GENEROVANIE TESTOV PRE PORUCHY ONESKORENÍ DIGITÁLNYCH OBVODOV PRE APLIKÁCIU CEZ TESTOVACIE OKOLIE

Marcel Baláž
Odbor: Aplikovaná informatika, V. ročník externého doktorandského štúdia
Školiteľ: doc. RNDr. Elena Gramatová, CSc.

Ústav informatiky, Slovenská akadémia vied
Dúbravská cesta 9, Bratislava, Slovensko

marcel.balaz@savba.sk

Abstract. Práca rozoberá problematiku aplikovania dvojvektorových testov pri testovaní jadier v obvodoch na čipe (SoC). Nová metodika generuje test pre poruchy prechodov aplikovateľný cez testovacie okolie (wrapper) za účelom žiadnej, prípadne minimálnej zmeny architektúry štandardného testovacieho okolia. Aplikovanie takéhoto testu nevyžaduje žiadne prídavné riadenie.

Keywords. System on chip, core test, test wrapper, delay fault, test generation, scan chain.

1 Úvod

Systémy integrované na čipe (*System on Chip* – SoC) vyžadujú nový prístup k aplikácii testov. Vnorené bloky v SoC potrebujú vhodné testovacie okolie (*wrapper*), aby bolo možné aplikovať testovacie vektory z externého zariadenia alebo zo vstavaného samočinného testera využiteľného pre testovanie viacerých jadier (*cores*). Medzi najpoužívanejšie architektúry testovacích okolí patria štandardy IEEE 1149.1 (JTAG) [2], IEEE 1500 (SECT – Standard for Embedded Core Test) [1]. Testovacie okolie jadra zabezpečuje prístup k jeho primárnym vstupom, výstupom a v mnohých prípadoch aj k interným registrom jadra [4]. Pomocou takejto architektúry je možné vnorené jadrá efektívne testovať pre rôzne modely porúch. Oba spomenuté štandardy umožňujú návrhárom zadefinovať si vlastné inštrukcie pre testovanie, a tým aj nové, špecifické typy testovacích postupov. Samotná architektúra je uspôsobená na aplikáciu testov nad modelom trvalých porúch a porúch, ktoré vyžadujú na ich pokrytie jeden testovací vektor. V prípade, že je potrebné otestovať poruchy spôsobujúce oneskorenie pri zmene logických hodnôt, nie je možné zaužívané štandardy využiť bez použitia zložitejších buniek scan s dvoma pamäťovými bunkami, prípadne iných modifikácií a to z dôvodu nutnosti aplikácie dvoch testovacích vektorov za sebou.

Súčasné a budúce technológie sú a budú veľmi náchylné na defekty, ktoré spôsobia zmenu v časových parametroch logických členov, častí alebo aj celého digitálneho obvodu. Pre takéto defekty boli definované viaceré modely porúch oneskorení, ktoré je nevyhnutné testovať na dosiahnutie kvalitnej výťažnosti z výroby digitálnych obvodov. Testy pre poruchy oneskorení sú zložené vždy z dvoch testovacích vektorov, ktoré musia byť aplikované na primárne vstupy v následných hodinových taktoch. Ak je ale digitálny obvod vnorený v SoC s prístupom k jeho vstupom a výstupom len cez testovacie okolie, je potrebné nájsť optimálny spôsob aplikácie testu. Touto problematikou sa napríklad zaoberá článok [5], kde je nevýhodu nárast riadenia aplikácie testu.

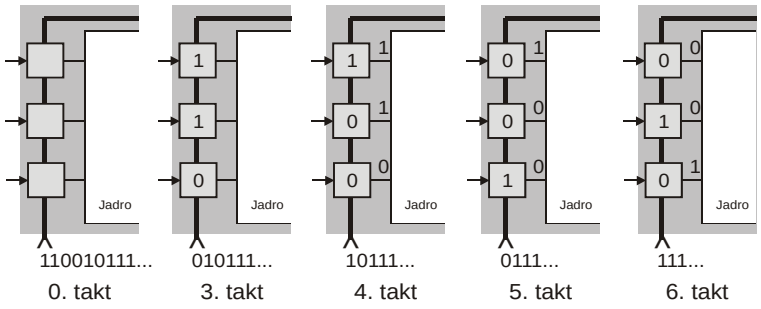
Táto práca opisuje metodiku, ktorá sa snaží využiť pôvodnú architektúru testovacieho okolia uvedených štandardov bez zásadnejšej zmeny na aplikáciu dvojvektorových testov.

