TECHNICKÁ UNIVERZITA V LIBERCI

FAKULTA MECHATRONIKY, INFORMATIKY A MEZIOBOROVÝCH STUDIÍ

KOMPRESE TESTOVACÍCH DAT ZALOŽENÁ NA PŘEKRÝVÁNÍ VZORKŮ

AUTOREFERÁT DISERTAČNÍ PRÁCE

Disertant:	Ing. Jiří Jeníček
Studijní program:	P 2612 Elektrotechnika a informatika
Studijní obor:	2612V045 Technická kybernetika
Pracoviště:	Ústav informačních technologií a elektroniky
	Fakulta mechatroniky, informatiky a mezioborových studií
	Technická univerzita v Liberci
Školitel:	Prof. Ing. Ondřej Novák, CSc.

Rozsah disertační práce a příloh:

Počet stran:	114
Počet obrázků:	25
Počet grafů:	11
Počet tabulek:	13
Počet příloh:	1

Abstrakt

Tato práce pojednává o cílech a metodách komprese testovacích vektorů. Cílem práce je vytvoření nového algoritmu komprese testovacích vektorů tak, aby bylo možné urychlit výpočet, s důrazem na urychlení především u rozsáhlých obvodů. Práce vychází z již funkčního kompresoru testovacích vektorů COMPAS.

Nový algoritmus efektivně využívá struktury vstupních dat pro snížení časových i paměťových nároků komprese. Analýzou vstupních dat je zároveň dosaženo zlepšení kompresního poměru. Součástí práce je ověření efektivity použitých algoritmů na testovacích obvodech ISCAS 85, ISCAS 89 a ITC99.

Uvedené úpravy výrazně snížily dobu běhu programu i jeho systémové nároky při současném zkrácení délky komprimované modifikační posloupnosti a zachování pokrytí poruch.

Abstract

The thesis deals with methods and objectives of test pattern compression. The main target is to design a new compression algorithm, which will speed up the compression, especially in case of large circuits. The work is based on functional test pattern compressor called COMPAS.

The new algorithm effectively uses the input data structure to lower both compression time and memory requirements. Improved analysis of the input data allows further shortening of the compressed sequence. The thesis also includes algorithm efficiency comparison based on measuring of the time consumed for compression of test vectors for ISCAS 85, ISCAS 89 and ITC99 benchmark circuits.

The introduced modifications have significantly decreased program runtime, lowered program system requirements, shortened compressed sequence length and maintain fault coverage.

Obsah

1. Úvod

Ústav informačních technologií a elektroniky (ITE) na fakultě mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci se dlouhodobě zabývá problematikou testování číslicových obvodů. Jedním ze zaměření výzkumu je problém vestavěných generátorů testů.

Tato práce se zabývá návrhem nového algoritmu komprese testovacích dat, který je založený na překrývání testovacích vektorů [16]. Nový algoritmus urychlí a zlepší kompresi a sníží systémové nároky. Práce staví na již funkčním programu COMPAS (COmpressed test Pattern Sequencer), který byl již dříve členy týmu vyvinut a který úzce spolupracuje se simulátorem poruch.

2. Systém COMPAS

COMPAS – Compressed Test Pattern Sequencer je software pro kompresi testovacích vektorů pro obvody se sériovým diagnostickým přístupem. Program slouží pro kompresi testovacích dat pro kombinační obvody nebo pro obvody s plným skenem. Je zaměřen na zpracování testovacích dat pro model trvalých poruch, přičemž využívá popis obvodu na úrovni hradel a simulátor poruch. Vstupní data mají formu dvojic, které jsou tvořeny nekomprimovaným testovacím vektorem a příslušnou poruchou. Je vhodné, aby testovací vektor obsahoval co největší množství nespecifikovaných bitů, protože ty lze nejsnáze překrýt.

Algoritmus obecně umožňuje kompresi libovolných testovacích vektorů. Vektory mohou obsahovat jen specifikované bity, pak je ale obtížné je překrýt a komprese nedosahuje dobrých výsledků. Vektory mohou být vytvořeny pro testování jiných než trvalých poruch. V tom případě ale není možné využít úzké spolupráce se simulátorem poruch, a je tak dosaženo horší komprese. Simulátor není možné použít ani v případě, že není k dispozici popis obvodu.

Kompresor COMPAS je platformě nezávislý, testován byl na platformách x86, x86-64 a sparc, pod operačními systémy Windows XP, Linux a SunOS ve verzích pro 32 i 64 bitů. Zdrojový kód je v jazyce C.

2.1. Komprese

Při kompresi dat systémem COMPAS s běžným nastavením nedochází ke ztrátě hlavní testovací informace, tedy pokrytí poruch, z hlediska pokrytí poruch se jedná o bezeztrátovou kompresi. Po dekompresi komprimovaných dat nedostaneme všechny původní vektory, z pohledu testovacích vektorů je tedy komprese ztrátová.

Jako vstupní data pro program slouží nekomprimované testovací vektory obsahující nespecifikované vstupní bity (don’t care bity). Vzorky mají formu dvojice porucha –

testovací vektor, tedy každé poruše přísluší právě jeden testovací vektor s co nejvíce nespecifikovanými bity. Výstupem programu je posloupnost nul a jedniček neobsahující žádné nespecifikované bity, která tvoří komprimovanou testovací posloupnost. Po kompresi programem COMPAS může být výsledná komprimovaná testovací posloupnost využita např. pro testování SoC.

Metoda komprese spočívá v postupném hledání testovací posloupnosti po jednom bitu z předem připravené sady testovacích vektorů – úplného testu. Pro kombinační obvody tato posloupnost při vlastním testu vstupuje do skenovací smyčky, jejíž velikost je stejná jako počet vstupů obvodu, pro sekvenční obvody je vhodné použít některou z testovacích architektur, např. RESPIN [26].

Na začátku algoritmu máme seznam zatím nedetekovaných poruch obvodu, seznam všech testovacích vektorů tvořících úplný test a ve skenovací smyčce jsou samé nuly. Krok algoritmu pak probíhá tak, že je nalezen další jeden bit posloupnosti, dále je provedena simulace a ze seznamu poruch jsou odstraněny ty, které jsou simulací detekovány. Algoritmus končí tehdy, když je seznam nedetekovaných poruch prázdný.

Kompresní algoritmus COMPASu začíná ze známého stavu, kterým je většinou vynulovaný skenovací řetězec. Dá se totiž předpokládat, že test obvodu začíná zapnutím napájení a vynulováním. Komprese ale může v případě nutnosti začít i s jiným stavem skenovacího řetězce.

Použitý algoritmus nepoužívá zpětné prohledávání, nevrací se k bitům vygenerovaným v předchozích krocích. Jakmile je jednou komprimovaný bit vygenerován, je zařazen do výsledné posloupnosti a už není nikdy změněn. Toto omezení bylo zavedeno kvůli rychlosti. Při prvotních experimentech byla vyzkoušena i metoda se zpětným prohledáváním, z hlediska rychlosti byla ale naprosto nepoužitelná (komprese dat trvala hodiny pro obvody o desítkách hradel). Obecně by kompresní algoritmus mohl vyzkoušet všechna myslitelná pořadí seřazení vektorů, tím by se kompletně prohledal stavový prostor a bylo by nalezeno optimální řešení. Hledání by ale trvalo příliš dlouho, protože by bylo nutné vyzkoušet všechny permutace pořadí vektorů.

Tím, že nedochází k návratu k předchozím bitům, dochází k výraznému zrychlení algoritmu. Algoritmus pracuje tak, že v každém kroku ohodnotí kritériem všechny možnosti, a vybere aktuální lokální minimum s tím, že existuje šance, že trvalým vybíráním lokálního minima bude nalezeno minimum globální. Jedná se tedy o hladový algoritmus (greedy search). Nalezené řešení není nejmenší možné, je ale dostatečně kvalitní.

