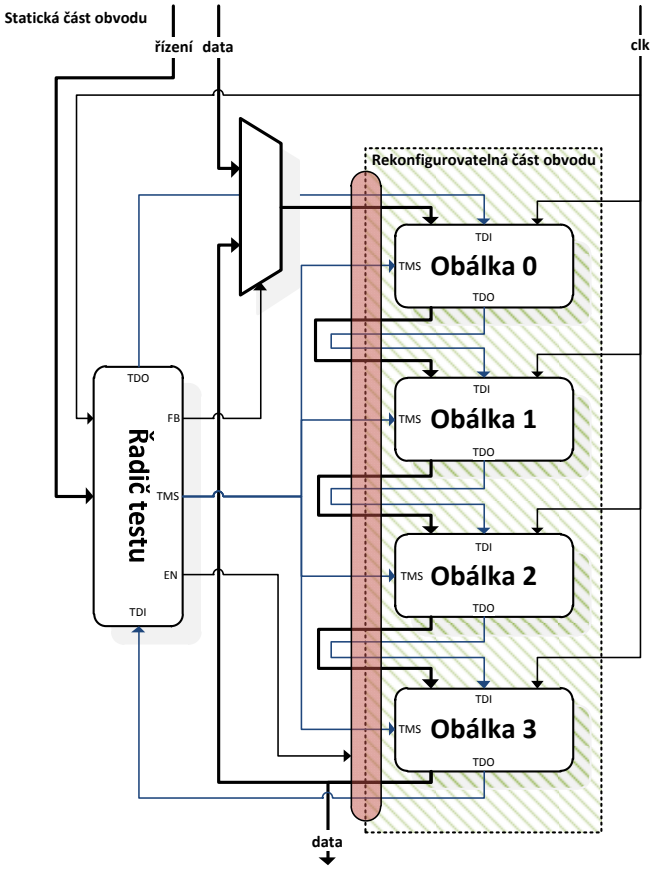
2.6.1 Obálka modulu

Obálka modulu unifikuje přístup k jednotlivým částem obvodu – funkčním modulům, generátorům vektorů a analyzátorům odezev. Rozhraní každé obálky je tvořeno původními primárními vstupy a výstupy funkčních modulů a signály řízení testu. Řídící signály jsou tři a obdobně jako u standardu JTAG jsou pojmenovány názvy TMS (Test Mode Select), TDI (Test Data In) a TDO (Test Data Out). Vstup TDI představuje sériové datové rozhraní do obvodu, výstup TDO je jeho výstupním ekvivalentem. Řídící vstup TMS řídí stavové automaty uvnitř decentralizovaného testeru. Na obrázku 11 je zobrazeno principiální schéma propojení obálek.

2.6.2 Generátory testovacích vektorů

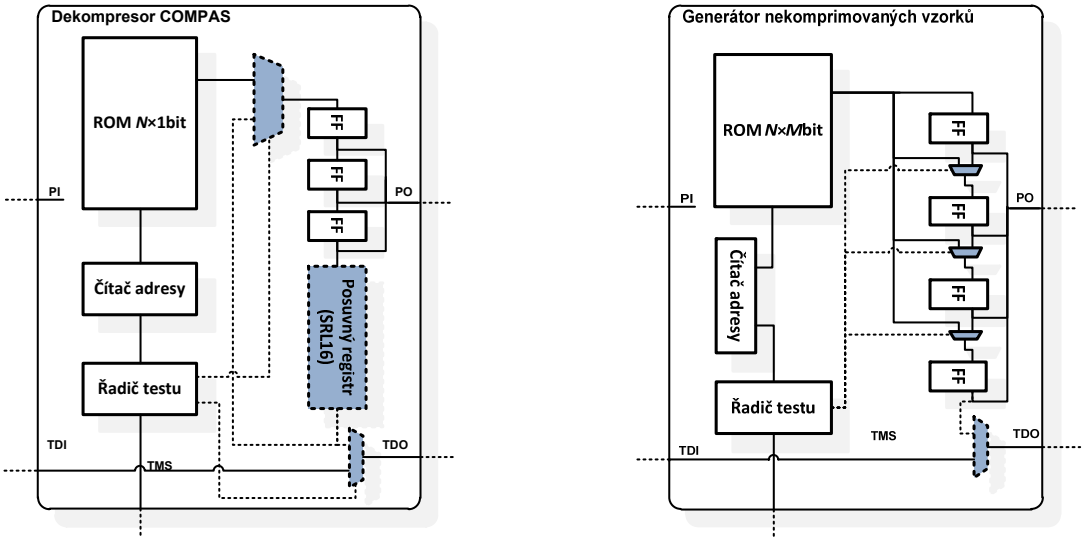
V práci byly navrženy, implementovány a nasazeny tři verze generátoru testovacích vektorů: Generátor pseudonáhodných vzorků s využitím posuvného registru s lineární zpětnou vazbou, Deterministický generátor s řadičem a pamětí ROM pro vzorky komprimované algoritmem COMPAS a nekomprimované vzorky.

Generátor založený na posuvném registru s lineární zpětnou vazbou je vytvářen skriptem *tpg_lfsr*, jež analyzuje počet primárních vstupů modulů a dle nich vybírá z tabelovaných hodnot primitivní polynom o minimálním počtu členů [10], jehož binární reprezentaci použije k zavedení zpětných vazeb v generické šabloně VHDL souboru, kterou v obálce nahradí funkční popis jádra. Společně se zpětnovazebním posuvným registrem je do obvodu vložen i distribuovaný řadič testu.



Obrázek 11: Zapojení obálek

Druhým typem generátoru je deterministický generátor implementovaný jako dekompresor sekvence COMPAS. Dekompresor se skládá ze čtyř hlavních komponent – generické paměti ROM, čítače adres, dekompresního posuvného registru a stejně jako u pseudonáhodného generátoru z řadiče testu.