Príspevok je rozdelený troch častí. V prvej sú uvedené dôvody a motivácia zaoberať sa generovaním testov pre poruchy oneskorení cielene na ich aplikáciu cez sekvenčné testovacie okolie. Ďalšia časť opisuje návrh metodiky generovania testov, teda postupnosti dvojíc testovacích vektorov na pokrytie porúch oneskorení. Tretia časť je zameraná na opis modifikácií architektúr testovacieho okolia na aplikáciu testov pre poruchy oneskorení, ktoré v porovnaní s tradičným testovaním majú menšiu pridanú plochu k čipu a test je optimálnej dĺžky. Navrhovaná metodika bola overená na obvodoch ISCAS'85 a v príspevku sú uvedené čiastkové výsledky.

2 Motivácia a ciele dizertačnej práce

Cieľom dizertačnej práce je prispieť k zvýšeniu kvality testovania digitálnych obvodov s testovacím okolím vnorených v SoC. Jedným bodom naplnenia tohto cieľu je zvýšenie kvality testovania z hľadiska dostatočného pokrytia porúch oneskorení a z hľadiska optimalizácie aplikácie testu (na časovej aj hardvérovej úrovni). Časť dizertačnej práce je návrh metodiky, ktorá by umožnila aplikáciu testov oneskorení pomocou štandardného testovacieho okolia bez zložitého riadenia a s minimálnou modifikáciou. Aplikácia testov je zameraná na štandardné sériové rozhranie. V každom takte sa posunie postupnosť bitov o jednu pozíciu, čím sa vytvorí nový testovací vektor, ktorý sa aplikuje v nasledujúcom takte. Priebeh aplikácie napr. dvojice testovacích vektorov (110, 100) na jadro s tromi vstupmi je ilustrovaný na obrázku 1. Načítavanie prvej vzorky sa vykoná na tri hodinové takty. V štvrtom takte je aplikovaný prvý vektor a zároveň je posunutá postupnosť o ďalší bit. V piatom hodinovom takte sa aplikuje druhý testovací vektor. Ak by táto dvojica vektorov testovala nejakú poruchu oneskorenia vo vnútri obvodu, ich aplikácia je ideálna a testovacie okolie je také isté ako pri aplikovaní testov pre trvalé poruchy. V ďalšom takte sa načíta ďalší testovací vektor. Ak by dvojica vektorov nemohla byť aplikovaná za sebou, nemôže otestovať poruchu oneskorenia.

Poruchy časovania (nazývané aj poruchy oneskorení) sú modelované rôznymi spôsobmi podľa toho, kde je potrebné oneskorenie merať. Jednotlivé logické členy, z ktorých sa logický obvod navrhuje v danej technológii, má presne definované nominálne oneskorenie. Ak je oneskorenie väčšie ako nominálne oneskorenie logického člena, môže nastať taký stav, že v čase čítania bude na výstupe logického člena iná hodnota v porovnaní s očakávanou bezporuchovou hodnotou. Takúto poruchu je možné detekovať len vtedy, ak je obvod testovaný v jeho funkčnej rýchlosti. V tomto prípade hovoríme o poruche oneskorenia na logickom člene (*gate delay fault*). Pri tomto modeli treba uvažovať o zmene hodnôt z logickej hodnoty 0 na 1 a z logickej hodnoty 1 na 0. Testuje sa porucha prechodu a spomalenia nábežných či dobežných hrán – v literatúre sa tieto poruchy označujú ako poruchy prechodu typu STR (*slow to rise*) and STF (*slow to fall*). Pre detekovanie všetkých typov porúch oneskorení je nutné aplikovať test v podobe postupnosti dvojíc testovacích vektorov, z ktorých prvý má funkciu nastavenia východzej hodnoty, obvykle sa nazýva inicializačný vektor, a druhý má funkciu realizácie preklopenia logických hodnôt, nazývaný ako vektor detekujúci alebo testovací [3].



Obrázok 1: Priebeh aplikácie testu cez testovacie okolie jadra

V mojej práci som sa sústredil na poruchy oneskorení typu prechodov – STR a STF s cieľom navrhnúť ich optimálnu aplikáciu na obvod cez sekvenčné testovacie rozhranie.