2.2. Dekomprese

Dekomprese testu v kombinačních obvodech potom tak, že posloupnost bitů vstupuje po jednom bitu do řetězce klopných obvodů, tvořených např. Boundary Scanem. Při

posunutí posloupnosti o jeden bit je přitom vygenerován celý testovací vektor, čímž dochází ke značné úspoře paměti testeru.

Dekomprese při použití sekvenčního obvodu vyžaduje složitější dekompresní hardware, jedním z řešení je RESPIN architektura. V obou případech je dekompresní hardware tvořen jednoduchým řetězcem klopných obvodů. Architektura RESPIN umožňuje dekomprimovat pomocí vestavěného testovacího jádra ETC výslednou testovací posloupnost na testovací vzorky pro testované jádro CUT, a je plně kompatibilní s normou IEEE P1500.

2.3. Výhody

Výhodou této metody jsou především vysoká komprese použitých testovacích vektorů. Při kompresi se nevyužívá pouze kompakce testovacích vektorů, ale zároveň jejich posuv, což značně zvyšuje stupeň komprese. Dalšími výhodami jsou rychlé a snadné testování a velmi jednoduchý testovací hardware.

3. Cíle disertační práce

Prvním úkolem bylo z důvodu rychlosti integrovat do COMPASu simulátor poruch, protože spouštění externího simulátoru má příliš velkou režii.

Následným cílem bylo vyvinout lepší kompresní metodu a optimalizovat systém COMPAS pro dosažení maximální rychlosti komprese testovacích dat, zároveň minimalizovat délku výsledné testovací posloupnosti. Tím bude umožněna komprese testovacích dat pro velké obvody.

Dalším úkolem bylo přizpůsobit kompresní systém COMPAS více skenovacím řetězcům, které jsou v RESPINu používány. Větší obvody zpravidla pro testování používají více skenovacích řetězců, což je výhodné z hlediska rychlosti testu.

Dalším cílem bylo ověřit schopnost komprimovat testovací data i pro jiný model poruch než trvalé poruchy, a ověřit nezávislost kompresního algoritmu na použitém ATPG.

Posledním úkolem bylo zlepšit výkonnost COMPASu při využití moderních víceprocesorových výpočetních systémů, protože vícejádrové procesory se už staly standardním vybavením počítače.

Disertabilní jádro bude spočívat v novém kompresním algoritmu.

3.1. Stručný popis nového algoritmu

Prvním krokem je vygenerování souboru testovacích vektorů TPL (Test Pattern List) společně se souvisejícím seznamem nedetekovaných poruch UFL (Undetected Fault List) pro daný testovaný obvod. K tomu je nutné použít takový ATPG, který je schopen generovat nekompaktní testovací vektory. Pro každou poruchu, kterou chceme

detekovat, je nutné vygenerovat nejméně jeden testovací vektor s nespecifikovanými bity.

Dalším krokem provedeným před samotnou kompresí je analýza testovacích dat. Při ní jsou určeny obtížně detekovatelné poruchy a testovací vektory jsou ohodnoceny podle schopnosti detekovat obtížně detekovatelné poruchy. Využití této dodatečné informace vede k zlepšení kompresního poměru.

Hlavní cyklus algoritmu začíná (bez újmy na obecnosti) tím, že je skenovací řetězec vynulován, a první testovací vektor je tedy tvořen samými log. nulami (algoritmus umožňuje začít kompresi s libovolným známým stavem skenovacího řetězce). Tento první testovací vektor je odsimulován simulátorem poruch, detekované poruchy jsou odstraněny z UFL a jim příslušné testovací vektory jsou odstraněny z TPL.

Dále se algoritmus snaží zkomprimovat testovací data postupným překrýváním zbývajících testovacích vektorů s aktuálním stavem skenovacího řetězce. Algoritmus nejprve deterministicky hledá, jestli je jako další bit, který se má nasunout do skenovacího řetězce, vhodnější logická 1 nebo logická 0. Při hledání nového bitu je kontrolováno, jestli není kvůli dříve zvoleným testovacím vektorům nutné nastavit některé budoucí bity. Tyto bity jsou ukládány v poli budoucích hodnot FA (Future Array) společně s ohodnocením jejich užitečnosti a identifikačním číslem testovacího vektoru.

Pokud je v FA některá pozice rezervována pro konkrétní logickou hodnotu, je tato okamžitě použita jako výsledek a není nutné jí vypočítávat prohledáváním všech zbývajících testovacích vektorů. Pokud není v FA rezervován žádný bit, je nutné maximálně překrýt všechny zbývající testovací vektory s aktuálním stavem skenovacího řetězce. Z takto překrytých vektorů pak vyhledat všechny takové, které na aktuální pozici bitu, který by měl být zvolen jako nový bit, mají specifikovaný bit.

Poté co jsou vhodné vektory vybrány, jsou všechny ohodnoceny kritériem užitečnosti U, kde menší hodnota znamená lepší ohodnocení. Po ohodnocení testovacích vektorů je následně spočítán počet vektorů s nejlepší (tedy nejmenší) hodnotou U, které nabízí jako aktuální řešení log. 1 a počet vektorů, které nabízí log. 0. Pokud je počet vektorů s log. 1 větší než počet vektorů s log. 0, je jako řešení zvolena jednička. Pokud je menší, je zvolena nula. Tento způsob výběru řešení garantuje, že je vybráno a zakódováno největší množství nejužitečnějších testovacích vektorů. Výsledný bit a všechny ostatní specifikované bity náležící vybrané skupině vektorů jsou zapsány do pole FA obsahujícího budoucí řešení, takže následující vyhledávání bitů může být výrazně urychleno.

Pokud je počet překrytých vektorů se specifikovaným bitem na aktuální pozici nulový, nebo je jich shodný počet pro log. 1 i pro log. 0, není možné přímo rozhodnout, jaká hodnota by měla být zvolena jako výsledek. Jako výsledek je proto zvolena hodnota

generovaná pseudonáhodným váženým generátorem kontrolovaným počtem jedniček a nul ve zbývajících dosud nezkomprimovaných testovacích vektorech.

Po zvolení výsledného bitu je tento nasunut do skenovacího řetězce, přičemž všechny stávající bity řetězce jsou posunuty o jednu pozici. Poté je spuštěna poruchová simulace. Detekované poruchy a jim příslušné testovací vektory jsou vymazány se seznamů TPL a UFL. Pokud již v UFL nebývá žádná nedetekovaná porucha, algoritmus končí.

4. Rozbor nového algoritmu

Všechny části nového algoritmu popsané v následujících kapitolách jsem samostatně realizoval a experimentálně ověřil. Při návrhu metod v kapitolách 4.1, 4.2 a 4.3 částečně navazuji na výzkum ostatních členů týmu, především [33] a [49]. Nově navržené metody jsou ale kompletně přepracovány především z hlediska použitelnosti s rozsáhlými obvody (více než 200 tisíc hradel). Pro testování jsou použity vybrané obvody testovací sady ISCAS85 [12], ISCAS89 [13] a ITC99 [14] ve velikostech od šesti do přibližně 225 tisíc hradel.