Obrázek 12: TPG ve variantách dekompresoru (vlevo) a generátoru nekomprimovaných vzorků (vpravo)

Paměť ROM je implementována pomocí vyhledávacích tabulek a je v konfiguraci 1 bit × *N*, kde N je počet bitů sekvence. Obsah paměti je inicializován přímo ve VHDL kódu

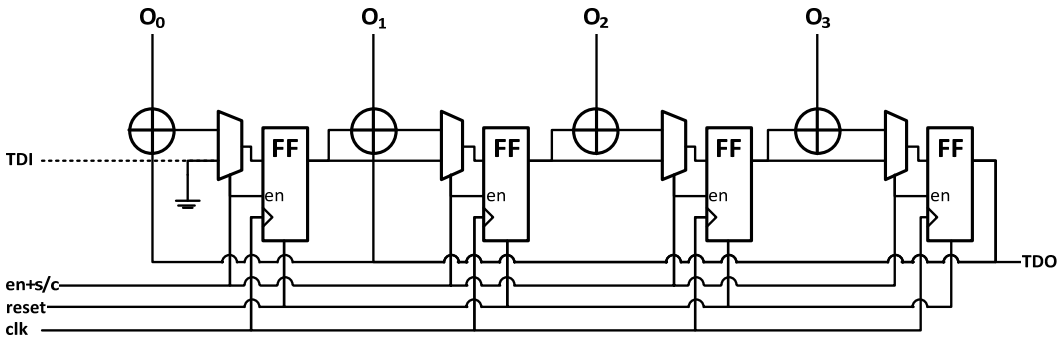
z *teststream* souboru, který byl vygenerován programem COMPAS. Schéma dekompresoru je zobrazeno na obrázku 12 vlevo.

Podobně jako dekompresor COMPAS je řešen i generátor nekomprimovaných deterministických testovacích vektorů (viz obrázek 12 vpravo). Obsahuje stejný řadič testu, ale čítač adresy, registr a paměť uchovávající testovací vzorky jsou odlišné. Paměť je v konfiguraci Mbit ×N, kde N je počet testovacích vektorů a M je počet primárních vstupů testovaného modulu. Paměť vzorků je tvořena výhradně z vyhledávacích tabulek.

2.6.3 Analyzátory odezev

V práci byly navrženy a nasazeny dva typy analyzátoru odezev: posuvný registr s paralelní předvolbou a vícevstupový příznakový registr (MISR). První obvod není analyzátorem v pravém slova smyslu, poněvadž neprovádí příznakovou analýzu, ale pouze snímá a vysouvá sejmuté odezvy. Vzhledem k pomalému sériovému přístupu není vhodný pro reálné nasazení, vyjma ladících účelů, kvůli kterým byl vytvořen.

Druhý ORA je implementován jako MISR a jeho schéma je zobrazeno na obrázku 13. Skládá se z posuvného registru s lineární zpětnou vazbou, do kterého jsou paralelně přes hradla XOR přivedeny výstupy (O_0až O_3 na obrázku 13) testovaného modulu. Implementovaný MISR pracuje ve dvou režimech: v režimu příznakové analýzy a režimu vysouvání vzorku. Obvod je řízen hodinovým signálem, signály reset, clock_enable a shift.

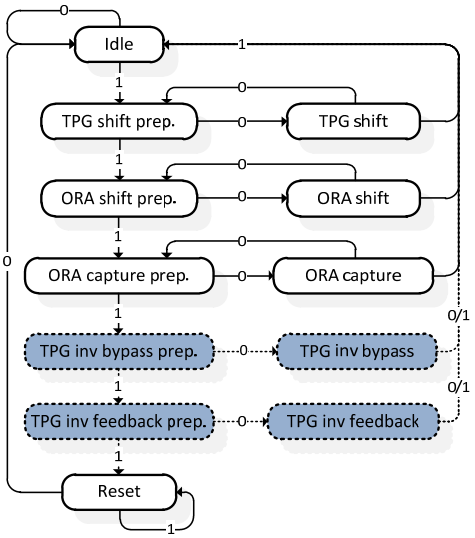


Obrázek 13: Přibližné schéma MISR

2.6.4 Distribuovaný řadič testu

Řadič testu je shodný jak pro generátory testovacích vektorů, tak pro analyzátory odezev. Vstupy řadiče jsou hodinový signál a řídící signál testu (TMS) synchronizovaný na náběžnou hranu hodinového signálu. Výstupy řadiče jsou signály *TPG_shift*, *ORA_shift*, *ORA_capture* a *chain_reset*. Výstupy jsou řízeny stavovým automatem typu „Moore“ zobrazeným na obrázku 14. Výstup *TPG_shift* ovládá adresní čítač nebo posuvný registr generátoru vektorů. Výstup řídí generování nového testovacího vektoru – v analyzátoru zapojen není. *TPG_shift* je přiřazena hodnota log. 1 pouze ve stavu TPG shift, který lze vyvolat přivedením sekvence 10X na vstup TMS.

Výstupy *ORA_shift* a *ORA_capture* jsou zapojeny pouze v modulu analyzátoru odezev a ovládají „ukládání“ nového vektoru do příznakového registru v případě *ORA_capture* a vysunutí odezvy z registru do nadřazeného řadiče testu v případě výstupu *ORA_shift*. Výstupy jsou zapojeny v modulu ORA, v modulu TPG zapojeny nejsou. ORA_capture i ORA_shift mohou být využity ve funkčním modulu, pokud je opatřen skenovacím řetězcem. Vyvolání posuvu je provedeno přivedením sekvence 110X, sejmutí odezvy pak s pomocí sekvence 1110X.



Obrázek 14: Stavový automat generátoru testovacích vektorů a analyzátoru odezev

Výstup *chain_reset* je společný pro generátor i analyzátor nastavuje počáteční hodnoty všech komponent – např. nastavení adresy na 0 v čítači adresy TPG, nebo inicializace LFSR registru v případě ORA. Reset je vyvolán sekvencí jedniček, která je úměrná počtu stavů automatu. Automat se vrací do výchozího stavu po přivedení log. 0 po této sekvenci.

Automat je navržen tak, aby bylo možno výše popsané povely provádět opakovaně, nebo sestavovat zajímavé uživatelské sekvence. Příkladem opakování může být vysouvání odezvy, kterou lze vyvolat sekvencí $11\{00\}_n1$, kde n je délka příznakového registru. Obdobně lze i sestavit sekvenci příkazů „vygeneruj vektor a ulož odezvu“ a to jako sekvenci 100110X.

Pro testování obvodů se skenovacím řetězcem by bylo nutno doplnit řadič o dva výstupy (zapojené na T klopné obvody), ze kterých by byl řízen zpětnovazebný multiplexor a bypass multiplexor. Doplněné stavy jsou na obrázku 14 označeny.