3 Návrh metodológie generovania testov pre poruchy prechodov

Z hľadiska jednoduchosti je test často zostavený náhodným generovaním dvojíc testovacích vektorov, pričom pokrytie porúch prechodov je overené poruchovou simuláciou. Ďalším spôsobom je využitie deterministického testu obvodu na pokrytie trvalých porúch, z ktorých sa zostavia dvojice vektorov pokrývajúce poruchu trvalej nuly a následne poruchy trvalej 1 (a opačne) na každom vodiči obvodu. Tento druhý prístup je efektívnejší z hľadiska dĺžky testu. Avšak, takýto test, teda postupnosť dvojíc vektorov, ktoré sú spárené, nie je jednoduché aplikovať na obvod, ktorý je doplnený o testovacie okolie (*wrapper* - štandard IEEE 1500). V najjednoduchšom prípade každý vektor z dvojice bude aplikovaný cez samostatný register *scan*, teda ide o 100 % redundanciu v porovnaní s testovaním trvalých porúch. Táto skutočnosť je motiváciou navrhnúť niečo lepšie – testy pre poruchy prechodu optimálnej dĺžky s aplikáciou cez jeden register *scan* aj za cenu, že by bol dlhší ako pre testovanie trvalých porúch. Pre tento účel bol definovaný M-vektor (*merging vector*).

Definícia 1: Nech (v,u) je dvojica vektorov pre testovanie poruchy prechodu, pričom $v = (v_1,v_2,...v_N)$ je inicializačný a $u = (u_1,u_2,...u_N)$ je detekujúci vektor. Vektor $m = (m_1,m_2,...m_N, m_{N+1})$, pre ktorý platí: $m_1 = v_1$, $m_{N+1} = u_N$ a $m_i = v_i$ pri podmienke $v_i = u_{i-1}$ pre i = 2,3,...N sa nazýva M-vektor.

Príklad M-vektora pre N=8

v_1 = (10xxx1x0)
u_1 = (00x1x101)
m_1 = (100x11101)

Poznámka 1: Ak nastane prípad nekonzistencie logických hodnôt v niektorom mieste v dvojici vektorov, teda platí $v_i \neq u_{i-1}$, kde i = 2,3,...N, potom M-vektor nie je generovaný.

Otázkou je ako efektívne vygenerovať postupnosť dvojíc vektorov pre testovanie porúch prechodov, z ktorých by bolo možné vygenerovať čo najväčší počet M-vektorov. Základom je deterministický test pre trvalé poruchy (poruchy trvalej 0 a trvalej 1) s podmienkou, že pre každú trvalú poruchu je vygenerovaných viacero testovacích vektorov. Použitie deterministického testu je z dôvodu existencie nezávislých logických hodnôt vo vektore (*don't care values*) a vygenerovanie viac ako jedného vektora pre jednotlivú poruchu poskytuje šancu vyberať vhodné vektory pre M-vektor z väčšej množiny. Pre navrhnutý algoritmus generovania testov pre poruchy prechodov uvedieme nasledovné definície.

Definícia 2: Množina Vs_0 (Vs_1) je množinou vektorov, ktoré nastavia hodnotu logickej 0 (logickej 1) na vodiči *s* v obvode.

Definícia 3: Množina Us_0 (Us_1) je množinou vektorov, ktoré detekujú poruchu trvalej 0 (trvalej 1) na vodiči *s* v obvode.

Definícia 4: Pre každý vodič *s* v testovanom obvode sú vygenerované množiny Vs_0, Vs_1, Us_0, Us_1, pre ktoré platí $Us_0 \subset Vs_0$ a $Us_1 \subset Vs_1$.

Algoritmus generovania testu pre poruchy prechodov.

1. Generovanie množín Us_0, Vs_0, Us_1, Vs_1.
2. Generovanie M-vektorov pre každý vodič v obvode:

a. Výber vektora *u* pre poruchu typu STF na vodiči *s* z množiny Us_0 s najväčším počtom nedefinovaných logických hodnôt, ktorý bude inicializačným vektorom pre túto poruchu.
b. Výber vektora *v* z množiny Vs_0 podľa definície 1.
c. Ak takáto kombinácia vektorov neexistuje, potom k tejto poruche nie je jednoznačne priradený M-vektor. Dvojice vektorov s najmenším počtom nekonzistentných stavov sa odkladajú na ďalšiu analýzu.
d. Proces sa opakuje pre poruchu STR a množiny Us_0 a Vs_0.
e. Výber ďalšieho vodiča a proces sa opakuje od bodu a.