4.1. Kritérium ohodnocení

Určuje vhodnost každého nekomprimovaného testovacího vektoru k nasazení do výsledné komprimované posloupnosti v aktuálním kroku kompresního algoritmu. Kritérium je vyjádřeno celým číslem větším než nula a je zvoleno tak, že menší hodnota značí užitečnější vektor. Užitečnost testovacího vektoru U je spočítána jako:

$$U = t * (overlapped\_dontcares + shift) + all\_dontcares$$

Kde *overlapped_dontcares* značí počet nespecifikovaných bitů v překryté části testovacího vektoru, *shift* počet nepřekrytých bitů vektoru, *all_dontcares* celkový počet nespecifikovaných bitů v testovacím vektoru a *t* je uživatelsky volitelný parametr. Parametr *t* hraje roli váhy důležitosti specifikovaných bitů v celém vektoru proti specifikovaným bitům v překryté části vektoru. Vhodnou volbou parametru *t* je možné dosáhnout vyšší komprese. Zpracováním všech tří sad testovacích obvodů bylo zjištěno, že jako vhodná hodnota *t* je počet vstupů obvodu dělený dvěma. Parametr *t* je možné v případě nutnosti volit z příkazové řádky. Obvyklé ale je, že jeho hodnota není zadána, a program pak sám spočítá vstupy obvodu a nastaví správně jeho hodnotu.

4.2. Predikce bitů komprimované posloupnosti

Metoda spočívá v předpovědi budoucího řešení na základě aktuálního ohodnocení užitečnosti vektoru, tedy myšlence, že je-li vektor nejvhodnější v současném kroku, bude pravděpodobně nejvhodnějším i v krocích následujících. Pro lepší předpověď je bráno v úvahu větší množství vektorů a ne jen ten nejlepší.

Tím, že jsou do výsledného řešení stabilně prosazovány hodnoty nejlepších vektorů, dochází ke zlepšení kompresního poměru. Vedlejším důsledkem zkrácení délky komprimované posloupnosti je, že program běží rychleji, protože ve svém průběhu spouští méněkrát simulátor poruch, a musí vyhledat méně bitů. Zrychlení ale nastává i jako přímý důsledek předpovídání bitů, protože v předchozích krocích předpovězený bit je využit bez výpočetně náročného překrývání všech zbývajících vektorů.

Je-li v aktuální pozici pole s budoucími bity specifikovaný bit, je tento ihned zvolen jako platné řešení. Protože byl bit zařazen v poli budoucích řešení, je jisté, že nebude kolidovat s ostatními vektory. Je také jisté, že zvolit tento bit je nejvýhodnější, protože bity jsou ukládány podle kritéria ohodnocení. Tím, že je možné okamžitě zvolit řešení, jsou kompenzovány časové ztráty související s údržbou datových struktur. Klasické překrývání všech vektorů s aktuálním skenovacím řetězcem je totiž časově mnohem náročnější.

Pokud není přítomen žádný předpovězený bit, dojde k ohodnocení všech zbývajících testovacích vektorů pomocí kritéria. Všechny vektory jsou tedy co nejvíce překryty s aktuálním stavem skenovacího řetězce a následně je vypočítáno jejich kritérium užitečnosti. Seznam vektorů je pak předán k zařazení do pole případných budoucích řešení. Vektory jsou zařazovány postupně od nejlepšího po nejhorší, přičemž je kontrolována jeho použitelnost a případné kolize s jinými vektory.

4.3. Náhodné bity

Při hledání dalšího komprimovaného bitu může nastat případ, že řešení nebude nalezeno v poli budoucích řešení. Pokud není možné získat řešení ani pomocí překrývání všech vektorů, je nutné zvolit bit náhodně, protože do skenovacího řetězce nesmí vstoupit nespecifikovaná hodnota.

Zcela náhodná volba bitu ale není vhodná, protože pro každý běh programu mohou vyjít různé výsledky. Nelze pak rozumně porovnávat kvalitu komprese ani její dobu. To lze vyřešit např. tak, že se za náhodnou hodnotu bude dosazovat vždy log. 0 nebo log. 1, případně poslední použitá hodnota. Ani jedna z těchto metod ale nebere přímo ohled na komprimovaná data.

Proto byla pro výběr pseudonáhodného bitu navržena metoda, která spočítá poměr všech log. 0 vůči všem log. 1 ve zbývajících, dosud nezkomprimovaných datech. Pokud je ve zbývajících datech více log. 1 a přijde požadavek na pseudonáhodnou hodnotu, je vrácena log. 1 (a naopak). Protože jsou tak do skenovacícho řetězce umisťovány hodnoty, u kterých je větší pravděpodobnost výskytu, teoreticky může dojít ke zlepšení komprese. Poměr zbývajících jedniček a nul je v průběhu programu přepočítáván tak, aby odpovídal aktuálním datům.

4.4. Interní simulátor

Výpočet komprimované testovací posloupnosti fungoval, ale trval velmi dlouho (např. data obvodu s15850 přes 76 hodin). Velkou nevýhodou byla zdlouhavá příprava dat pro externí simulátor poruch, jeho spuštění pro simulaci jediného testovacího vektoru, a následná analýza výstupních dat. Tento nedostatek je obtížně odstranitelný, neboť kvalitní simulátor poruch je velice obtížné získat se zdrojovými kódy a souhlasem autora k dalšímu použití. Vestavěním simulátoru poruch do systému COMPAS na úrovni zdrojových kódů je však možné dosáhnout značné časové úspory.

Jako vestavěný simulátor poruch byl nejprve zvolen simulátor FSIM [11] z balíku ATALANTA (trvalé poruchy, paralelní simulace testovacích vektorů), spouštěný po vygenerování každého jednotlivého bitu komprimované testovací posloupnosti. Simulátor je plně začleněn do systému COMPAS, je tedy součástí stejného spustitelného souboru.

Vestavěním simulátoru poruch byla zbytečná režie v podobě opakovaného zápisu, čtení a spouštění dalšího procesu odstraněna, navíc simulace probíhá v každém kroku, nedochází proto k degradaci kompresního poměru, protože jsou poruchy detekovány bez zpoždění několika kroků. Spojení COMPASu s FSIMem je na úrovni vnitřních datových struktur a funkcí, algoritmy navzájem využívají přímo interní datové struktury druhé části, obě části jsou v jediném spustitelném souboru.

Tím, že se kompletně odstranil jakýkoli zápis do souborů, neustálé opakované načítání popisu obvodu, připravování struktury obvodu a vytváření a ukončování procesu simulátoru, bylo dosaženo značného urychlení v souvislosti se simulací poruch.

4.5. Běh bez simulátoru poruch

Běh programu bez simulátoru poruch může být vynucen z příkazové řádky. Tato vlastnost je užitečná pro kompresi libovolných obecných vektorů, vektory mohou být vytvořeny i pro testování jiných než trvalých poruch. Simulátor poruch také není možné použít, pokud není k dispozici popis obvodu. Algoritmus se následně snaží maximálně komprimovat posloupnost prostým vzájemným překrýváním vektorů. Protože ale není možné využít úzké spolupráce se simulátorem poruch, je dosaženo horší komprese. Komprese bez použití simulátoru poruch má ale i výhodu: každý nekomprimovaný vektor je zcela jistě přítomen ve výsledném komprimovaném řetězci. Pokud není použit simulátor poruch, kompresní algoritmus je bezeztrátový jak z hlediska detekovaných poruch, tak z pohledu testovacích vektorů obsažených v komprimované sekvenci..

Tabulka 1: Příklad komprese dat pro IDDQ test ukazuje výsledky komprese dat pro IDDQ test. Data byla vygenerována pomocí ATPG MILEF [41]. Protože COMPAS nemá k dispozici simulátor pro IDDQ model poruch, byl spuštěn bez použití simulátoru,

a bylo tak využito prosté překrývání nekomprimovaných testovacích vektorů. Přestože není simulace poruch použita, dosahuje komprese velmi dobrých výsledků.