3 Experimentální ověření metody

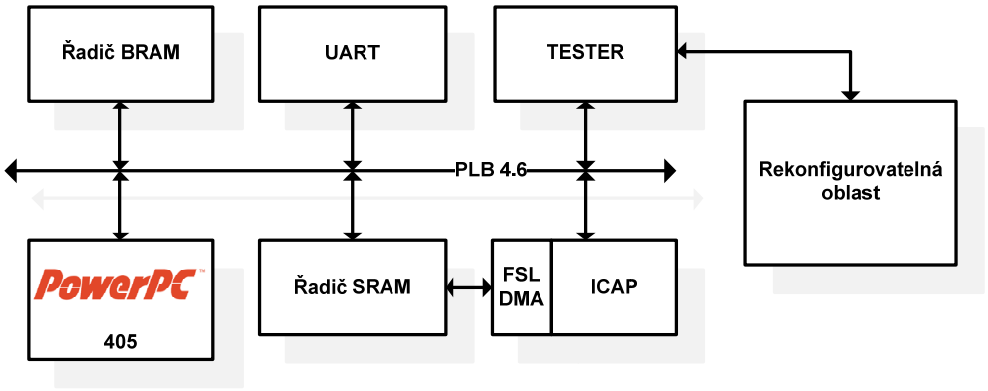
Za účelem ověření metody byly vytvořeny dva experimentální systémy. Na prvním systému byly pouze ověřovány kvality testovacích vektorů, druhý byl sestaven jako prototypové nasazení kompletní metody.

3.1.1 Experiment 1

První systém (viz obrázek 15) byl sestaven z procesorového jádra PowerPC405, sběrnice PLB, řadičů pamětí (externích SRAM a CompactFlash karty, interní BRAM), systémový časovač, testeru a testovaného obvodu. Systém komunikuje s PC prostřednictvím standardního rozhraní RS232. Přístup do konfigurační paměti obvodu je v systémech realizován modifikovaným řadičem ICAP, který je připojen přímo na řadič externí paměti SRAM. Oproti řešení dodávanému výrobcem hradlového pole je rychlost rekonfigurace navýšena téměř čtyřnásobně. Systémy byly implementovány na vývojové desce ML403 osazené obvodem Virtex-4FX12 s pomocí vývojových nástrojů ISE a XPS verze 9.2.04i PR7/8 na platformě x86.

Na prvním experimentálním systému bylo prováděno ověřování kvalit testovacích vektorů na benchmarkových obvodech sad ISCAS. Testování obvodů probíhalo tak, že byl do záložní kopie částečného bitsteramu vložen poškozený bit, konfigurační data byla nahrána do paměti zařízení a byl započat test. Po ukončení testu byl do konfigurační paměti FPGA nahrán záložní nepoškozený bitstream. Obvod byl znovu otestován, zdali se v něm nevyskytují trvalé

poruchy. Velikost všech částečných bitstreamů byla 144320 bitů, což odpovídá počtu všech možných SEU poruch v konfigurační paměti, která přísluší testovanému obvodu.

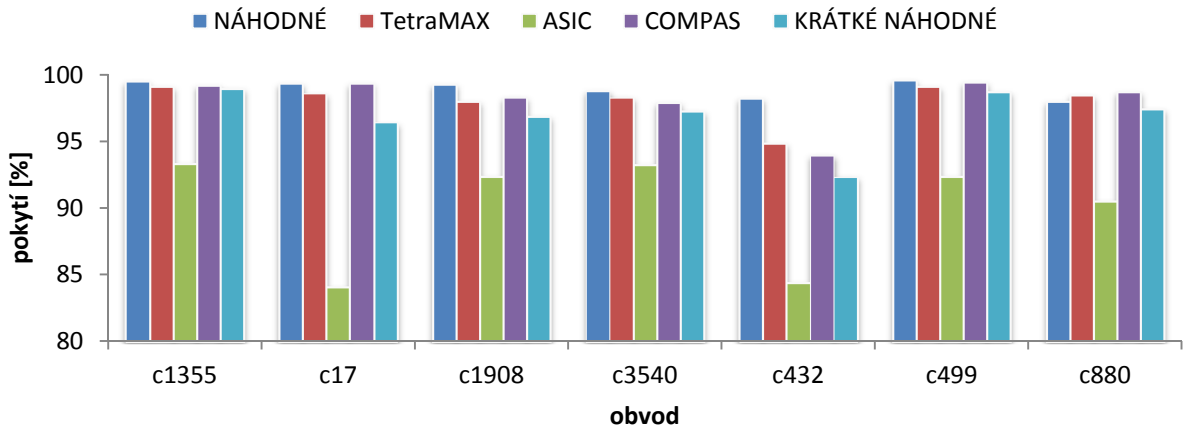


Obrázek 15: Blokové schéma systému 1

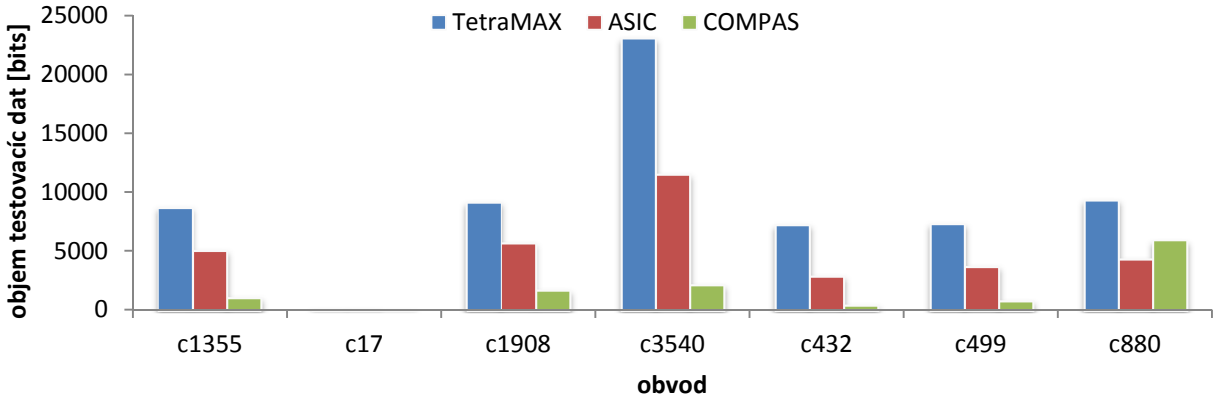
Ověřováno bylo několik sad testovacích vektorů. První sada byla vygenerována v ATPG TetraMAX se standardními nastaveními a vysokým stupněm interní komprese (nastavení „merge high“). Druhá sada byla zkomprimována posloupnost COMPAS, třetí sada deterministických vektorů byla vygenerována v ATPG Atalanta pro původní variantu obvodu (sada označovaná ASIC). Čtvrtou sadou byla posloupnost pseudonáhodných testovacích vektorů (označená jako NÁHODNÉ) a pátou jejich zkrácená verze označovaná jako KRÁTKÉ NÁHODNÉ. Část výsledků experimentu jsou v tabulkách a na grafech na obrázcích 16 a 17. První graf zobrazuje pokrytí poruch testovacích sad, druhý objem dat, který jednotlivé testovací sady vyžadují (v grafu nejsou uvedeny NÁHODNÉ vektory, které lze generovat v hardware).