3. Stanovenie pokrytia porúch prechodov M-vektormi.
4. Ak je pokrytie nízke alebo nedostačujúce, rieši sa analýza nekonzistentných stavov dvojíc vektorov pri generovaní M-vektorov.
5. Zostavenie mapy nekonzistentných bitov pre každú dvojicu spárovaných vektorov.
6. Určenie optimálneho počtu pridaných bitov a vygenerovanie nových M-vektorov.
7. Určenie pokrytia porúch prechodov.
8. Vytvorenie výslednej bitovej postupnosti pomocou prekrývania M-vektorov.

Na implementáciu algoritmu generovania M-vektorov bol použitý generátor testov Atalanta [5], ktorý generuje deterministické testy pre jednotlivé poruchy. Pre navrhovaný algoritmus je potrebné vygenerovať viacero vektorov (prípadne všetky možné) pre jednotlivé poruchy. Z tohto dôvodu bol generátor Atalanta modifikovaný tak, aby generoval požadovaný počet testovacích vektorov pre každú poruchu s odstránením používania ekvivalencie porúch.

V ideálnom prípade vygenerované M-vektory pokrývajú dostatočný počet porúch oneskorení už v 3. bode algoritmu. Tým je možná aplikácia postupnosti M-vektorov cez jeden štandardný reťazec *scan* a plocha oproti testovaniu trvalých porúch sa nezvyšuje. Avšak pri zložitých, ťažko testovateľných obvodoch, nedochádza k ideálnym prípadom a preto treba navrhnúť modifikovanú architektúru testovacieho okolia pre zabezpečenie testovania porúch prechodov. Takéto metódy sú opísané v nasledujúcej časti, pričom dvojice sú generované vyššie uvedeným algoritmom. V tabuľke 1 sú uvedené výsledky generovania M-vektorov pre obvody ISCAS'85, ktoré túto skutočnosť potvrdzujú.

Tabuľka 1: Pokrytie detekovateľných porúch prechodov po kroku generovania M-vektorov

	c17	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552	priemer
pokrytie	61,76%	53,72%	23,14%	47,35%	8,60%	23,92%	43,64%	33,71%	35,96%	21,51%	21,47%	34,07%

4 Metódy na zvýšenie pokrytia porúch prechodov

Na zvýšenie možnosti vytvorenia M-vektora z príslušných množín vzoriek Vs_x a Us_x boli navrhnuté nasledujúce metódy:

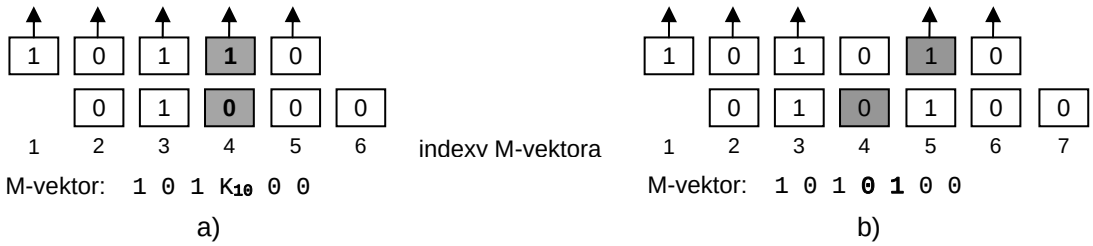
1. Metóda vkladania voľných buniek
2. Metóda preusporiadania buniek

Navrhnuté metódy vkladajú nové bunky alebo preusporiadavajú poradie buniek v reťazci buniek testovacieho okolia jadra. Obe teda zasahujú do pôvodnej štruktúry testovacieho okolia ale len v usporiadaní buniek. Tieto rozšírenia nevyžadujú ďalšie riadiace obvody a tým ďalšie zvyšovanie plochy obvodu. Z tohto dôvodu, ak je niektorá z metód aplikovaná na M-vektor za účelom odstránenia nekonzistencie, jej aplikácia ovplyvní v danom mieste aj všetky ostatné M-vektory.