Tabulka 1: Příklad komprese dat pro IDDQ test

Obvod	Množství spec. bitů	Komprimovaná délka [b]	Čas komprese [s]
s15850	0,81%	2392	1,3
s38417	0,25%	3873	13,4
s35932	0,20%	1831	6,9
s38584	0,27%	7175	18,8

4.6. Množství dat u velkých obvodů

Do operační paměti byla ukládána nekomprimovaná data. To se ukázalo jako problém, protože např. pro největší obvod z testovací sady ITC99 je vygenerováno přes 6 GB testovacích dat. V práci [34] bylo zjištěno, že využití více testovacích vektorů významně zlepšuje kompresi, stejný předpoklad je možné učinit i u COMPASu. Odhad objemu nekomprimovaných testovacích dat ovšem vychází pro větší obvody a při použití třeba jen deseti různých vektorů na jednu poruchu v desítkách gigabytů. Pro optimální rozhodování kompresního algoritmu je nutné načíst všechna data najednou do operační paměti počítače, proto je nutné pro ukládání dat do paměti vyvinout novou metodu.

První způsob zakódování je lepší zakódování ternární abecedy tvořené písmeny ‘01X’ tak, že každý znak není uložen v osmi bitech ale jen ve dvou, tedy přímý převod osmibitového znaku na dva bity. Dva bity umožňují zakódovat čtyři různé kombinace, tři jsou využity pro zakódování znaků ‘01X’, čtvrtá možnost je nevyužitá. Tímto kódováním, nazývejme ho ternární kódování, je každý znak, který původně zabíral 8 bitů převeden na 2 bity, je tedy ušetřeno 6 bitů (tj. 75% paměti) bez ohledu na vstupní data.

Druhý způsob kódování vytváří řídké vektory, což znamená, že do paměti jsou ukládány pouze specifikované bity a jejich pozice. Jejich použití je možné vzhledem k množství nespecifikovaných bitů v testovacích datech. Nespecifikované bity nejsou při využití řídkých vektorů vůbec do paměti ukládány. Jako základní datový typ prvku řídkého vektoru je možné použít libovolný celočíselný datový typ. Z tohoto základního typu je pak vyhrazen jeden bit pro uchování skutečné hodnoty bitu a zbytek je použit pro uložení pozice bitu v původním nekomprimovaném testovacím vektoru. Jako základní datový typ byl s ohledem na rozsah zvolen *short int*, protože poskytuje nejmenší rozsah dostatečný i pro větší obvody. Nepředpokládá se tedy obvod se skenovacím řetězcem delším než 32767 buněk. Pokud by takový případ nastal, je možné základní datový typ jednoduše změnit na *int*, který umožňuje skenovací řetězec o délce přes dvě miliardy buněk.

Protože efektivita komprese u řídkých vektorů je závislá na načtených datech, je způsob kódování dynamicky zvolen tak, aby bylo množství spotřebované paměti minimální. Po analýze dat je možné zvolit kódování jednotně buď pro celou testovací sadu vektorů, nebo i pro každý vektor zvlášť. Zvolené metody kódování garantují minimálně 75% úsporu paměti. Pro větší obvody, které mají v testovacích datech více nespecifikovaných bitů, úspora dosahuje běžně přes 95%.

4.7. Urychlení vynecháváním výpočtů

Rychlost výpočtu je možné významně urychlit vynecháním některých výpočetních kroků, je-li zřejmé, že jejich vykonání není přínosné. Protože lze nespecifikované bity překrýt s libovolnou hodnoutou, je potřeba zjistit kolik následujících taktů je ve vektoru nespecifikovaný bit. Tento výpočet proběhne pro každý nově zjištěný výskyt série nespecifikovaných bitů ve vektoru jen jednou, zároveň je rychlejší, než zjištění překrytí s už zkomprimovanou posloupností. Při každém vyhodnocení překrytí je tedy zároveň zjištěno, zda následuje série nespecifikovaných bitů, a jak je dlouhá. Pokud následuje alespoň jeden nespecifikovaný bit, je vyhodnocení překrytí vektoru v dalším běhu programu na patřičný počet kroků vynecháno.

Nekomprimovaná testovací data obsahují značné množství nespecifikovaných bitů, navíc se poměr zlepšuje pro velké obvody. Vynecháním výpočtu překrytí při výskytu nespecifikovaného bitu je tedy možné ušetřit velké množství výpočtů, pro větší obvody je rychlost komprese více než dvojnásobná.

4.8. Závislost na použitém generátoru poruch

Systém COMPAS byl původně vyvíjen pro použití s programovým balíkem ATALANTA. Kompresní algoritmus byl opakovaně laděn na stejných vstupních datech, vzniká tedy otázka na jeho chování v případě použití jiného ATPG.

Hypotéza: Při dané struktuře vstupních dat pro COMPAS je možné použít libovolný ATPG, aniž by byl výsledný kompresní poměr výrazně ovlivněn.

K ověření vyslovené hypotézy byl systém COMPAS plnohodnotně rozšířen o podporu nástrojů z balíku TurboTester [40] vyvíjeného na Technické univerzitě v Tallinnu, Estonsko. Jedná se o komplexní sadu programů zaměřených na testování obvodů pomocí SSBDD (strukturně syntetizované binární rozhodovací diagramy) [39]. Pro generování deterministických testů slouží ATPG Generate, jako simulátor poruch byl vybrán SPSimul. Pro ověření hypotézy byl také použit upravený ATPG MILEF. Simulátor poruch z programu MILEF nebyl do COMPASu začleněn, namísto toho byl používán už vestavěný simulátor FSIM.

COMPAS byl rozšířen o podporu softwarového balíku TurboTester, jako součást COMPASu byl implementován další simulátor poruch SPSimul. Účinnost algoritmu byla ověřena použitím třech různých ATPG. Přestože kompresní algoritmus nebyl pro

data generovaná ATPG Generate nebo MILEF nijak upravován, ani volitelný parametr *t* nebyl nijak měněn, vykazuje COMPAS velmi dobrou účinnost, srovnatelnou s předchozími výsledky získanými kompresí dat z ATPG ATALANTA. Všechny tři ATPG přitom používají jiný algoritmus pro generování testovacích vektorů. Je tedy možné prohlásit, že kompresní algoritmus je možné úspěšně použít s libovolným ATPG schopným generovat vektory s nespecifikovanými bity. Hypotéza ze začátku kapitoly je tedy platná.

Pro ověření hypotézy byly prozatím použity pouze ATPG z akademické sféry, dalším krokem tedy bude využití komerčních generátorů testů. Tím bude tento směr výzkumu pravděpodobně uzavřen.

4.9. Obtížně testovatelné poruchy

Dalšího zlepšení kompresního poměru je možné dosáhnout změnou pořadí překrývaných vzorků, jejich lepším řazením. Je výhodné na začátku zakódovat testovací vzorky, které detekují obtížně detekovatelné poruchy (poruchy odolné proti náhodnému testu), protože poruchy lehce testovatelné náhodnými vzorky budou pravděpodobně detekovány v mezilehlých krocích algoritmu.

Lepší volba vektoru je umožněna zavedením hodnoty *obtížnost vektoru*, kde vektor s nižší hodnotou obtížnosti detekuje méně lehce detekovatelných poruch a více obtížně detekovatelných. Vektor s nízkou hodnotou obtížnosti by měl být umístěn v komprimované posloupnosti co nejdříve.

Použitím hodnoty obtížnosti vektoru můžeme definovat nové kritérium ohodnocení užitečnosti vektoru:

$$U = t * (overlapped\_cares + shift) + all\_cares + hard\_patt$$

Kde *hard_patt* značí hodnotu obtížnosti vektoru a ostatní parametry mají stejný význam jako v původním kritériu. Když algoritmus využije tento nový vzorec, všechny testovací vektory budou ohodnoceny jinak než v předchozí verzi. Odlišné hodnoty užitečnosti pak vedou k jinému pořadí výběru vektorů a ke zlepšení komprese. Průměrné zvýšení kompresního poměru je přibližně deset procent

4.10. Optimalizace pro víceprocesorové systémy

Jako úseky vhodné pro spuštění ve více vláknech byly vybrány především časově nejnáročnější části kódu, tedy překrývání vektorů, zařazování vektorů do seznamu budoucích řešení a řadicí algoritmus, který je často používán v různých částech algoritmu.