Tabulka 4: Výsledky experimentu 1 na ISCAS 85 obvodech

Obvod		**c1355**	**c17**	**c1908**	**c3540**	**C432**	**c499**	**c880**
Využití zdrojů [%]		31,84	2,15	33,98	84,77	25,20	31,84	40,23
Detekované poruchy		4100	138	5444	17461	3729	3965	5714
Detekované / Všechny poruchy[%]		2,84	0,1	3,77	12,1	2,58	2,75	3,96
Normalizované detekované poruchy [%]		8,92	4,45	11,10	14,27	10,26	8,63	9,84
Náhodné poruchy		22	1	35	134	56	18	63
NÁHODNÉ	**pokrytí[%]**	99,46	99,28	99,25	98,76	98,2	99,52	97,97
	vektorů	10000	10000	10000	10000	10000	10000	10000
TetraMAX	**pokrytí[%]**	99,07	98,55	97,94	98,24	94,8	99,04	98,42
	vektorů	212	17	277	460	201	179	156
	délka [bity]	8692	85	9141	23000	7236	7339	9360
ASIC	**pokrytí[%]**	93,29	84,06	92,32	93,22	84,39	92,31	90,51
	vektorů	124	7	173	231	80	91	72
	délka [bity]	5084	35	5709	11520	2880	3731	4320
COMPAS	**pokrytí[%]**	99,15	99,28	98,24	97,89	93,91	99,42	98,67
	vektorů	1099	26	1668	2182	423	824	5926
	délka [bity]	1099	26	1668	2182	423	824	5926
KRÁTKÉ NÁHODNÉ	**pokrytí[%]**	98,93	96,38	96,84	97,22	92,3	98,64	97,37
	vektorů	1099	26	1668	2182	423	824	5926



Obrázek 16: Srovnání pokrytí SEU pro různé sady testovacích vektorů a obvodů



Obrázek 17: Srovnání objemu testovacích dat pro různé sady testovacích vektorů a obvody

Závěrem prvního experimentu je, že všechny testy, které byly generovány pro strukturní testování FPGA obvodu jsou schopny odhalit dostatečný počet poruch způsobených změnami v konfigurační paměti. Z hlediska navržené metody se nejvhodnějším testem jeví test COMPAS (ačkoliv byla použita vývojově starší verze, jež neužívá simulátor poruch). Vysoký počet testovacích vektorů protestuje i poruchy, které nejsou zahrnuty v modelu obvodu, pro který byl test vytvořen. Způsob rekonstrukce vektorů je hardwarově nenáročný a paměťové nároky generátoru testeru jsou nízké. Experiment zdárně odpověděl i na otázku, zdali je možno testování obvodu FPGA pomocí představené metody realizovat.

3.1.2 Experiment 2

Druhý systém byl sestaven jako prototypové nasazení kompletní metody. Na rozdíl od předchozího systému není je systém vybaven tstovatelným jádrem v podobě FPU jednotky, která realizuje akcelerovaný výpočet součtu ve formátu IEEE754.

Jádro sčítačky bylo pomocí postupů uvedených v kapitole 2 rozděleno na čtyři rekonfigurovatelné moduly, pro něž byly vytvořeny deterministické testy. Byly vygenerovány čtyři sady testů, které budou v textu označovány následovně: sada COMPAS označuje zkomprimovanou testovací sekvenci, sada TetraMax testovací sekvenci vytvořenou pomocí standardního postupu v ATPG TetraMAX, sada LFSR bude označovat vektory, které generuje posuvný registr s lineární zpětnou vazbou a jménem „LFSR krátké“ bude označena sada

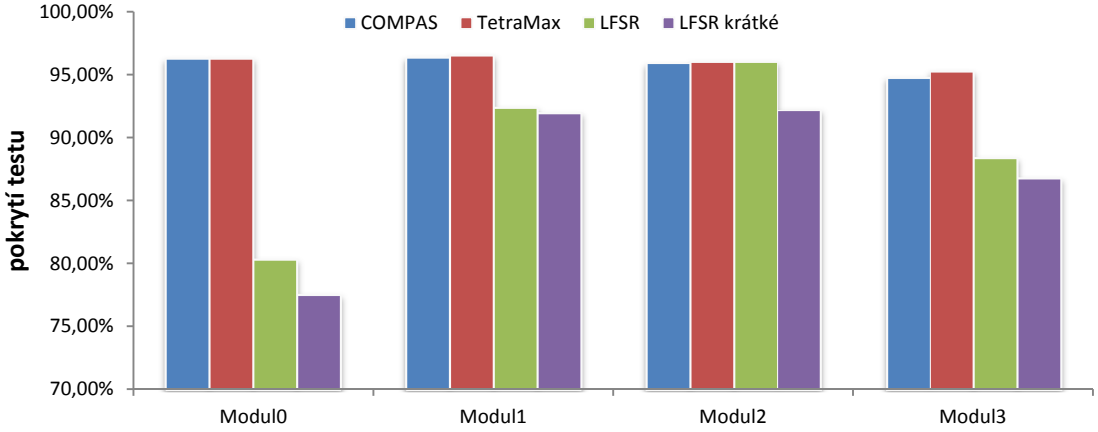
vektorů, která vychází ze sady LFSR, ale počet jejích vektorů je nižší – shodný s počtem vektorů sekvence COMPAS.