4.1 Metóda vkladania voľných buniek

Na odstránenie nekonzistencie sa do reťazca buniek v testovacom okolí jadra zapojí ďalšia bunka, ktorá nebude mať výstup vyvedený na primárne vstupy jadra. Jednoduchý príklad takéhoto riešenia je

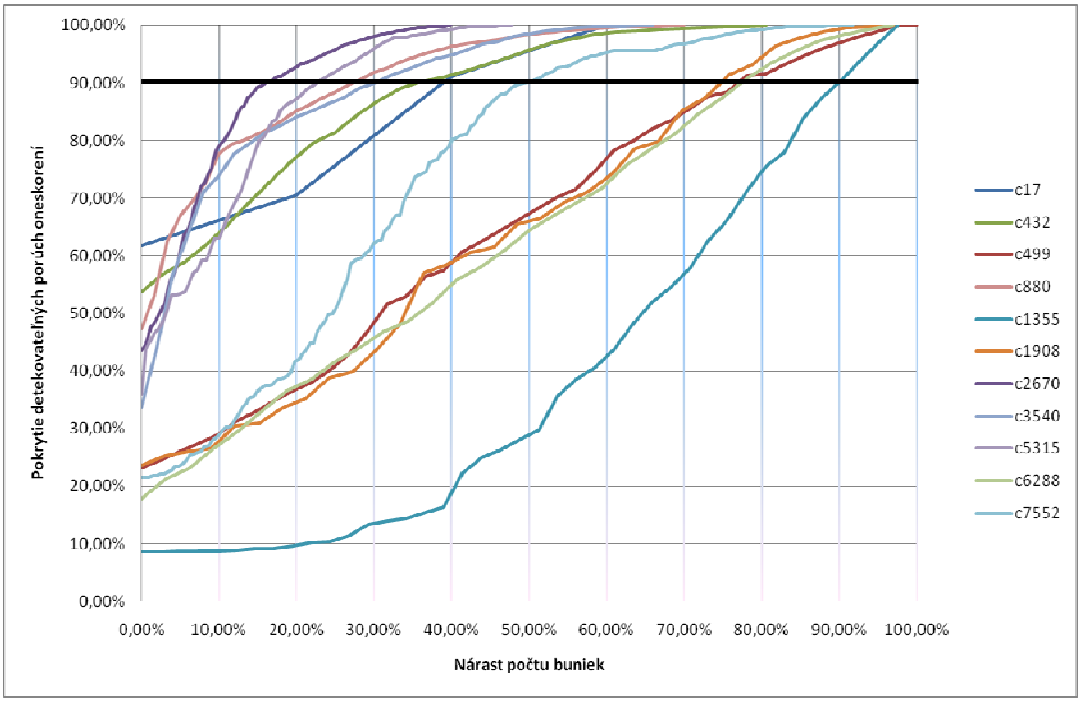
znázornený na obrázku 2. Na obrázku 2a sa v prvom riadku nachádza inicializačný vektor, v druhom detekujúci vektor. Nekonzistencia medzi vektormi je v 4. bite. Pridaním voľnej bunky na toto miesto v posuvnom registri testovacieho okolia bude nekonzistencia odstránená (obrázok 2b). Výsledný M-vektor je oproti pôvodnému M-vektoru o 1 bit dlhší.



Obrázok 2: Príklad vkladania voľnej bunky

Ak do posuvného registra testovacieho okolia jadra je pridaná voľná bunka, všetky M-vektory musia byť na danom mieste rozšírené o jeden bit. Z tohto je zrejmé, že pridaná bunka odstráni prípadné nekonzistencie v ostatných M-vektoroch, ale len na tomto mieste. Metóda vkladania voľných buniek vyberá miesta s najväčším počtom nekonzistencií. Postupným vkladaním buniek sa odstránia nekonzistencie a následne sú vytvorené nové M-vektory. Pri každom vložení bunky je zobrazený nárast pokrytia porúch oneskorení v obvode. Používateľ si nakoniec vyberie počet vložených buniek s ohľadom na celkové pokrytie porúch.