Překrývání vektorů je přirozeným kandidátem na paralelizaci. K zjištění překrytí jednoho každého vektoru se skenovacím řetězcem dochází zcela nezávisle na ostatních

vektorech. Nezávisle je vypočteno i ohodnocení vektoru. Při použití dvou procesorů jsou tedy vektory rozděleny na dvě skupiny, kde každá je nezávisle ohodnocena.

Po skončení překrývání a ohodnocení jsou vektory seřazeny podle hodnoty kritéria. Jako řadicí algoritmus byl implementován paralelní mergesort, protože se dobře paralelizuje, je to přirozený a stabilní řadicí algoritmus a má nízkou složitost $O(N \log N)$.

Nakonec jsou seřazené vektory zařazovány do pole budoucích řešení, což je časově poměrně náročné. Protože je to ale poslední krok před konečným výběrem aktuálního komprimovaného bity a simulací poruch, mohou už být prováděny některé údržbové práce na pomocných datových strukturách. V tomto případě jsou tedy v každém programovém vlákně prováděny rozdílné činnosti.

Paralelizace časově náročných částí algoritmu bohužel nepřinesla zrychlení programu (výsledky byly přibližně shodné), přestože vytížení procesorů v systému vzrostlo z 50% na přibližně 80%. Důvodem může být příliš málo paralelní práce a příliš velká režie vícevláknového běhu. Množství paralelní práce je totiž účinně snižováno vynecháváním výpočtů. Velkou část doby komprese navíc tvoří simulace, proto se zrychlení vyhledávání komprimovaných bitů projeví poměrně málo.

4.11. Paralelní skenovací řetězce

Pokud je použito více skenovacích řetězců, je vhodné použít jako dekompresní hardware RESPIN architekturu. Použití této architektury vyžaduje dodržení několika podmínek. Jejich rozbor a zdůvodnění je možné nalézt např. v [52]. Pro kompresi je důležité, jak návrhář zapojil skenovací řetězce, které skenovací buňky jsou v jakém řetězci, kolik řetězců je použito, jaké je propojení s ETC jádrem apod. Všechny tyto informace ovlivňují pořadí bitů v testovacích vektorech. Pokud jsou ale dodrženy všechny podmínky použití RESPIN architektury, projeví se všechny varianty zapojení jen jako změna pořadí vstupů obvodu.

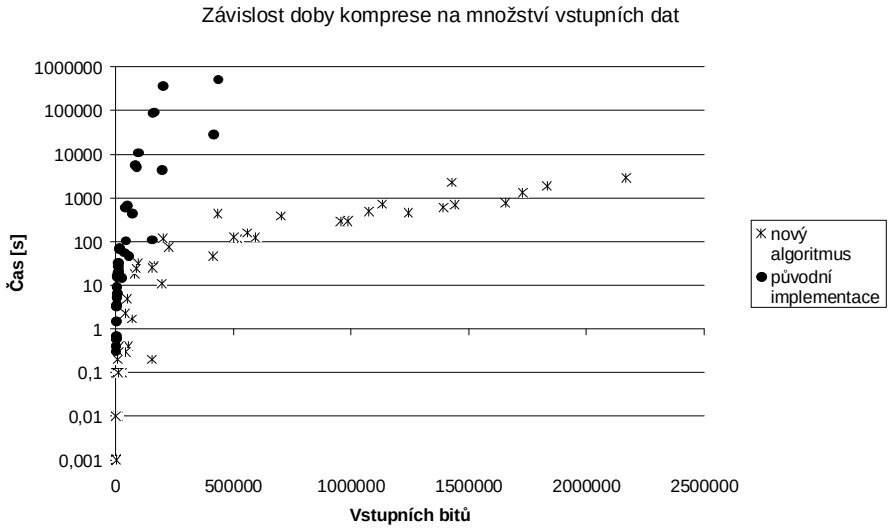
Je důležité upozornit, že tato práce se nezbývá vlivem zapojení testovací architektury na pořadí vstupů obvodu. Testovací hardware navrhuje podle různých potřeb návrhář, je tedy jeho úkolem dodat popis pořadí vstupů, jaké si zvolil. Popis pořadí vstupů je COMPASu předán jako jednoduchý textový soubor s řadou čísel, která postupně určuje pořadí jednotlivých vstupů obvodu. COMPAS soubor načte a při postupném načítání nekomprimovaných testovacích dat ihned provede změnu pořadí bitů.

Komprese dat s prohozeným pořadím vstupů pak probíhá stejně jako komprese bez prohození, s výjimkou simulace poruch. Před spuštěním simulace je nutné vždy změnit pořadí bitů ve skenovacím řetězci tak, aby odpovídalo pořadí vyžadovanému návrhářem obvodu. Po dokončení simulace poruch je obsah skenovacího řetězce obnoven do původního stavu a komprese pokračuje běžným způsobem.

5. Shrnutí a závěr

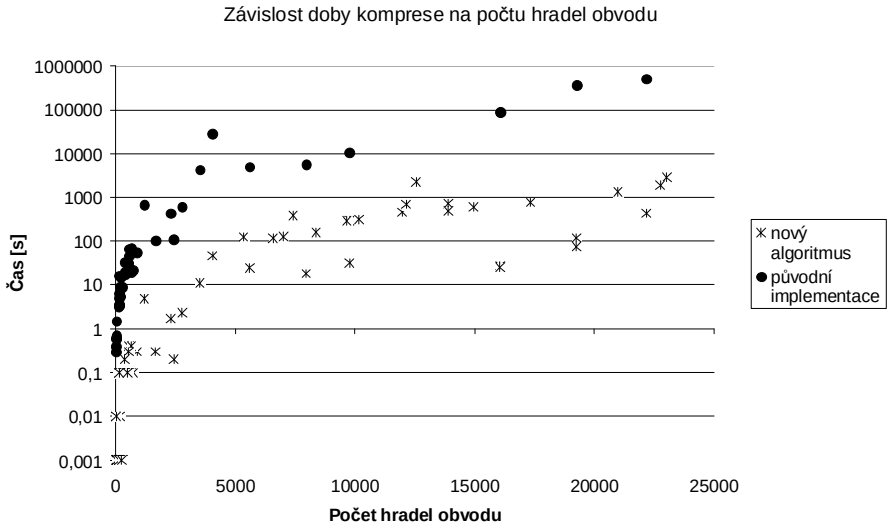
5.1. Srovnání s původní verzí algoritmu

Ve srovnání s původní verzí algoritmu byly výrazně sníženy paměťové nároky programu. Zvolená metoda garantuje minimálně 75% úsporu paměti. Pro větší obvody, které mají v testovacích datech více nespecifikovaných bitů, úspora dosahuje běžně přes 95%. Pomocí využití informací o obtížně testovatelných poruchách byla průměrně o 10% zlepšena komprese. Nový algoritmus má výrazně lepší škálovatelnost (viz Graf 1).



Graf 1: Závislost doby komprese na množství vstupních dat

Algoritmus je také mnohem rychlejší (viz Graf 2), v některých případech až o čtyři řády.