Testovací vektory jednotlivých sad byly nejprve podrobeny simulaci poruch na modelu obvodu v interním simulátoru TetraMax, viz tabulka 5. Tabulka s výsledky obsahuje údaje o pokrytí testu, pokrytí poruch a efektivitě ATPG tak, jak byly definovány v rovnicích (1), (2) a (3)

Tabulka 5: Strukturní pokrytí, délka a objem testovacích sekvencí

Obvod		COMPAS	TetraMax	LFSR	LFSR krátké
$Modul_0$	**Pokrytí testu [%]**	96,25	96,25	80,35	77,49
	Pokrytí poruch [%]	90,5	90,5	75,55	72,85
	Efektivita ATPG [%]	96,48	96,48	81,53	78,83
	Počet vektorů	1835	101	1835	10001
	Velikost ROM [bit]	1835	4646	0	0
$Modul_1$	**Pokrytí testu [%]**	96,33	96,56	92,41	91,99
	Pokrytí poruch [%]	92,75	92,97	88,97	88,57
	Efektivita ATPG [%]	96,47	96,69	92,69	92,29
	Počet vektorů	312	63	312	10001
	Velikost ROM [bit]	312	2520	0	0
$Modul_2$	**Pokrytí testu [%]**	95,89	96,03	96,03	92,17
	Pokrytí poruch [%]	92,18	92,31	92,31	88,6
	Efektivita ATPG [%]	96,05	96,18	96,18	92,47
	Počet vektorů	152	45	152	10001
	Velikost ROM [bit]	152	3150	0	0
$Modul_3$	**Pokrytí testu [%]**	94,78	95,27	88,41	86,8
	Pokrytí poruch [%]	89,99	90,45	83,94	82,41
	Efektivita ATPG [%]	95,05	95,51	88,99	87,46
	Počet vektorů	1993	309	1993	10001
	Velikost ROM [bit]	1993	19776	0	0

Graf pokrytí testu pro každý modul a každou sadu testovacích vektorů je vynesen na obrázku 18.



Obrázek 18: Pokrytí testu (strukturních poruch) modulů rozdělené sčítačky

V další fázi experimentu byly testovací sady ověřovány na injektoru poruch. Pro každý modul byly vytvořeny částečné bitstreamy ve variantách funkční jádro, generátory testovacích vektorů (COMPAS a LFSR) a analyzátory odezev (skenovací řetězec a MISR).

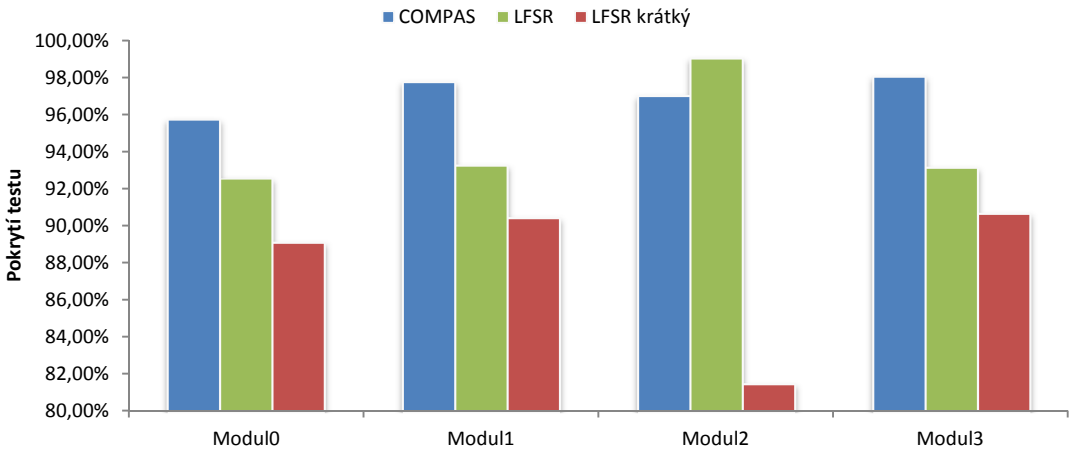
Testování probíhalo obdobně jako v experimentu 1 (viz 3.1.1). Ověřovány byly tři testovací sady – sada COMPAS, LFSR a LFSR krátké. Zpracované výsledky testu jsou zaznamenány v tabulce 6.

Tabulka 6: Pokrytí poruch konfigurační paměti

		$Modul_0$	$Modul_1$	$Modul_2$	$Modul_3$
Všechny testy	**Všechny poruchy**	5803	8247	8503	16521
	Náhodné poruchy	61	88	75	140
	Poruchy testu	697	766	1526	1553
	Pravidelné poruchy	5045	7393	6902	14828
COMPAS	**Všechny poruchy**	5608	8146	8308	16311
	Náhodné poruchy	55	84	63	118
	Poruchy testu	508	669	1343	1365
LFSR krátký	**Všechny poruchy**	5199	7527	6982	15078
	Náhodné poruchy	30	69	53	102
	Poruchy testu	124	65	27	148
LFSR	**Všechny poruchy**	5405	7769	8494	15489
	Náhodné poruchy	36	79	75	107
	Poruchy testu	324	297	1517	554
Pokrytí poruch	**COMPAS**	95,69%	97,76%	96,97%	98,01%
	LFSR krátký	89,07%	90,43%	81,49%	90,65%
	LFSR	92,52%	93,25%	99,01%	93,11%
Maximální pokrytí poruch		98,95%	98,93%	99,12%	99,15%
Efektivita pokrytí	**COMPAS**	96,71%	98,81%	97,83%	98,85%
	LFSR krátký	90,02%	91,41%	82,21%	91,42%
	LFSR	93,50%	94,25%	99,89%	93,90%

Kompletní testy byly provedeny vícenásobně a na základě výsledků byly vytvořeny tři kategorie poruch: „pravidelné poruchy“, „náhodné poruchy“ a „poruchy testu“. Mezi „pravidelné poruchy“ byly zařazeny poruchy, které byly detekovány vždy všemi testy a lze je považovat za poruchy detekované. Poruchy, které byly detekovány nepravidelně alespoň jedním testem, byly zařazeny do kategorie „náhodné poruchy“ (např. COMPAS 9(detekovaných)/10(provedených pokusů), LFSR 10/10, LFSR krátké 10/10) a jsou považovány za poruchy nedetekované. Do poslední kategorie patří poruchy, které byly detekovány při všech opakováních experimentu, ale ne u všech testů (např. COMPAS 10/10, LFSR 10/10, LFSR krátké 0/10). Poslední kategorie poruch byla zařazena do kategorie detekované.