Na obrázku 3 je vidieť nárast pokrytia detekovateľných porúch prechodov pri aplikovaní metódy vkladania voľných buniek. Napríklad pre obvod c2670 stačí do reťazca pridať 18% vstupných buniek a pokrytie porúch oneskorení bude viac ako 90%. Stopercentné pokrytie je dosiahnuté pri 40% náraste počtu buniek. Najhorší variant nárastu buniek v testovacom okolí je 100%, pri ktorom sú zaručene odstránené všetky nekonzistencie a teda pokrytie porúch oneskorení je maximálne. Charakteristiku podobnú tomuto variantu má obvod c3540, ktorý dosiahne 90% pokrytie až pri 90% náraste počtu



Obrázok 3: Závislosť nárastu počtu buniek a pokrytia pri metóde vkladania voľných buniek

vstupných buniek reťazca. Na druhej strane 7 obvodov z 11 dosiahne hranicu 90-percentného pokrytia pri náraste počtu buniek menšom ako 50%.

4.2 Metóda preusporiadania buniek

Metóda odstraňuje nekonzistencie tým, že na miesto nekonzistenzie vloží iný bit M-vektora. Keďže takéto preusporiadanie sa netýka len daného M-vektora, ale všetkých, je potrebné nájsť taký bit, ktorý v žiadnom ďalšom M-vektore nespôsobí inú nekonzistenciu. Výhodou tejto metódy je fakt, že nerastie plocha testovacieho okolia. Jej veľkou nevýhodou však je pomerne veľká závislosť jednotlivých bitov a náročnosť výpočtu. Pravdepodobnosť, že použitím len tejto metódy odstránime všetky nekonzistencie v teste, nie je veľká. Využitie metódy preusporiadania vidím ale v súčasnom použití s metódou vkladania voľných buniek.

5 Zhodnotenie a záver

Cieľom dizertačnej práce je zvýšiť kvalitu testovania vnorených digitálnych jadier s testovacím okolím. Článok sa zaoberá časťou dizertačnej práce, riešiacou generovanie a následnú aplikáciu testov pre poruchy oneskorení cez testovacie okolie jadra. Práca predstavuje algoritmus na generovanie testov porúch prechodov aplikovaných pomocou testovacieho okolia jadra. Cieľom tejto práce bolo navrhnúť metodiku, ktorá by aj znížila plochu testovacieho okolia prispôsobeného na aplikáciu testov porúch oneskorení. Z celkového algoritmu ešte nie je zrealizované generovanie množín inicializačných vektorov Vs_x. Zatiaľ sú ako inicializačné vektory pre nastavenie hodnoty 0 použité množiny pre trvalú poruchu 1 a pre nastavenie inicializačnej hodnoty 1 sú použité vektory pre trvalú poruchu 0. Z definície 4 je zrejmé, že implementácia tejto časti by mohla napomôcť k zvýšeniu percentuálneho pokrytia už pri vytváraní M-vektorov. Okrem toho prebieha implementácia metódy preusporiadania buniek, ako aj záverečného radenia M-vektorov do výslednej postupnosti za účelom jej najkratšej dĺžky. Zatiaľ čo implementácia metódy preusporiadania by mala napomôcť k lepšiemu pokrytiu porúch prechodov bez dodatočného zvýšenia plochy, metóda záverečného zoradenia skráti celkový čas aplikácie testu. Výhodou aplikácie takéhoto testu bude aj fakt, že test pokryje nielen poruchy prechodov, ale aj poruchy trvalé.

Vybraná literatúra

[1] IEEE Standard Testability Method for Embedded Core-based Integrated Circuits, IEEE Std 1500-2005, s. 117, ISBN: 978-2-8318-9481-2

[2] IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE Std 1149.1-2001, s. 200, ISBN: 0-7381-2944-5

[3] Bushnell, M., Agrawal, V.: Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits, 2005, s. 712, ISBN: 0-7923-7991-8

[4] Baláž, M., Gramatová, E., Fischerová, M.: Test Wrapper Application To Embedded Cores as a Java Applet. In: Proceedings of the IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop (DDECS), Poznañ, Poland, April 2003, ISBN 83-7143-557-6, pp. 33-40.

[5] Vermaak, H.J., Kerkhoff, H.G.: Enhanced P1500 Compliant Wrapper suitable for Delay Fault Testing of Embeeded Cores, Proc. of the 8th IEEE European Test Workshop, Maastricht, NL, 2003, str. 257-262.

[4] Gonciari, P., T., Al-Hashimi, B., M., Nicolici, N.: Integrated Test Data Decompression and Core Wrapper Design for Low-Cost System-on-a-Chip Testing. Proc. IEEE International Test Conference (ITC), Baltimore, MD, USA, 2002, str. 64-73.

[5] H.K. Lee and D.S. Ha, "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.