Graf 2: Závislost doby komprese na počtu hradel obvodu

5.2. Srovnání s metodami pro kompresi testovacích dat

Tabulka 2: Porovnání výsledků komprese dat různých metod

	MinTest	Stat. Coding	LFSR Reseeding	Illinois Scan	FDR Codes	EDT	RESPIN++	COMPAS	
1	2	3	4	5	6	7	8	9	10
Obvod	bitů	bitů	bitů	bitů	bitů	bitů	bitů	bitů	čas [s]
s13207	163,100	52,741	11,285	109,772	30,880	10,585	26,004	3,819	8
s15850	58,656	49,163	12,438	32,758	26,000	9,805	32,226	6,930	11
s38417	113,152	172,216	34,767	96,269	93,466	31,458	89,132	19,597	177
s38584	161,040	128,046	29,397	96,056	77,812	18,568	63,232	5,778	57

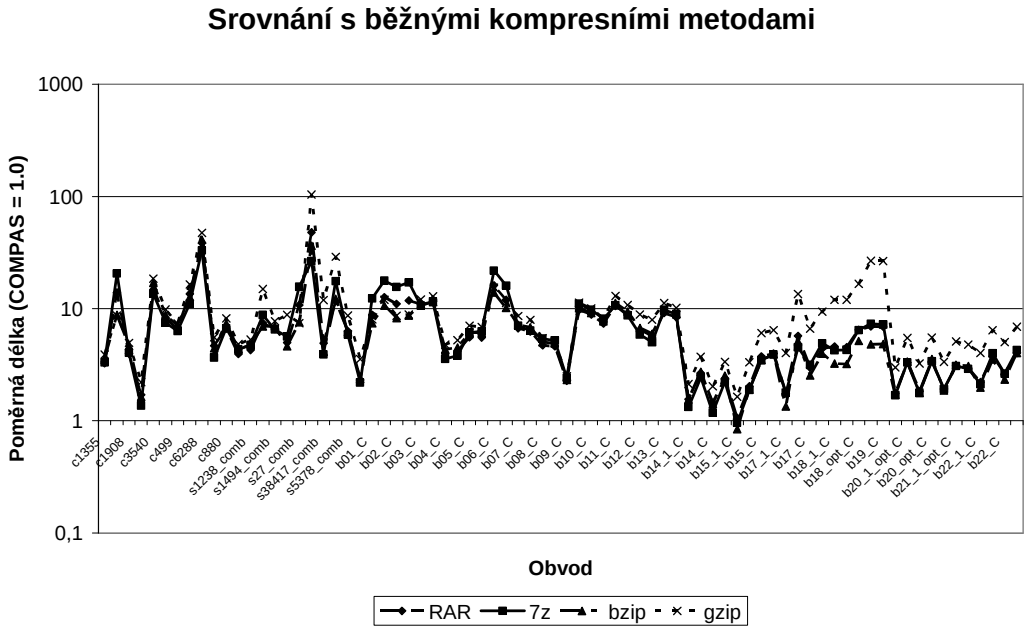
Tabulka 2 obsahuje výsledky některých dalších metod komprese testovacích vektorů ve srovnání s výsledky COMPASu, pokud použijeme všechna dostupná vylepšení. (Měřeno na procesoru Intel Core2Duo 2,4 GHz.) Výsledky jsou zobrazené jen pro čtyři obvody, protože pouze tyto byly zkoumány i ostatními metodami. V druhém sloupci je zobrazeno množství testovacích dat pro ATPG vektory, které prošly pouze kompakcí [44]. Sloupec 3 ukazuje počet uložených bitů pro statistické kódování testovacích vzorků z předchozího sloupce [45]. Sloupec 4 obsahuje výsledky statistického kódování společně s znovunastavením LFSR [29]. Sloupec 5 ukazuje výsledky komprese s paralelními/sériovými skenovacími řetězci [46], sloupec 6 výsledky frekvenčně řízeného kódování [28]. Výsledky metody Embedded Deterministic Test [47] jsou prezentovány ve sloupci 7. Sloupec 8 ukazuje množství bitů uložených v ATE pro architekturu RESPIN++ prezentovanou v [31]. Je zřejmé, že počet v ATE uložených bitů je pro metodu COMPAS (sloupec 9) výrazně nižší, než pro ostatní kompresní metody.

Je nutné poznamenat, že většina zmíněných metod komprese testovacích dat nepoužívá simulaci poruch při kódovaní jednotlivých testovacích vektorů (s výjimkou metody RESPIN++). Tyto metody ale používají kompaktní sadu testovacích vektorů, pokrytí poruch bylo proto simulátorem poruch zjišťováno v průběhu generování testu a kompakce pomocí ATPG. Počet simulací poruch je v těchto případech roven celkovému počtu nekompaktních testovacích vektorů. V případě RESPIN++ architektury a COMPASu jsou ATPG vektory generovány bez simulace poruch, protože jsou poruchy simulovány v průběhu komprese po každém vygenerování bitu komprimované testovací posloupnosti. Celkový počet spuštění simulátoru poruch je tak rovný výsledné délce komprimované testovací posloupnosti v bitech. Přestože je počet simulací poruch velký, celkový výpočetní čas algoritmu zůstává poměrně nízký. Obtížnost vzájemného srovnání výsledků komprese také způsobuje využití pseudonáhodných vektorů pro některé kompresní metody. I přes obtížné možnosti srovnání různých kompresních metod lze konstatovat, že COMPAS podává nejlepší výsledky.

5.3. Srovnání s běžnými kompresními metodami

COMPAS dosahuje výrazně lepšího kompresního poměru než běžné kompresní programy jako gzip, bzip2, RAR nebo 7z, viz Graf 3. Vynesen je poměr dosažených délek komprimovaných posloupností jednotlivých programů oproti COMPASu, který je brán jako základ, tedy 1,0. COMPAS je výrazně lepší pro drtivou většinu z 77 testovaných obvodů (v případě srovnání s gzip programem a obvodem s35932 dokonce více než 100 krát). Jedinou výjimkou je varianta obvodu b15 z testovací sady ITC99, kdy je COMPAS překonán programy bzip2 a 7z.

Velkou výhodou algoritmu COMPAS oproti běžným metodám je použití simulátoru poruch při kompresi. Tím, že je každý krok algoritmu a tím i každý vygenerovaný vektor simulován, dochází k rychlé detekci poruch testovatelných náhodným testem. Jakmile je libovolná porucha detekována, je jí odpovídající vektor odstraněn. Přitom nezáleží, jaké je rozložení bitů ve vektoru, vektor může být ze statistického hlediska velmi odlišný oproti aktuálnímu stavu skenovacího řetězce, ale protože je schopen detekovat stejnou poruchu, je redundantní a jako takový je odstraněn. Algoritmus tedy šetří místo nejen statistickou metodou, kdy vyhledává největší podobnost mezi jednotlivými testovacími vektory, ale i využitím informací o vnitřní struktuře obvodu.



Graf 3: Srovnání délky komprimované posloupnosti oproti běžným kompresním metodám

Výsledný komprimovaný řetězec spočítaný COMPASem by mohl být dále komprimován nějakou statistickou metodou, např. Huffmanovým kódem nebo aritmetickým kódem. To by ale vynucovalo použití značně komplikovaného a rozsáhlého dekompresního hardware uvnitř testovaného obvodu. Proto není žádné takové kódování použito.

5.4. Závěr

Systém pro kompresi testovacích dat COMPAS ukazuje, že metodu založenou na překrývání testovacích vektorů je možné úspěšně použít i na relativně velké obvody a že výsledný objem testovacích dat je velmi nízký. Dosažené výsledky ukazují, že algoritmus použitý v COMPASu může být dále vylepšen zjištěním dodatečných informací o vstupních datech.