Pokrytí testu je počítáno v souladu s rovnicí (2). Maximální pokrytí poruch je vypočítáno z parametrů vzniklých průnikem všech testů. Efektivita pokrytí je počítána jako poměr maximálního pokrytí poruch a pokrytí poruch každého testu. Graf pokrytí poruch je vynesen na obrázku 19.



Obrázek 19: Pokrytí poruch (konfigurační paměti) modulů rozdělené sčítačky

Společně s měřením kvality testovacích sad byla provedena i měření dostupnosti FPU. Měřena byla časová závislost testu na počtu aplikovaných testovacích vektorům, doba rekonfigurace okolních modulů a průměrná doba kompletního testu všech obvodů při použití TPG COMPAS. Údaje byly získány s pomocí systémového časovače s časovým rozlišením 10ns.

V tabulce 7 je zanesena naměřená závislost doby testu na počtu testovacích vektorů. Závislost je lineární, ale doba generování vektorů se může skokově zvyšovat, pokud bude jako TPG použit COMPAS dekompresor a kapacita jeho interní paměti nebude dostatečná. V případě generátoru založeného na LFSR bude závislost lineární vždy.

Tabulka 7: Časová závislost testu na počtu vektorů

Počet vektorů	1	1000	2500	5000	10000
Čas testu [µs]	10	80	185	360	714

Průměrná doba rekonfigurace – výměny obou okolních modulů – trvala *200µs*, kompletní test FPU pak v průměru *1943µs* a jeho jednotlivé fáze, tj. testy $modulu_0$ až $modulu_3$ trvaly *540*, *432*, *421* a *550µs* respektive.

Aplikace metody vnáší do testovaného obvodu určitou míru nežádoucí režie, která se projeví nárůstem spotřeby zdrojů FPGA a snížením pracovní frekvence obvodu. Majoritní podíl na nárůstu spotřebovaných zdrojů je způsoben začleněním částečné rekonfigurace a dvojí implementace testovaného obvodu. Pro vyčíslení míry režie bylo provedeno srovnávací měření spotřeby zdrojů tří odlišných systémů – systému bez rekonfigurace, systému se standardní rekonfigurací a skenovacími řetězci a prototypového systému. Výsledky časové analýzy jsou uvedeny v tabulce 8, statistiky spotřeby zdrojů v tabulce 9.

Tabulka 8: Kritické cesty

standardní rekonfigurace		**standardní rekonfigurace + scan**		**modifikovaná rekonfigurace**	
prvek	**zpoždění [ns]**	**prvek**	**zpoždění [ns]**	**prvek**	**zpoždění [ns]**
vstupní BM	3,829	vstupní BM	3,139	BM_0;mux;BM_3	10,003
IP jádro	7,764	IP jádro	15,220	IP jádro	13,105
celkem	11,593	celkem	18,359	celkem	23,108

Tabulka 9: Režie rekonfigurace v různých variantách systému

	varianta systému					
	bez rekonfigurace		**rekonfigurace + skenovací řetězce**		**prototypový systém**	
	počet SLICE	**% FPGA**	**počet SLICE**	**% FPGA**	**počet SLICE**	**% FPGA**
statická část	2107	38,51	2383	43,55	2479	45,30
oddělovací elementy	0	0	48	0,88	240	4,39
oblast rekonfigurace	0	0	768	9,36	1024	18,71
IP jádro	339	6,20	590	6,20	457	8,35
režie rekonfigurace	0	0	487	8,90	805	14,71

V systému, který lze chápat jako prototypové nasazení prezentované metody do reálných podmínek, dosahuje režie více než dvojnásobku spotřeby zdrojů původního IP jádra. Režie systému se skenovacími řetězci dosahuje režie poloviční. Nižší režie sytému se skenovacími řetězci je vykoupena zvýšením nedostupnost FPU během testu: doba dekomprese jednoho testovacího vektoru se kvůli sériovému přístupu k testovanému obvodu zvýší o několik řádů.

Snížení frekvence jádra sčítačky v prototypovém systému je pravděpodobně způsobeno přepisem obvodu a jeho druhou syntézou, u systému se skenovacími řetězci je způsobeno vložením skenovacích řetězců.

4 Závěr

Prezentovaná metoda aplikačně závislého testování FPGA obvodů se skládá z celé řady dílčích kroků, které dohromady vytvářejí velmi komplexní návrhový postup. V následujícím textu budou dílčí kroky shrnuty a bude analyzován jejich vliv na kvalitu aplikace metody.

Metoda pracuje s částmi testovaného obvodu, které jsou v textu označovány jako moduly a vznikají procesem dělením obvodu dodaného ve formě strukturního netlistu. Nástroje, které byly pro potřeby práce vyvinuty, buď umožňují dělit sekvenční obvody, které obsahují registry bez zpětných vazeb, anebo umožňují vkládat skenovací řetězce do obvodů, které obsahují meziregistrové zpětné vazby. Kombinace obou přístupů prozatím realizována není.

Samotné dělení obvodu je prováděno korektně – chování opětovně složeného obvodu odpovídá původnímu návrhu nejen dle behaviorálních simulací, ale i při nasazení na hradlovém poli. Dělení obvodů více než dostačuje potřebám práce, problematické je až další zpracování rozdělených modulů.

Při opětovné syntéze a implementaci dochází k podstatnému nárůstu spotřeby zdrojů hradlového pole. Režie, která dvojí implementací vzniká, silně limituje užití metody v menších hradlových polích; vznikají absurdní situace, kdy je část rozděleného obvodu větší než původní obvod, ačkoliv při dělení dochází k redukci obvodových prvků. Ve zkvalitnění způsobu dělení obvodů je možno hledat jeden ze směrů vylepšení práce, přičemž důraz by měl být kladen především na kvalitnější klasifikaci prvků užitých v nerozděleném obvodu a na analýzu jejich vzájemného rozmístění.

Implementované moduly jsou v dalším kroku metody podrobeny automatickému zpracování v nástrojích ATPG. Za tímto účelem jsou moduly přepsány ze strukturního popisu, který sestává ze standardních FPGA primitiv do popisu, který je tvořen primitivy ASIC.

Kvůli limitaci dané uzavřenou architekturou FPGA obvodů je vytvářen pouze přibližný strukturní model FPGA a i zvolený model poruch (trvalá 1/0) je limitován strukturní závislostí. Z analýzy experimentů se však ukazuje, že původní teze, že i přibližný model může být pro tvorbu kvalitních testovacích vektorů dostatečný, byla správná.