Jako vstupní data používá COMPAS nekomprimované testovací vektory s nespecifikovanými bity. Vstupní data mohou být generována libovolným ATPG, který je schopen vytvořit dvojici porucha a vektor s nespecifikovanými bity. Protože algoritmus pracuje s páry tvořenými poruchou a jí příslušným vektorem, je COMPAS schopný komprimovat data vytvořená několika současně běžícími ATPG procesy, což může významně urychlit celý proces přípravy testu. Data generovaná ATPG jsou předzpracována a je určena možnost jednotlivých testovacích vektorů detekovat obtížně detekovatelné poruchy.

Na úrovni zdrojových kódů byly integrovány dva různé simulátory poruch, tato úprava vedla k více než stonásobnému urychlení. Urychlením komprese a efektivnějším kódováním nekomprimovaných testovacích dat v operační paměti počítače byly výrazně sníženy nároky systému COMPAS a otevřela se tak cesta pro další vylepšování algoritmu z hlediska lepšího kompresního poměru, především pomocí využití více testovacích vektorů pro jednu poruchu. Běh programu byl výrazně urychlen vynecháním opakovaných výpočtů a byla zlepšena škálovatelnost kompresního algoritmu. Byla ověřena nezávislost algoritmu COMPAS na použitém generátoru testovacích vektorů a použitém poruch. Byly úspěšně provedeny základní experimenty s paralelním zpracováním dat.

Všechny komprimované testovací vektory jsou v průběhu komprese simulovány simulátorem poruch zda nepokrývají dodatečné poruchy. Tento mechanismus redukuje počet výsledných vektorů, protože mezilehlé stavy vzniklé při posouvání dat ve skenovacím řetězci jsou používány pro detekci náhodně testovatelných poruch. Tyto poruchy jsou v jiných metodách obvykle detekovány náhodným testem a vedou na mixed-mode test, v COMPASu použitá metoda tento obvykle časově i energeticky náročný krok eliminuje. Přestože algoritmus při kompresi používá simulaci mezilehlých stavů a šly by chápat jako pseudonáhodné, nelze ho přímo zařadit mezi metody smíšeného testu, protože deterministická a pseudonáhodná část se překrývají.

Zvolenou metodu komprese je možné velmi dobře použít v kombinaci s metodou Boundary Scan, protože pak pro dekompresi testovacích dat není vyžadován žádný dodatečný hardware. Může být použitá i pro sekvenční obvody s jedním či více skenovacími řetězci, pak je nutné použít dekompresního hardwaru, např. RESPIN architekturu. Testovací sekvence je pro použití s RESPIN architekturou a více skenovacími řetězci patřičně přeřazena tak, aby byla správně dekomprimována. Při

dodržení IEEE 1500 standardu není nutné přidávat žádný dodatečný hardware s výjimkou jednoho multiplexoru a jedné zpětné vazby pro každé jádro. Testovací sekvence generovaná COMPASem může být použita pro méně časově náročný test sekvenčních obvodů než pro test dosažitelný s mixed-mode testovacími metodami.

Systém COMPAS vytváří vysoce komprimované testovací vektory, dle dostupných materiálů je kompresní poměr lepší než pro ostatní srovnatelné metody a pro test je tak potřebná menší paměť, než pro srovnatelné metody. Některé kompresní metody sice mohou dosáhnout ještě vyšší komprese, potřebují ale velké množství pseudonáhodných testovacích vektorů. To výrazně zvyšuje čas komprese a energetickou náročnost testu. Použití algoritmu COMPAS vyžaduje pro vykonání testu nižší počet hodinových cyklů, test obvodu je proto rychlejší a levnější než při použití jiných kompresních metod.

Hardwarové nároky jsou tvořeny dekompresním hardwarem a pamětí nutnou pro uložení komprimované posloupnosti. Při použití architektury RESPIN není potřeba žádný speciální hardware, dekomprese testovacích vzorků probíhá ve skenovacích řetězcích jiných jader. Hardwarové nároky tedy tvoří jen nutná paměť.

Shrnutím výsledků je, že navržená kompresní a testovací metoda dosahuje při kompletním hodnocení (náročnost návrhu, velikost nutného hardware, rychlost a energetická náročnost testu) výrazně lepších výsledků než ostatní kompresní metody.

Shrnutí přínosů k rozvoji vědního oboru

V práci je popsán nový algoritmus komprese testovacích dat, který využívá metody překrývání testovacích vektorů. Algoritmus je podrobně popsán a je uveden přínos jednotlivých částí z hlediska vylepšení kompresního poměru, snížení doby komprese a snížení paměťových nároků.

Výsledky kompresní metody ukazují, že pro dosažení výborných výsledků není nutné kompletní prohledávání stavového prostoru, ale postačuje hladový algoritmus. Je prokázáno, že pro dosažení dobrého kompresního poměru je vhodné v průběhu komprese využívat simulátor poruch a brát v úvahu obtížně detekovatelné poruchy.

Shrnutí přínosů pro praxi

Všechny navržené metody byly otestovány na sadách ISCAS85, ISCAS89 a ITC99 s výbornými výsledky. Testované obvody tvoří dostatečně široké spektrum, aby bylo možné prohlásit, že COMPAS je dobře použitelný v praxi. Návrhářům obvodů je tak k dispozici rychlá, kvalitní a univerzálně použitelná kompresní metoda.

Budoucí práce a experimenty

Díky provedeným úpravám je možné uvažovat o použití více testovacích vektorů pro jednu poruchu, což pravděpodobně povede k dalšímu zlepšení komprese. Budoucí úlohou je také použití dalších ATPG, zvláště komerčních.

Algoritmus již částečně využívá paralelní zpracování dat, dalším úkolem bude dopracovat a vyladit program pro efektivnější využití více procesorů.

Seznam literatury

Vlastní publikace

[1] Jeníček, J.:Optimalizace systému pro kompresi testovacích vektorů COMPAS, PAD2005, ISBN 80-01-03298-1, pp.73-76

[2] Novák, O., Zahrádka, J., Holubec, M., Jeníček, J.:Test Set Compaction and Compression for circuits with Scan, Informal Digest of Papers of the IEEE European Test Symposium, Ajacio Corsica, France, 2004, pp.13-14

[3] Novák, O.; Plíva, Z.; Jeníček, J.; Mader, Z.; Jarkovský, M.:Self Testing SoC with Reduced Memory Requirements and Minimized Hardware Overhead, The 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'06), October 2006, Washington DC, USA

[4] Jeníček, J.: Optimalizace kompresního systému COMPAS. Proc. of PAD 2006, Papradno, SK, pp. 77-82, ISBN 80-969202-2-7

[5] Jeníček, J.: Použití algoritmu COMPAS s různými ATPG, Proc. of PAD 2007, pp. 27-32, ISBN 978-80-7043-605-9

[6] Jeníček J., Novák O.:A Test Pattern Compression Based on Pattern Overlapping, Proc. of DDECS 2007, Apr. 2007, Krakow, Poland, pp.29 - 34, ISBSN: 1-4244-1161-0

[7] Novák, O., Plíva, Z., Jeníček, J., Mader, Z., Jarkovský, M.: Self-Testing SoC with Reduced Memory Requirements and Minimized Hardware Overhead. Acta Electrotechnica et Informatica, Vol. 8, No. 1, pp. 22-32, 2008, ISSN 1335-8243

[8] Jeníček, J.: Efficient Test Pattern Compression Method Using Hard Fault Preferring. Proc. of DSD 2008, September 2008, Parma, Italy

[9] Jeníček, J.: Nový kompresní algoritmus pro systém COMPAS. Proc. of PAD 2008, September 2008, Hejnice, Czech, pp. 81-88, ISBN 978-80-7372-378-1

[10] Novák, O., Jeníček, J.: Test Pattern Overlapping - a Promising Compression Method for Narrow Test Access Mechanism SOC Circuits, Radioelectronics & Informatics, No. 1, pp. 26-33, 2008, ISSN 1563-0064

Použitá literatura

[11] ATALANTA, FSIM a HOPE http://www.ee.vt.edu./ha/cadtools

[12] ISCAS 85 Benchmark http://www.cbl.ncsu.edu/www/CBL_Docs/iscas85.html

[13] ISCAS 89 Benchmark http://www.cbl.ncsu.edu/www/CBL_Docs/iscas89.html

[14] ITC 99 Benchmark http://www.cerc.utexas.edu/itc-99benchmarks/bench.html

[15] Jaderný, J: Evoluční a genetické algoritmy, ročníkový projekt, TU Liberec 2004

[16] Novák, O., Nosek, J.: Test Pattern Decompression Using a Scan Chain, proc. of IEEE International Symposium on Defects and Fault Tolerance in VLSI Systems 2001

[17] Zahrádka, J., Holubec, M., Novák, O.: COMPAS – System for Finding of a Compress Test Pattern Sequence, proc. of ECMS Liberec 2003

[18] Novák, O., Zahrádka, J., Plíva, Z.: COMPAS - Compressed Test Pattern Sequencer for Scan Based Circuits. EDCC2005, Lecture Notes in Computer Science 3463, pp. 403-414, Springer-Verlag 2005, ISSN 0302-9743

[19] Plíva, Z., Novák, O., Siekierska, K., Grodner, M.: Test Access Circuit for Education. proceedings of DDECS2005, Sopron, Hungary, April 2005, pp. 27-32, ISBN 963-9364-48-7

[20] Jarkovský, M., Plíva, Z., Novák, O.: Software for Test_Access Circuit for Education. Proceedings ECMS 2005, Electronique, Cotrôle, Modélisation, Mesure et Signal, 17-20 May 2005. Toulouse: Université Paul Sabatier

[21] Jarkovský, M.: Uživatelské rozhranní a řídicí programové vybavení pro diagnostiku SoC obvodů s využitím RESPIN architektury. Počítačové architektury & diagnostika 2005

(PAD 2005), str.61-65, 21.- 23.9.2005, Lázně Sedmihorky. ISBN 80-01-03298-1
[22] Mader, Z.: Diagnostika SoC obvodů s využitím RESPIN architektury. In: Seminář PAD, Moravany nad Váhom, Slovak Republic, Sept. 2004, ISBN 80-969202-0-0, s. 66-71
[23] Tvrdík, P.: Paralelní systémy a algoritmy, Vydavatelství ČVUT, 2002, ISBN 80-01-02267-6, 226 s.
[24] Podrazký, M: Analýza možností výstavby clusterů v operačním prostředí Linux, diplomová práce, Vojenská akademie v Brně, 2003
[25] Geist, A. et al.: PVM: Parallel Virtual Machine – A Users' Guide and Tutorial for Networked Parallel Computing, The MIT Press, 1994, 279 s.
[26] Dorsch, R., Wunderlich, H.-J.: Reusing Scan Chains for Test Pattern Decompression, Proc. of the IEEE European Test Workshop 2001, pp. 24 – 32
[27] Chandra, A., Chakrabarty, K.: Test Data Compression for System-on-a-Chip Using Golomb Codes, Proc. of VTS Symp. 2000
[28] C Chandra, A., Chakrabarty, K.:Frequency/Directed Run Length (FDR) Codes with Application to System/on/Chip Test Sata Compression, Proc. of VLSI Test Symp., 2001, pp. 42-47
[29] Krishna, C.V., Touba, N.A.: Reducing Test Data Volume Using LFSR Reseeding with Seed Compression, Proc. of ITC 2002, pp. 321-330
[30] Hellebrand, S., Liang, H. G., Wunderlich, H. J.: A mixed mode BIST scheme based on reseeding of folding counters, Proc. of IEEE ITC, 2000
[31] Schafer, Dorsch, R., Wunderlich, H.J.: RESPIN++ - Deterministic Embedded Test, Proc. European Test Workshop, 2002, pp. 37-42
[32] ATALANTA-M http://service.felk.cvut.cz/vlsi/prj/Atalanta-M
[33] Holubec, M.: Komprese testovacích vektorů pro obvody se sériovým diagnostickým přístupem, diplomová práce, TU Liberec, 2004
[34] Fišer, P., Kubátová, H.: Multiple-Vector Column-Matching BIST Design Method, Proc. 9th IEEE Design and Diagnostics of Electronic Circuits and Systems 2006 (DDECS'06), Prague, CZ, 18.-21.4.2006, pp. 268-273
[35] OpenMP http://www.openmp.org/
[36] GCC - GNU Compiler Collection http://gcc.gnu.org/onlinedocs/gcc-4.1.1/gcc.pdf, 478 stran
[37] Fišer, P.: Mixed-Mode BIST Based on Column Matching, Počítačové Architektury & Diagnostika 2005, Lázně Sedmihorky, ČR, 21. – 23. 9. 2005, pp. 45-50
[38] Novák, O., Gramatová, E., Ubar, R. and coll.: Handbook of testing electronic systems. Nakladatelství ČVUT, srpen 2005, 395 stran, ISBN 80-01-03318-X
[39] A. Jutman, J. Raik, R. Ubar: SSBDDs: Advantageous Model and Efficient Algorithms for Digital Circuit Modeling, Simulation & Test, proc. of IWSBP'02, Freiberg, Germany, Sept. 19-20, 2002, pp. 157-166
[40] TurboTester http://www.pld.ttu.ee/tt/
[41] Gläser, U., Vierhaus, H. T. 1992. MILEF: an efficient approach to mixed level automatic test pattern generation. In Proc. of the Conference on European Design Automation, Los Alamitos, CA, pp. 318-321.
[42] L.H. Goldstein: Controllability/Observability Analysis of Digital Circuits, IEEE Trans. On Circuits and Systems, Vol. CAS-26, No. 9, pp. 685-693, September, 1979
[43] Marinissen, E. J., Zorian, Y., Kapur, R. Taylor T., and Whetsel. L.:Towards a Standard for Embedded Core Test: An Example. Proc. of ITC, pp. 616–627. IEEE, 1999.
[44] Bernhart et al.: OPMISR: the foundation for compressed ATPG vectors. Proc. ITC, 2001, pp. 748-757
[45] Abhijit Jas, Jayabrata Ghos-Dastir, and Nur A. Touba: Scan Vector Compression/Decompression Using Statistical Coding. Proc. VTS 1999
[46] Pandey, A. R., Patel, H. J.: Reconfiguration Technique for Reducing Test Time and Test Data Volume in Illinois Scan Architecture Based Designs. Proc. IEEE VLSI Test Symp, 2002, pp. 9-15

[47] Rajski, J. et al.: Embedded Deterministic Test. IEEE Trans. on CAD, vol. 23, No. 5, May 2004, pp. 776-792
[48] Mitra, S., Kim, K.S.: X-Compact: An Efficient Response Compaction for Test Cost Reduction, proc. of ITC 2002
[49] Jaderný, J.: Optimalizace parametru systému komprese diagnostických dat COMPAS, diplomová práce, TU Liberec 2005
[50] Jutman, A., Tsertov, A., Tsepurov, A., Aleksejev, I., Ubar, R., Wuttke, H.-D.: BIST analyzer: A training platform for SoC testing, proc. of FIE' 07, pp. S3H-8-S3H-13
[51] Vierhaus, H. T.: Introduction to IC test technology, Summer Academy „Design and Test Technology for Dependable Hardware / Software Systems", Cottbus, 2008
[52] Mader, Z.: Diagnostický systém jádrově založených SoC obvodů s nízkými nároky na paměť, disertační práce, TU Liberec, 2006
[53] Crouch, L. A.: Design for Test, Prentice Hall, 2000, ISBN 0-13-084827-1