Model obvodu byl vytvářen s vědomím, že konfigurační paměť hradlového pole je nedílnou součástí jeho struktury. Jelikož výsledky pokrytí poruch strukturního testu korelují s výsledky, které byly získány na emulátoru poruch konfigurační paměti, lze model prohlásit za celkem věrohodný a lze předpokládat, že detekuje i strukturní poruchy, přestože to není možné dokázat.

Reprezentativnost výsledků emulace poruch konfigurační paměti, o kterou se předchozí myšlenku opírá, podporují výsledky nezávislých výzkumů (např. [26]) a údaje výrobce, které udávají poměr konfiguračních bitů ovlivňujících implementovaný obvod. Tento poměr se shoduje s výsledky, které byly získány na základě vlastních experimentů. Vytvořené testy detekují přesně ty poruchy konfigurační paměti, které mají co dočinění s funkcí obvodu, což lze považovat za úspěšné naplnění principu aplikačně závislého testování.

Vytvořené nástroje umožňují generování popisů pro celou řadu ATPG, ačkoliv se z výsledků experimentů ukazuje, že komerční ATPG nabízejí o něco vyšší výkon než jejich nekomerční představitelé. S drobnými úpravami by bylo možné přenést metodu na jinou architekturu hradlových polí, užít odlišného modelu poruch či využít ATPG, která nejsou založena na vytváření citlivých cest. Prvotní pokusy s nástroji prezentovanými v [11] již proběhly.

V rámci práce byly navrženy a úspěšně ověřeny různé implementace generátorů testovacích vektorů a analyzátorů testovacích odezev společně s nadřazeným řadičem testu. Experimentálně byl vytvořen systém na programovatelném čipu, který umožňuje pomocí navržené metody testovat běžná IP jádra. Nejen pro účely experimentálního systému byl zkonstruován výkonný řadič přístupu do konfigurační paměti hradlového pole, jež nabízí několikanásobně vyšší přenosovou rychlost, než řadič standardně dodávaný výrobcem.

Experimenty provedené na prototypovém systému dokazují realizovatelnost a funkčnost navržené metody, ale poukazují i na některé nedostatky.

Je to především režie spojená s nasazením částečné dynamické rekonfigurace, která se nepříznivě projevuje na celkové využitelnosti metody v praxi. Aplikací částečné rekonfigurace vznikají požadavky na dodatečné zdroje obvodu a zhoršuje se jeho celkový výkon. Omezení režie není zcela v kompetenci návrháře systému – je značná, i když je dodržena řada pravidel, které ji redukují. Využití novějších architektur FPGA redukci režie neposkytne, naopak ji paradoxně zvýší. Je nutno si uvědomit, že režií způsobenou částečnou dynamickou rekonfigurací netrpí pouze prezentovaná metoda, ale i jiné metody, jež jsou na částečné rekonfiguraci založeny, ať už účel rekonfigurace souvisí s testováním, anebo ne.

Závěrem lze konstatovat, že veškeré cíle práce se podařilo splnit. Byla vytvořena metoda aplikačně závislého testování FPGA obvodu, jež kombinuje výhody moderních programovatelných obvodů s nástroji, postupy a technikami BIST pro obvody ASIC, které byly pro potřeby práce upraveny. Nástroje a postupy, které byly pro metodu vytvořeny, jsou opakovatelně a univerzálně použitelné.

Bylo by však naivní konstatovat, že metoda, tak jak je prezentována, je úplná a práce na ní může ustat. Práce může být rozšířena a vylepšena v celé řadě směrů. Lze zkoumat využití moderních metod vytváření testovacích vektorů, aplikaci jemnozrnné rekonfigurace pro lepší kontrolovatelnost a pozorovatelnost testovatelného obvodu, vytváření strukturně podrobnějších modelů FPGA nebo možnosti jiných modelů poruch.

4.1 Přínos navržené metody

Metoda aplikačně závislého testování FPGA obvodů, která byla představená v předcházejícím textu, nabízí inovativní způsob testování programovatelných obvodů. Testování pomocí metody stojí na několika pilířích, z nichž nejvýznamnější jsou částečná dynamická rekonfigurace a užití zkomprimovaných deterministických testovacích vektorů.

Částečná dynamická rekonfigurace je v metodě využita k několika nezávislým činnostem: umožňuje přístup k testovanému obvodu, slouží jako transportní kanál přivádějící testovací vzorky, umožňuje provádění pravidelných oprav testovaného modulu na úrovni konfigurační paměti a dovoluje omezenou míru oprav implementovaného obvodu na strukturní úrovni. Metoda zásadně zvyšuje dostupnost a řiditelnost testovaného obvodu bez omezení činnosti ostatních částí obvodu, které testovány nejsou; je umožněno paralelní nezávislé testování několika obvodových celků v rámci jednoho FPGA. Metoda užitím částečné rekonfigurace pro test účelně navyšuje funkční hustotu programovatelného obvodu.

Zkomprimované deterministické testovací vektory, přinášejí nesporné výhody v podobě zkrácení doby testu, navýšení dostupnosti zařízení a snížení počtu nutných rekonfigurací; využití zkomprimovaných testovacích sekvencí poskytuje nejlepší poměr kvality testu vůči době testu, což dokazují provedené experimenty. Proces vytváření vektorů je uživatelsky ovlivnitelný a není omezen výběrem softwarového vybavení v podobě ATPG. Test je možno cílit na širokou paletu modelů poruch nebo lze různé testy kombinovat.

Vlastní publikace

[1] Rozkovec, M.: Přechod DyRESPIN architektury na platformu Xilinx Virtex-4, Proc. Of PAD 2007, Září 2007, Srní, ISBN 978-80-7043-605-9

[2] Rozkovec, M.: Koncept rekonfigurovatelné testovací architektury pro FPGA obvody. Proc. Of PAD 2008, Září 2008, Hejnice, Czech, pp. 37-44, ISBN 978-80-7372-378-1

[3] Rozkovec, M.: Implementation of Dynamically Reconfigurable Test Architecture for FPGA Circuits. Proc. of DDECS 2008, April 2008, Bratislava, Slovakia, pp.182-185, ISBN 978-1-4244-2276-0

[4] Rozkovec, M.; Novák, O.: Structural test of programmed FPGA circuits, Proceedings of the 2009 IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2009, art. no. 5012114, pp. 136-139

[5] Rozkovec, M.; Jeníček, J.; Novák, O.: "Application dependent FPGA testing method using compressed deterministic test vectors," On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International, pp.192-193, 5-7 July 2010

[6] Rozkovec, M.; Jeníček, J.; Novák, O.: "Application Dependent FPGA Testing Method," Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on, pp. 525-530, 1-3 Sept. 2010

[7] Jenicek, J.; Rozkovec, M.; Novak, O.: "Test vector overlapping based compression tool for narrow test access mechanism," Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on, vol., no., pp.387-392, 13-15 April 2011

Použitá literatura

[8] Abramovici, M.; Stroud, C. E.; Hamilton, C.; Wijesuriya, S.; Verma,V.: Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications, IEEE International Test Conference (TC), pp. 973-982

[9] Abramovici, M.; Stroud, C.E.: BIST-based test and diagnosis of FPGA logic blocks, IEEE Transactions on Very Large Scale Integration Systems 9, pp. 159-172

[10] Alfke, P., Xilinx Inc.: Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators, online [2011-07-08]

[11] Balcárek, J.; Fišer, P.; Schmidt, J.; "Test Patterns Compression Technique Based on a Dedicated SAT-Based ATPG," Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on, pp. 805-808, 1-3 Sept. 2010

[12] Carmichael, C.,Tseng C. W., Xilinx Inc: "Correcting Single-Event Upsets in Virtex-4Platform FPGA Configuration Memory", online [2011-07-08]

[13] Chapman, K., Jones, L.: SEU strategies for Virtex-5 devices, Xilinx Inc., XAPP864, [online]

[14] Glaser, U., Vierhaus, H.T.: MILEF: an efficient approach to mixed level automatic test pattern generation. Proc. Of the Conference on European Design Automation, Los Alamitos, CA, pp. 318-321

[15] Heron, O.; Arnaout, T.; Wunderlich, H.-J.; , "On the reliability evaluation of SRAM-based FPGA designs," Field Programmable Logic and Applications, 2005. International Conference on, vol., no., pp. 403- 408, 24-26 Aug. 2005, doi: 10.1109/FPL.2005.1515755

[16] Hlavička, J.: Diagnostika a spolehlivost, Skripta ČVUT, 1998, ISBN 80-01-01846-6

[17] „IEEE Standard Test Access Port and Boundary - Scan Architecture," IEEE Std 1149.1-1990 , vol., no., pp.0_1, 1990

[18] Jacobs, A., George, A.D., Cieslewski, G.: Reconfigurable fault tolerance: A framework for environmentally adaptive fault mitigation in space, FPL 09: 19th International Conference on Field Programmable Logic and Applications, art. no. 5272313, pp. 199-204

[19] Jeníček, J.: Komprese testovacích dat založená na překrývání vzorků. Disertační práce. TU Liberec 2008.

[20] Kastensmidt, F.L., Carro, L., Reis, R.: Fault-Tolerance Techniques for SRAM-based FPGAs, Springer, 2006, ISBN-10 0-387-31068-1

[21] Lee, H.K., Ha D.S.: Atalanta: an Efficient ATPG for Combinational Circuits, Technical Report, 93-12, Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1993

[22] Lesea, A., Drimer, S., Fabula, J.J., Carmichael, C., Alfke, P.: The rosetta experiment: Atmospheric soft error rate testing in differing technology FPGAs, IEEE Transactions on Device and Materials Reliability 5 (3), pp. 317-328

[23] Ma, S., Shaik, I., Fetherson,R.S.: Comparison of bridging fault simulation methods, IEEE International Test Conference (TC), pp. 587-595

[24] Mader, Z.: Diagnostický systém jádrově založených SoC obvodů s nízkými nároky na paměť. Disertační práce. TU Liberec 2008

[25] Novák, O., Gramatová, E., Ubar, R. and coll.: Handbook of testing electronic systems. Nakladatelství ČVUT, srpen 2005, 395 stran, ISBN 80-01-03318-X

[26] Pratt, B.H., Wirthlin, M.J., Caffrey, M., Graham, P., Morgan, K.: Noise impact of single-event upsets on an FPGA-based digital filter, FPL 09: 19th International Conference on Field Programmable Logic and Applications, art. no. 5272554, pp. 38-43
[27] Renovell, M., Faure, P., Portal. J.M., Figueras, J., Zorian, Y.: IS-FPGA : a new symmetric FPGA architecture with implicit scan, International Test Conference, 2001. Proceedings, Page(s):924-931
[28] Renovell, M., Figueras, J., Zorian, Y.: Testing the interconnect of RAM-based FPGAs, IEEE Design and Test of Computers 15 , pp. 45-50
[29] Straka, M.; Kastil, J.; Kotasek, Z.; , "Fault Tolerant Structure for SRAM-Based FPGA via Partial Dynamic Reconfiguration," Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on , vol., no., pp.365-372, 1-3 Sept. 2010
[30] Stroud,C.:A Designer's Guide to Built-In Self-Test, Kluwer Academic Publishers, 2002, ISBN 1-4020-7050-0
[31] Synopsys: TetraMAX® ATPG User Guide, Version D-2010.03, March 2010
[32] Tahoori, M.B, McCluskey, E.J., Renovell, M., Faure, P.: A Multi-Configuration Strategy for an Application Dependent Testing of FPGAs VLSI Test Symposium, 2004. Proceedings. 22nd IEEE, Page(s):154 – 159
[33] TetraMAX, [online], URL: www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/TetraMAXATPG.aspx
[34] Tille, D., Drechsler, R.: A Fast Untestability Proof for SAT-based ATPG, Proc. of the 2009 IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2009
[35] Wang, L.T., Stroud, C.E., Touba, N.A.: System on chip test architectures, Nanometer design for testability, Morgan Kaufmann Publishers, 2008, ISBN: 978-0-12-373973-5
[36] Xilinx Inc.: Development System Reference Guide 9.2i (ug628), [online] [2009-12-10]
[37] Xilinx Inc.: Device Reliability Report, third Quarter 2009[online] [2010-01-04].
[38] Xilinx Inc.: Virtex-4 FPGA User Guide (ug070),[online] [2009-12-10]
[39] Xilinx Inc.: XST User Guide for Virtex-4, Virtex-5, Spartan-3, and Newer CPLD Devices, [online] [2010-14-12]